IBM DB2 Information Integrator

ラッパー開発における Java API リファレンス

バージョン 8.2

SC88-9922-00
(英文原典：SC18-9173-00)

IBM DB2 Information Integrator

ラッパー開発における Java API リファレンス

バージョン 8.2

SC88-9922-00
(英文原典：SC18-9173-00)

ご注意！

本書および本書で紹介する製品をご使用になる前に、 205 ページの『特記事項』に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC18-9173-00
IBM DB2 Information Integrator
Java API Reference for Developing Wrappers
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

目次

本書について

本書では、データ・ソースのラッパーを作成する際に使用可能な Java™ API クラスについて解説します。各クラスについて、概要と使用法を記載します。また、各クラスのメソッド (および該当するコンストラクターとデストラクター) の目的、構文、戻り値、および必須入出力引き数をリストします。データ・ソースのラッパーを作成すると、そのデータ・ソースをフェデレーテッド・データベース・システムで使用できるようになります。

本書の対象読者

本書は、IBM® が提供する DB2® Information Integrator で Java API を使用する、DBA およびラッパー開発者を対象としています。

Java クラスおよびメソッド

以下のセクションでは、Java API で使用可能なクラスについて説明します。クラスには、以下のようなものがあります。

- カタログ・クラス
- ラッパー・クラス
- サーバー・クラス
- ユーザー・クラス
- ニックネーム・クラス
- 接続クラス
- 操作クラス
- プランニング・クラス
- データ・クラス
- 例外およびユーティリティー・クラス

各クラスについて、概要と使用法を記載します。また、各クラスのメソッド (および該当するコンストラクターとデストラクター) の目的、構文、戻り値、および必須入出力引き数をリストします。

Java API のカタログ・クラス

以下の表で、Java API の各カタログ・クラスについて説明します。

表 1. カタログ・クラス

クラス名	説明
CatalogInfo	すべてのカタログ・クラスのベース・クラスを処理するクラスで、オプションのリストを管理するインフラストラクチャーを提供します。
ColumnInfo	ニックネームの列のカタログ情報をカプセル化するクラス。このクラスは、列の統計情報を含みます。
NicknameInfo	CREATE NICKNAME および ALTER NICKNAME ステートメントから、列定義を含む、ニックネームの列のカタログ情報をカプセル化するクラス。
ServerInfo	CREATE SERVER ステートメントおよび ALTER SERVER ステートメントから、サーバー・オブジェクトのカタログ情報をカプセル化するクラス。
UserInfo	CREATE USER MAPPING ステートメントおよび ALTER USER MAPPING ステートメントからマップするユーザーのカタログ情報をカプセル化するクラス。
WrapperInfo	CREATE WRAPPER および ALTER WRAPPER ステートメントから、ラッパー・オブジェクトのカタログ情報をカプセル化するクラス。

表 1. カタログ・クラス (続き)

クラス名	説明
CatalogOption	カタログ・オブジェクトのオプションのベース・クラスを処理するクラス。このクラスは、1 つ以上の対の名前値をカプセル化し、次のオプションおよび前のオプションにリンクします。

関連資料:

- 2 ページの『CatalogInfo クラス (Java)』
- 50 ページの『CatalogOption クラス (Java)』
- 5 ページの『ColumnInfo クラス (Java)』
- 22 ページの『NicknameInfo クラス (Java)』
- 33 ページの『ServerInfo クラス (Java)』
- 41 ページの『UserInfo クラス (Java)』
- 45 ページの『WrapperInfo クラス (Java)』

CatalogInfo クラス (Java)

このセクションでは、CatalogInfo クラスについて説明し、このメソッドの詳細について解説します。

CatalogInfo クラスには、コンストラクターは含まれません。

概要

CatalogInfo クラスはすべてのカタログ・クラスのベース・クラスを処理し、オプションのリストを管理するインフラストラクチャーを提供します。

このクラスは、オプション順序またはオプション名のいずれかでアクセス可能な、オプションの順次リストを保管するコレクション・スタイル・クラスです。このクラスは、現行オプションとそれに関連するメソッドを使用します。

ColumnInfo、NicknameInfo、ServerInfo、UserInfo、および WrapperInfo クラスは、CatalogInfo クラスのサブクラスです。

CatalogInfo クラスは、Java API のカタログ・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、CatalogInfo クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 2. CatalogInfo クラスのメソッド

メソッド	説明
addOption	オプションをオプション・チェーンに追加します。

表 2. CatalogInfo クラスのメソッド (続き)

メソッド	説明
dropOption	オプション・チェーンからオプションを削除します。
getFirstOption	チェーンから、最初のオプションを戻します。
getNextOption	チェーンから、次のオプションを戻します。
getOption	指定したオプション名のオプション・オブジェクトを戻します。

addOption メソッド

目的 オプションをオプション・チェーンに追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      int action,
                      java.lang.String optionType,
                      java.lang.String objectName,
                  throws WrapperException
```

パラメーター

表 3. addOption メソッドのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。
action	オプションのアクション・フラグ。オプションの有効アクションは、CatalogOption クラスで指定されます。
optionType	オプションのタイプ。このパラメーター値は、重複オプションである場合にオプション・タイプを識別するため、 SQL1884 エラー・メッセージで使用されるトークンです。
objectName	オプションを所有するオブジェクトの名前。このパラメーター値は、重複オプションである場合にオーナー・オブジェクト名を識別するため、 SQL1884 エラー・メッセージで使用されるトークンです。

戻り値 なし。

スロー オプションがチェーンに既存の場合、またはアクションが無効の場合、WrapperException オブジェクト。

dropOption メソッド

目的 オプション・チェーンからオプションを削除します。

構文

```
public void dropOption(CatalogOption option)
                throws WrapperException
```

パラメーター

表 4. dropOption メソッドのパラメーター

名前	説明
option	削除するオプション。

戻り値 なし。

スロー オプションが NULL の場合、WrapperException オブジェクト。

getFirstOption メソッド

目的 チェーンから、最初のオプションを戻します。

構文

```
public final CatalogOption getFirstOption()
```

パラメーター
なし。

戻り値 最初のオプションを処理する CatalogOption インスタンス。またはオプションが指定されていない場合には NULL。

スロー なし。

getNextOption メソッド

目的 チェーンから、次のオプションを戻します。

構文

```
public final CatalogOption getNextOption(CatalogOption currentOption)
```

パラメーター

表 5. getNextOption メソッドのパラメーター

名前	説明
currentOption	チェーン内の現行オプション。

戻り値 次のオプションを処理する CatalogOption インスタンス。または、オプションがない場合には NULL。

スロー なし。

getOption メソッド

目的 指定したオプション名のオプション・オブジェクトを戻します。

構文

```
public final CatalogOption getOption(java.lang.String optionName)
                              throws WrapperException
```

パラメーター

表 6. getOption メソッドのパラメーター

名前	説明
optionName	オプションの名前。

戻り値 検索済みオプションを処理する CatalogOption インスタンス。または、オプションが検出されない場合には NULL。

スロー オプション名が NULL の場合には WrapperException オブジェクト。

関連資料:

- 50 ページの『CatalogOption クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

ColumnInfo クラス (Java)

このセクションでは、ColumnInfo クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

ColumnInfo クラスは、ニックネームの列のカタログ情報をカプセル化します。このクラスは、列の統計情報を含みます。

public ColumnInfo クラスは、CatalogInfo クラスを拡張します。

ColumnInfo クラスは、Java API のカタログ・クラスの 1 つです。

使用法 ColumnInfo クラスはフェデレーテッド・サーバーによってインスタンス化され、 CREATE NICKNAME または ALTER NICKNAME ステートメントから、あるいはフェデレーテッド・サーバーのシステム・カタログからの情報が含まれます。 このクラスは CREATE NICKNAME ステートメントまたは ALTER NICKNAME ステートメントの処理時に情報が追加された場合に、ラッパーによりインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表では、ColumnInfo クラスのコンストラクターおよびメソッドについて説明されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 7. ColumnInfo クラスのコンストラクター

コンストラクター	説明
ColumnInfo	デフォルトの (空) 列情報オブジェクトを構成します。

ColumnInfo

表 8. ColumnInfo クラスのメソッド

メソッド	説明
addOption	オプションをオプション・チェーンに追加します。
getAvgLength	列の平均長を戻します。
getCodepage1	列の 1 バイト文字セット (SBCS) コード・ページを戻します。
getCodepage2	列の 2 バイト文字セット (DBCS) コード・ページを戻します。
getColCard	列のカーディナリティーを戻します。
getColumnID	列の ID (位置) を戻します。
getColumnName	列の名前を戻します。
getColumnType	列のタイプを戻します。
getDefault	列のデフォルト値を戻します。
getForBitData	列の FOR BIT DATA フラグを戻します。
getHigh2Key	列の 2 番目に大きい値を戻します。
getLow2Key	列の 2 番目に小さい値を戻します。
getNewColumnName	列の名前を変更する ALTER COLUMN または SET COLUMN 文節を含む ALTER COLUMN ステートメントに指定された新しい列名を戻します。
getNextColumn	列チェーン内の次の列を戻します。
getNulls	NULL 許可フラグを戻します。
getOrgLength	列の最大長 (バイト) を戻します。
getOrgScale	列の数値の位取りを戻します。
getTypeName	ローカル列タイプの名前を戻します。
getTypeSchema	ローカル列タイプのスキーマを戻します。
isAvgLengthValid	列の平均長が指定されているかどうかを検証します。
isCodepage1Valid	1 バイト文字セット (SBCS) コード・ページが列に指定されているかどうかを検証します。
isCodepage2Valid	2 バイト文字セット (DBCS) コード・ページが列に指定されているかどうかを検証します。
isColCardValid	カーディナリティー値が列に指定されているかどうかを検証します。
isColumnIDValid	列 ID (位置) が列に指定されているかどうかを検証します。
isColumnNameValid	列に名前が指定されているかどうかを検証します。
isColumnTypeValid	ローカル・タイプが列に指定されているかどうかを検証します。
isDefaultValid	デフォルト値が列に指定されているかどうかを検証します。

表 8. *ColumnInfo* クラスのメソッド (続き)

メソッド	説明
isForBitDataValid	FOR BIT DATA フラグが列に指定されているかどうかを検証します。
isHigh2KeyValid	列で 2 番目に大きい値が指定されているかどうかを検証します。
isLow2KeyValid	列で 2 番目に小さい値が指定されているかどうかを検証します。
isNewColumnNameValid	列の新規名が指定されているかどうかを検証します。
isNullsValid	NULL 許可フラグが列に指定されているかどうかを検証します。
isOrgLengthValid	オリジナル長が列に指定されているかどうかを検証します。
isOrgScaleValid	オリジナルの位取りが列に指定されているかどうかを検証します。
isTypeNameValid	ローカル・タイプ名が列に指定されているかどうかを検証します。
isTypeSchemaValid	ローカル・タイプ・スキーマが列に指定されているかどうかを検証します。
setAvgLength	列の平均長を設定します。
setCodepage1	列の 1 バイト文字セット (SBCS) コード・ページを設定します。
setCodepage2	列の 2 バイト文字セット (DBCS) コード・ページを設定します。
setColCard	列のカーディナリティーを設定します。
setColumnID	列 ID (位置) を設定します。
setColumnName	列の名前を設定します。
setColumnType	列のタイプを設定します。
setDefault	列のデフォルト値を設定します。
setForBitData	列の FOR BIT DATA フラグを設定します。
setHigh2Key	列の 2 番目に大きい値を設定します。
setLow2Key	列の 2 番目に小さい値を設定します。
setNewColumnName	列の新規名を設定します。
setNulls	NULL 許可フラグを設定します。
setOrgLength	列の最大長 (バイト) を設定します。
setOrgScale	列の数値の位取りを設定します。
setTypeName	ローカル列タイプの名前を設定します。
setTypeSchema	ローカル列タイプのスキーマを設定します。

ColumnInfo コンストラクター

目的 デフォルトの (空) 列情報オブジェクトを構成します。

構文

```
public ColumnInfo()
```

パラメーター

なし。

AddOption メソッド

目的 オプションをオプション・チェーンに追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      int action)
     throws WrapperException
```

パラメーター

表 9. AddOption メソッドのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。
action	オプションのアクション・フラグ。オプションの有効アクションは、CatalogOption クラスで指定されます。

戻り値 なし。

スロー オプションがチェーンに既存の場合、またはアクションが無効の場合、WrapperException オブジェクト。

getAvgLength メソッド

目的 列の平均長を戻します。

構文

```
public int getAvgLength()
```

パラメーター

なし。

戻り値 平均列長。

スロー なし。

getCodepage1 メソッド

目的 列の 1 バイト文字セット (SBCS) コード・ページを戻します。

構文

```
public short getCodepage1()
```

パラメーター

なし。

戻り値 1 バイト文字セット (SBCS) コード・ページ。

スロー なし。

getCodepage2 メソッド

目的 列の 2 バイト文字セット (DBCS) コード・ページを戻します。

構文

```
public short getCodepage2()
```

パラメーター
なし。

戻り値 2 バイト文字セット (DBCS) コード・ページ。

スロー なし。

getColCard メソッド

目的 列のカーディナリティーを戻します。

構文

```
public long getColCard()
```

パラメーター
なし。

戻り値 列カーディナリティー。

スロー なし。

getColumnID メソッド

目的 列 ID を戻します。列 ID は列の位置を表します。列位置の値は 0 から始まります。

構文

```
public short getColumnID()
```

パラメーター
なし。

戻り値 列の ID (位置)。

スロー なし。

getColumnName メソッド

目的 列の名前を戻します。

構文

```
public java.lang.String getColumnName()
```

パラメーター
なし。

戻り値 列名。

スロー なし。

getColumnType メソッド

目的 列のタイプを戻します。

構文

```
public java.lang.String getColumnType()
```

パラメーター
なし。

戻り値 列タイプ。

スロー なし。

getDefault メソッド

目的 列のデフォルト値を戻します。

構文

```
public java.lang.String getDefault()
```

パラメーター
なし。

戻り値 デフォルト値。

スロー なし。

getForBitData メソッド

目的 列の FOR BIT DATA フラグを戻します。

構文

```
public boolean getForBitData()
```

パラメーター
なし。

戻り値 列に FOR BIT DATA フラグ・セットがあるかどうかを示す値。

スロー なし。

getHigh2Key メソッド

目的 列の 2 番目に大きい値を戻します。

構文

```
public java.lang.String getHigh2Key()
```

パラメーター
なし。

戻り値 high2Key 値。

スロー なし。

getLow2Key メソッド

目的 列の 2 番目に小さい値を戻します。

構文

```
public java.lang.String getLow2Key()
```

パラメーター
なし。

戻り値 low2Key 値。

スロー なし。

getNewColumnName メソッド

目的 列の名前を変更する ALTER (または SET) COLUMN 文節を含む ALTER COLUMN ステートメントに指定された新しい列名を戻します。

構文

```
public java.lang.String getNewColumnName()
```

パラメーター
なし。

戻り値 新規列名。

スロー なし。

getNextColumn メソッド

目的 列チェーン内の次の列を戻します。

構文

```
public ColumnInfo getNextColumn()
```

パラメーター
なし。

戻り値 チェーン内の次の列。

スロー なし。

getNulls メソッド

目的 NULL 許可フラグを戻します。

構文

```
public boolean getNulls()
```

パラメーター
なし。

戻り値 NULL 許可フラグ。

スロー なし。

getOrgLength メソッド

目的 列の最大長 (バイト) を戻します。

構文

```
public int getOrgLength()
```

パラメーター
なし。

戻り値 列長。

スロー なし。

getOrgScale メソッド

目的 列の数値の位取りを戻します。

構文

```
public short getOrgScale()
```

パラメーター

なし。

戻り値 列の位取り。

スロー なし。

getTypeName メソッド

目的 ローカル列タイプの名前を戻します。

構文

```
public java.lang.String getTypeName()
```

パラメーター

なし。

戻り値 タイプの名前。

スロー なし。

getTypeSchema メソッド

目的 ローカル列タイプのスキーマを戻します。

構文

```
public java.lang.String getTypeSchema()
```

パラメーター

なし。

戻り値 タイプのスキーマ名。

スロー なし。

isAvgLengthValid メソッド

目的 列の平均長が指定されているかどうかを検証します。

構文

```
public boolean isAvgLengthValid()
```

パラメーター

なし。

戻り値 平均長値が指定されている場合には真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isCodepage1Valid メソッド

目的 1 バイト文字セット (SBCS) コード・ページが列に指定されているかどうかを検証します。

構文

```
public boolean isCodepage1Valid()
```

パラメーター
なし。

戻り値 1 バイト文字セット値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isCodepage2Valid メソッド

目的 2 バイト文字セット (DBCS) コード・ページが列に指定されているかどうかを検証します。

構文

```
public boolean isCodepage2Valid()
```

パラメーター
なし。

戻り値 2 バイト文字セット値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isColCardValid メソッド

目的 カーディナリティー値が列に指定されているかどうかを検証します。

構文

```
public boolean isColCardValid()
```

パラメーター
なし。

戻り値 カーディナリティー値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isColumnIDValid メソッド

目的 列 ID (位置) が列に指定されているかどうかを検証します。

構文

```
public boolean isColumnIDValid()
```

パラメーター
なし。

戻り値 列 ID (位置) 値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isColumnNameValid メソッド

目的 列に名前が指定されているかどうかを検証します。

構文

```
public boolean isColumnNameValid()
```

パラメーター
なし。

戻り値 列名値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isColumnTypeValid メソッド

目的 ローカル・タイプが列に指定されているかどうかを検証します。

構文

```
public boolean isColumnTypeValid()
```

パラメーター
なし。

戻り値 ローカル・タイプ値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isDefaultValid メソッド

目的 デフォルト値が列に指定されているかどうかを検証します。

構文

```
public boolean isDefaultValid()
```

パラメーター
なし。

戻り値 デフォルト値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isForBitDataValid メソッド

目的 FOR BIT DATA フラグが列に指定されているかどうかを検証します。

構文

```
public boolean isForBitDataValid()
```

パラメーター
なし。

戻り値 FOR BIT DATA フラグ値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isHigh2KeyValid メソッド

目的 列で 2 番目に大きい値が指定されているかどうかを検証します。

構文

```
public boolean isHigh2KeyValid()
```

パラメーター
なし。

戻り値 2 番目に大きい値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isLow2KeyValid メソッド

目的 列で 2 番目に小さい値が指定されているかどうかを検証します。

構文

```
public boolean isLow2KeyValid()
```

パラメーター
なし。

戻り値 2 番目に小さい値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isNewColumnNameValid メソッド

目的 列の新規名が指定されているかどうかを検証します。

構文

```
public boolean isNewColumnNameValid()
```

パラメーター
なし。

戻り値 新規名前値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isNullsValid メソッド

目的 NULL 許可フラグが列に指定されているかどうかを検証します。

構文

```
public boolean isNullsValid()
```

パラメーター
なし。

戻り値 NULL 許可フラグ値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isOrgLengthValid メソッド

目的 オリジナル長が列に指定されているかどうかを検証します。

構文

```
public boolean isOrgLengthValid()
```

パラメーター
なし。

戻り値 オリジナル長さが指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isOrgScaleValid メソッド

目的 オリジナルの位取りが列に指定されているかどうかを検証します。

構文

```
public boolean isOrgScaleValid()
```

パラメーター
なし。

戻り値 オリジナルの位取りが指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isTypeNameValid メソッド

目的 ローカル・タイプ名が列に指定されているかどうかを検証します。

構文

```
public boolean isTypeNameValid()
```

パラメーター
なし。

戻り値 ローカル・タイプ名値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isTypeSchemaValid メソッド

目的 ローカル・タイプ・スキーマが列に指定されているかどうかを検証します。

構文

```
public boolean isTypeSchemaValid()
```

パラメーター
なし。

戻り値 ローカル・タイプ・スキーマ値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

setAvgLength メソッド

目的 列の平均長を設定します。

構文

```
public void setAvgLength(int avgLength)
```

パラメーター

表 10. setAvgLength メソッドのパラメーター

名前	説明
avgLength	設定する長さ。

戻り値 なし。

スロー なし。

setCodepage1 メソッド

目的 列の 1 バイト文字セット (SBCS) コード・ページを設定します。

構文

```
public void setCodepage1(short codepage1)
```

パラメーター

表 11. setCodepage1 メソッドのパラメーター

名前	説明
codepage1	設定する 1 バイト文字セット (SBCS) コード・ページ。

戻り値 なし。

スロー なし。

setCodepage2 メソッド

目的 列の 2 バイト文字セット (DBCS) コード・ページを設定します。

構文

```
public void setCodepage2(short codepage2)
```

パラメーター

表 12. setCodepage1 メソッドのパラメーター

名前	説明
codepage2	設定する 2 バイト文字セット (DBCS) コード・ページ。

戻り値 なし。

スロー なし。

setColCard メソッド

目的 列のカーディナリティーを設定します。ラッパーは、CREATE NICKNAME ステートメントまたは ALTER NICKNAME ステートメントの処理中に、列のカーディナリティー (既知の場合) を設定します。 DB2 Universal Database オプティマイザーは、最適パフォーマンス・プランの作成時にこの情報を使用します。固有の値を持つ列 (重複なし) については、列のカーディナリティーとニックネームのカーディナリティーが同じである必要があり

ます。列のカーディナリティーがニックネームのカーディナリティーより大きい場合は、 DB2 Universal Database オプティマイザーがエラーを生成します。

構文

```
public void setColCard(long colCard)
```

パラメーター

表 13. setColCard メソッドのパラメーター

名前	説明
colCard	設定する値。

戻り値 なし。

スロー なし。

setColumnID メソッド

目的 列 ID を設定します。列 ID は列の位置を表します。列位置の値は 0 から始まります。

構文

```
public void setColumnID(short columnID)
```

パラメーター

表 14. setColumnID メソッドのパラメーター

名前	説明
columnID	列の ID (位置)。

戻り値 なし。

スロー なし。

setColumnName メソッド

目的 列の名前を設定します。

構文

```
public void setColumnName(java.lang.String columnName)
```

パラメーター

表 15. setColumnName メソッドのパラメーター

名前	説明
columnName	設定する名前。

戻り値 なし。

スロー なし。

setColumnType メソッド

目的 列のタイプを設定します。

構文

```
public void setColumnType(java.lang.String columnType)
```

パラメーター

表 16. setColumnType メソッドのパラメーター

名前	説明
columnType	列タイプ。

戻り値 なし。

スロー なし。

setDefault メソッド

目的 列のデフォルト値を設定します。

構文

```
public void setDefault(java.lang.String defaultValue)
```

パラメーター

表 17. setDefault メソッドのパラメーター

名前	説明
defaultValue	デフォルト値。

戻り値 なし。

スロー なし。

setForBitData メソッド

目的 列の FOR BIT DATA フラグを設定します。

構文

```
public void setForBitData(boolean forBitData)
```

パラメーター

表 18. setForBitData メソッドのパラメーター

名前	説明
forBitData	FOR BIT DATA フラグ。

戻り値 なし。

スロー なし。

setHigh2Key メソッド

目的 列の 2 番目に大きい値を設定します。ラッパーは、CREATE NICKNAME ステートメントまたは ALTER NICKNAME ステートメントの処理中に、列の 2 番目に大きい値を設定できます。 DB2 Universal Database オプティマイザーは、照会最適化プランの作成時に、この 2 番目に大きい値または最大値を使用できます。

構文

```
public void setHigh2Key(java.lang.String high2Key)
```

パラメーター

表 19. setHigh2Key メソッドのパラメーター

名前	説明
high2Key	設定する値。

戻り値 なし。

スロー なし。

setLow2Key メソッド

目的 列の 2 番目に小さい値を設定します。ラッパーは、CREATE NICKNAME ステートメントまたは ALTER NICKNAME ステートメントの処理中に、列の 2 番目に小さい値を設定できます。 DB2 Universal Database オプティマイザーは、照会最適化プランの作成時に、この 2 番目に小さい値または最小値を使用できます。

構文

```
public void setLow2Key(java.lang.String low2Key)
```

パラメーター

表 20. setLow2Key メソッドのパラメーター

名前	説明
low2Key	設定する値。

戻り値 なし。

スロー なし。

setNewColumnName メソッド

目的 列の新規名を設定します。列の名前を変更する ALTER または SET COLUMN 文節を含む ALTER COLUMN ステートメントに指定された新しい列名を設定します。

構文

```
public void setNewColumnName(java.lang.String newColumnName)
```

パラメーター

表 21. setNewColumnName メソッドのパラメーター

名前	説明
newColumnName	設定する新規列名。

戻り値 なし。

スロー なし。

setNulls メソッド

目的 NULL 許可フラグを設定します。

構文

```
public void setNulls(boolean nulls)
```

パラメーター

表 22. setNulls メソッドのパラメーター

名前	説明
nulls	真は NULL 値を許可します。偽は NULL 値を許可しません。

戻り値 なし。

スロー なし。

setOrgLength メソッド

目的 列の最大長 (バイト) を設定します。

構文

```
public void setOrgLength(int orgLength)
```

パラメーター

表 23. setOrgLength メソッドのパラメーター

名前	説明
orgLength	設定する長さ。

戻り値 なし。

スロー なし。

setOrgScale メソッド

目的 列の数値の位取りを設定します。

構文

```
public void setOrgScale(short orgScale)
```

パラメーター

表 24. setOrgScale メソッドのパラメーター

名前	説明
orgScale	設定する位取り。

戻り値 なし。

スロー なし。

setTypeName メソッド

目的 ローカル列タイプの名前を設定します。

構文

```
public void setTypeName(java.lang.String typeName)
```

パラメーター

表 25. setTypeName メソッドのパラメーター

名前	説明
typeName	タイプの名前。

戻り値 なし。

スロー なし。

setTypeSchema メソッド

目的 ローカル列タイプのスキーマを設定します。

構文

```
public void setTypeSchema(java.lang.String typeSchema)
```

パラメーター

表 26. setTypeSchema メソッドのパラメーター

名前	説明
typeSchema	タイプのスキーマ名。

戻り値 なし。

スロー なし。

継承されるメソッド

以下のメソッドは、CatalogInfo クラスから継承され、ColumnInfo クラスと共に使用できます。

- addOption
- dropOption
- getFirstOption
- getNextOption
- getOption

関連資料:

- 2 ページの『CatalogInfo クラス (Java)』
- 50 ページの『CatalogOption クラス (Java)』

NicknameInfo クラス (Java)

このセクションでは、NicknameInfo クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

CREATE NICKNAME および ALTER NICKNAME ステートメントから、列定義を含む、ニックネームの列のカタログ情報をカプセル化するクラス。

NicknameInfo クラスは、CatalogInfo クラスを拡張します。

NicknameInfo クラスは、Java API のカタログ・クラスの 1 つです。

使用法 NicknameInfo クラスはフェデレーテッド・サーバーによってインスタンス化され、 CREATE NICKNAME または ALTER NICKNAME ステートメントからの情報、あるいはフェデレーテッド・サーバーのシステム・カタログからの情報が含まれます。 このクラスは CREATE NICKNAME ステートメントまたは ALTER NICKNAME ステートメントの処理時に情報が追加された場合に、ラッパーによりインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表では、NicknameInfo クラスのコンストラクターおよびメソッドについて説明されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 27. NicknameInfo クラスのコンストラクター

コンストラクター	説明
NicknameInfo	デフォルトの (空) ニックネーム情報オブジェクトを構成します。

表 28. NicknameInfo クラスのメソッド

メソッド	説明
addOption	オプションをオプション・チェーンに追加します。
getCard	カーディナリティー値を戻します。
getColumn	指定された名前の列を戻します。
getColumnWithRemoteColumnName	指定されたリモート列名の列を戻します。
getFirstColumn	最初の列情報オブジェクトを戻します。
getFPages	ニックネームの fpages 統計を戻します。
getNextColumn	次の列情報オブジェクトを戻します。
getNickname	ニックネームの名前を戻します。
getNPages	ニックネームの npages 統計を戻します。
getNumColumns	列の数を戻します。
getOverflow	ニックネームのオーバーフロー統計を戻します。
getSchema	ニックネームのローカル・スキーマ名を戻します。
getServerName	ニックネームのデータ・ソース・サーバー名を戻します。
insertColumn	列 ID フィールドで指定された位置に、列情報オブジェクトを挿入します。
isCardValid	カーディナリティー値が指定されているかどうかを検証します。
isFPagesValid	fpages 統計が指定されているかどうかを検証します。

表 28. NicknameInfo クラスのメソッド (続き)

メソッド	説明
isNicknameValid	ニックネーム名が指定されているかどうかを検証します。
isNPagesValid	npages 統計が指定されているかどうかを検証します。
isOverflowValid	オーバーフロー統計が指定されているかどうかを検証します。
isSchemaValid	スキーマ名が指定されているかどうかを検証します。
isServerNameValid	データ・ソース・サーバー名が指定されているかどうかを検証します。
setCard	カーディナリティー値を設定します。
setFPages	ニックネームの fpages 統計を設定します。
setNickname	ニックネームの名前を設定します。
setNPages	ニックネームの npages 統計を設定します。
setOverflow	ニックネームのオーバーフロー統計を設定します。
setSchema	ニックネームのローカル・スキーマ名を設定します。
setServerName	ニックネームのデータ・ソース・サーバー名を設定します。

NicknameInfo コンストラクター

目的 デフォルトの (空) ニックネーム情報オブジェクトを構成します。

構文

```
public NicknameInfo()
```

パラメーター

なし。

addOption メソッド

目的 オプションをオプション・チェーンに追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      int action)
               throws WrapperException
```

パラメーター

表 29. addOption メソッドのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。

表 29. addOption メソッドのパラメーター (続き)

名前	説明
action	オプションのアクション・フラグ。オプションの有効アクションは、CatalogOption クラスで指定されます。

戻り値 なし。

スロー オプションがチェーンに既存の場合、またはアクションが無効の場合、WrapperException オブジェクト。

getCard メソッド

目的 カーディナリティー値を戻します。

構文

```
public long getCard()
```

パラメーター

なし。

戻り値 カーディナリティー値。

スロー なし。

getColumn メソッド

目的 指定された名前の列を戻します。

構文

```
public ColumnInfo getColumn(java.lang.String name)
```

パラメーター

表 30. getColumn メソッドのパラメーター

名前	説明
name	戻される列の名前。

戻り値 列記述子。

スロー なし。

getColumnWithRemoteColumnName メソッド

目的 指定されたリモート列名の列を戻します。

構文

```
public ColumnInfo getColumnWithRemoteColumnName(java.lang.String remoteColumnName)
                                                throws WrapperException
```

パラメーター

表 31. getColumnWithRemoteColumnName メソッドのパラメーター

名前	説明
remoteColumnName	戻される列のリモート名。

戻り値 列記述子。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

getFirstColumn メソッド

目的 最初の列情報オブジェクトを戻します。

構文

```
public ColumnInfo getFirstColumn()
```

パラメーター

なし。

戻り値 最初の列記述子。

スロー なし。

getFPages メソッド

目的 ニックネームの fpages 統計を戻します。

構文

```
public int getFPages()
```

パラメーター

なし。

戻り値 fpages 統計値。

スロー なし。

getNextColumn メソッド

目的 次の列を戻します。

構文

```
public ColumnInfo getNextColumn(ColumnInfo currentColumn)
```

パラメーター

表 32. getNextColumn メソッドのパラメーター

名前	説明
currentColumn	現行の列記述子。

戻り値 次の列記述子。

スロー なし。

getNickname メソッド

目的 ニックネームの名前を戻します。

構文

```
public java.lang.String getNickname()
```

パラメーター

なし。

戻り値 名前。

スロー なし。

getNPages メソッド

目的 ニックネームの npages 統計を戻します。

構文

```
public int getNPages()
```

パラメーター
なし。

戻り値 npages 統計値。

スロー なし。

getNumColumns メソッド

目的 列の数を戻します。

構文

```
public int getNumColumns()
```

パラメーター
なし。

戻り値 列の数。

スロー なし。

getOverflow メソッド

目的 ニックネームのオーバーフロー統計を戻します。

構文

```
public int getOverflow()
```

パラメーター
なし。

戻り値 オーバーフロー統計値。

スロー なし。

getSchema メソッド

目的 ニックネームのローカル・スキーマ名を戻します。

構文

```
public java.lang.String getSchema()
```

パラメーター
なし。

戻り値 ローカル・スキーマ名。

スロー なし。

getServerName メソッド

目的 ニックネームのデータ・ソース・サーバー名を戻します。

構文

```
public java.lang.String getServerName()
```

パラメーター

なし。

戻り値 データ・ソース・サーバー名。

スロー なし。

insertColumn メソッド

目的 列 ID フィールドで指定された位置に、列情報オブジェクトを挿入します。

構文

```
public void insertColumn(ColumnInfo newColumn)
                    throws WrapperException
```

パラメーター

表 33. insertColumn メソッドのパラメーター

名前	説明
newColumn	挿入する列情報オブジェクト。

戻り値 なし。

スロー ColumnInfo オブジェクトが NULL の場合には、WrapperException オブジェクト。

isCardValid メソッド

目的 カーディナリティー値が指定されているかどうかを検証します。

構文

```
public boolean isCardValid()
```

パラメーター

なし。

戻り値 カーディナリティー値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isFPagesValid メソッド

目的 fpages 統計が指定されているかどうかを検証します。

構文

```
public boolean isFPagesValid()
```

パラメーター

なし。

戻り値 統計が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isNicknameValid メソッド

目的 ニックネーム名が指定されているかどうかを検証します。

構文

```
public boolean isNicknameValid()
```

パラメーター
なし。

戻り値 名前が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isNPagesValid メソッド

目的 npages 統計が指定されているかどうかを検証します。

構文

```
public boolean isNPagesValid()
```

パラメーター
なし。

戻り値 統計が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isOverflowValid メソッド

目的 オーバーフロー統計が指定されているかどうかを検証します。

構文

```
public boolean isOverflowValid()
```

パラメーター
なし。

戻り値 オーバーフロー統計値が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isSchemaValid メソッド

目的 スキーマ名が指定されているかどうかを検証します。

構文

```
public boolean isSchemaValid()
```

パラメーター
なし。

戻り値 スキーマ名が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

isServerNameValid メソッド

目的 データ・ソース・サーバー名が指定されているかどうかを検証します。

構文

```
public boolean isServerNameValid()
```

パラメーター
なし。

戻り値 データ・ソース・サーバー名が指定されている場合には、真の値。指定されていない場合には、戻り値は偽です。

スロー なし。

setCard メソッド

目的 カーディナリティー値を設定します。

構文

```
public void setCard(long card)
```

パラメーター

表 34. setCard メソッドのパラメーター

名前	説明
card	カーディナリティー値。

戻り値 なし。

スロー なし。

setFPages メソッド

目的 ニックネームの fpages 統計を設定します。

構文

```
public void setFPages(int fPages)
```

パラメーター

表 35. setFPages メソッドのパラメーター

名前	説明
fPages	fpages 統計値。

戻り値 なし。

スロー なし。

setNickname メソッド

目的 ニックネームの名前を設定します。

構文

```
public void setNickname(java.lang.String nickname)
```

パラメーター

表 36. setNickname メソッドのパラメーター

名前	説明
nickname	ニックネームの名前。

戻り値 なし。

スロー なし。

setNPages メソッド

目的 ニックネームの npages 統計を設定します。

構文

```
public void setNPages(int nPages)
```

パラメーター

表 37. setNPages メソッドのパラメーター

名前	説明
nPages	npages 統計値。

戻り値 なし。

スロー なし。

setOverflow メソッド

目的 ニックネームのオーバーフロー統計を設定します。

構文

```
public void setOverflow(int overflow)
```

パラメーター

表 38. setOverflow メソッドのパラメーター

名前	説明
overflow	オーバーフロー統計値。

戻り値 なし。

スロー なし。

setSchema メソッド

目的 ニックネームのローカル・スキーマ名を設定します。

構文

```
public void setSchema(java.lang.String schema)
```

パラメーター

表 39. setSchema メソッドのパラメーター

名前	説明
schema	ローカル・スキーマ名。

戻り値 なし。

スロー なし。

setServerName メソッド

目的 ニックネームのデータ・ソース・サーバー名を設定します。

構文

```
public void setServerName(java.lang.String serverName)
```

パラメーター

表 40. setServerName メソッドのパラメーター

名前	説明
serverName	データ・ソース・サーバー名。

戻り値 なし。

スロー なし。

継承されるメソッド

以下のメソッドは、CatalogInfo クラスから継承され、NicknameInfo クラスと共に使用できます。

- addOption
- dropOption
- getFirstOption
- getNextOption
- getOption

関連資料:

- 2 ページの『CatalogInfo クラス (Java)』
- 50 ページの『CatalogOption クラス (Java)』
- 5 ページの『ColumnInfo クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

ServerInfo クラス (Java)

このセクションでは、ServerInfo クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

ServerInfo クラスは、CREATE SERVER ステートメントおよび ALTER SERVER ステートメントから、データ・ソース・サーバー・オブジェクトのカタログ情報をカプセル化し、CatalogInfo クラスを拡張します。

ServerInfo クラスは、Java API のカタログ・クラスの 1 つです。

使用法 ServerInfo クラスはフェデレーテッド・サーバーによりインスタンス化され、 CREATE SERVER ステートメントまたは ALTER SERVER ステートメント、あるいはフェデレーテッド・サーバーのシステム・カタログからの情報を持ちます。また、このクラスは CREATE SERVER ステートメントまたは ALTER SERVER ステートメントの処理時に情報が追加された場合に、ラッパーによりインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

次の表に、ServerInfo クラスのコンストラクターおよびメソッドを示します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 41. ServerInfo クラスのコンストラクター

コンストラクター	説明
ServerInfo	デフォルトの (空) サーバー情報オブジェクトを構成します。
ServerInfo	完全に初期化されたサーバー情報オブジェクトを構成します。

表 42. ServerInfo クラスのメソッド

メソッド	説明
addOption	オプション・チェーンに単一値オプションを追加します。
addOption	オプション・チェーンに複数値オプションを追加します。
dropOption	オプション・チェーンからオプションを削除します。
dropOption	複数値オプションから値を削除します。残っている値がない場合には、オプション全体を削除してください。
getAuthID	サーバーの許可 ID を戻します。
getPassword	サーバーのパスワードを戻します。
getServerName	サーバーの名前を戻します。

表 42. *ServerInfo* クラスのメソッド (続き)

メソッド	説明
getType	サーバーのタイプを戻します。
getVersion	バージョン・ストリングを戻します。
getWrapperName	データ・ソース・サーバーが属するラッパー・ライブラリーの名前を戻します。
isAuthIDValid	許可 ID 値が指定されているかどうかを検証します。
isNameValid	名前値が指定されているかどうかを検証します。
isPasswordValid	パスワード値が指定されているかどうかを検証します。
isTypeValid	タイプ値が指定されているかどうかを検証します。
isVersionValid	バージョン値が指定されているかどうかを検証します。
isWrapperNameValid	ラッパーの名前値が指定されているかどうかを検証します。
setAuthID	データ・ソース・サーバーの許可 ID を設定します。
setPassword	データ・ソース・サーバーのパスワードを設定します。
setServerName	データ・ソース・サーバー名を設定します。
setType	データ・ソース・サーバーのタイプを設定します。
setVersion	バージョン・ストリングを設定します。
setWrapperName	データ・ソース・サーバーが属するラッパー名の名前を戻します。

ServerInfo コンストラクター

目的 デフォルトの (空) サーバー情報オブジェクトを構成します。

構文

```
public ServerInfo()
```

パラメーター

なし。

ServerInfo コンストラクター

目的 完全に初期化されたサーバー情報オブジェクトを構成します。

構文

```
public ServerInfo(java.lang.String name,
                  java.lang.String type,
                  java.lang.String version,
                  java.lang.String wrapper)
```

パラメーター

表 43. ServerInfo コンストラクターのパラメーター

名前	説明
name	サーバー名。
type	サーバー・タイプ。
version	バージョン。
wrapper	サーバーが属するラッパー名。

addOption メソッド

目的 オプション・チェーンに単一値オプションを追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      int action)
               throws WrapperException
```

パラメーター

表 44. addOption コンストラクターのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。
action	オプションのアクション・フラグ。

戻り値 なし。

スロー オプションがチェーンに既存の場合、またはアクションが無効の場合、WrapperException オブジェクト。

addOption メソッド

目的 オプション・チェーンに複数値オプションを追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      java.sql.Timestamp timestamp,
                      java.lang.String valueID,
                      int action)
               throws WrapperException
```

パラメーター

表 45. addOption コンストラクターのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。
timestamp	値のタイム・スタンプ。
valueID	値の ID。
action	オプションのアクション・フラグ。

戻り値 なし。

スロー 指定された ID が重複している場合、あるいはアクションが無効である場合、WrapperException オブジェクト。

dropOption メソッド

目的 オプション・チェーンからオプションを削除します。

構文

```
public void dropOption(CatalogOption option)
                throws WrapperException
```

パラメーター

表 46. dropOption メソッドのパラメーター

名前	説明
option	削除するオプション。

戻り値 なし。

スロー オプション・オブジェクトが NULL の場合、WrapperException オブジェクト。

dropOption メソッド

目的 複数値オプションから値を削除します。残っている値がない場合には、オプション全体を削除してください。

構文

```
public void dropOption(CatalogOption option,
                       java.lang.String valueID)
                throws WrapperException
```

パラメーター

表 47. dropOption メソッドのパラメーター

名前	説明
option	オプション。
valueID	削除する値の ID。

戻り値 なし。

スロー オプション・オブジェクトが NULL の場合、または値 ID が見つからない場合は、WrapperException オブジェクト。

getAuthID メソッド

目的 データ・ソース・サーバーの許可 ID を戻します。

構文

```
public java.lang.String getAuthID()
```

パラメーター

なし。

戻り値 許可 ID。

スロー なし。

getPassword メソッド

目的 データ・ソース・サーバーのパスワードを戻します。

構文

```
public java.lang.String getPassword()
```

パラメーター
なし。

戻り値 パスワード。

スロー なし。

getServerName メソッド

目的 データ・ソース・サーバーの名前を戻します。

構文

```
public java.lang.String getServerName()
```

パラメーター
なし。

戻り値 サーバー名。

スロー なし。

getType メソッド

目的 データ・ソース・サーバーのタイプを戻します。

構文

```
public java.lang.String getType()
```

パラメーター
なし。

戻り値 サーバー・タイプ。

スロー なし。

getVersion メソッド

目的 バージョン・ストリングを戻します。

構文

```
public java.lang.String getVersion()
```

パラメーター
なし。

戻り値 バージョン。

スロー なし。

getWrapperName メソッド

目的 データ・ソース・サーバーが属するラッパー・ライブラリーの名前を戻します。

構文

```
public java.lang.String getWrapperName()
```

パラメーター
なし。

戻り値 ラッパー名。

スロー なし。

isAuthIDValid メソッド

目的 許可 ID 値が指定されているかどうかを検証します。

構文

```
public boolean isAuthIDValid()
```

パラメーター
なし。

戻り値 許可 ID が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isNameValid メソッド

目的 名前値が指定されているかどうかを検証します。

構文

```
public boolean isNameValid()
```

パラメーター
なし。

戻り値 名前値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isPasswordValid メソッド

目的 パスワード値が指定されているかどうかを検証します。

構文

```
public boolean isPasswordValid()
```

パラメーター
なし。

戻り値 パスワード値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isTypeValid メソッド

目的 タイプ値が指定されているかどうかを検証します。

構文

```
public boolean isTypeValid()
```

パラメーター
なし。

戻り値 タイプ値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isVersionValid メソッド

目的 バージョン値が指定されているかどうかを検証します。

構文

```
public boolean isVersionValid()
```

パラメーター
なし。

戻り値 バージョン値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isWrapperNameValid メソッド

目的 ラッパーの名前値が指定されているかどうかを検証します。

構文

```
public boolean isWrapperNameValid()
```

パラメーター
なし。

戻り値 ラッパー名値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

setAuthID メソッド

目的 データ・ソース・サーバーの許可 ID を設定します。

構文

```
public void setAuthID(java.lang.String authID)
```

パラメーター

表 48. setAuthID メソッドのパラメーター

名前	説明
authID	許可 ID。

戻り値 なし。

スロー なし。

setPassword メソッド

目的 データ・ソース・サーバーのパスワードを設定します。

構文

```
public void setPassword(java.lang.String password)
```

パラメーター

表 49. setPassword メソッドのパラメーター

名前	説明
password	データ・ソース・サーバーのパスワード。

戻り値 なし。

スロー なし。

setServerName メソッド

目的 データ・ソース・サーバー名を設定します。

構文

```
public void setServerName(java.lang.String name)
```

パラメーター

表 50. setServerName メソッドのパラメーター

名前	説明
name	データ・ソース・サーバー名。

戻り値 なし。

スロー なし。

setType メソッド

目的 データ・ソース・サーバーのタイプを設定します。

構文

```
public void setType(java.lang.String type)
```

パラメーター

表 51. setType メソッドのパラメーター

名前	説明
type	データ・ソース・サーバー・タイプ。

戻り値 なし。

スロー なし。

setVersion メソッド

目的 バージョン・ストリングを設定します。

構文

```
public void setVersion(java.lang.String version)
```

パラメーター

表 52. setVersion メソッドのパラメーター

名前	説明
version	バージョン。

戻り値 なし。

スロー なし。

setWrapperName メソッド

目的 データ・ソース・サーバーが属するラッパー名の名前を戻します。

構文

```
public void setWrapperName(java.lang.String name)
```

パラメーター

表 53. setWrapperName メソッドのパラメーター

名前	説明
name	ラッパー名。

戻り値 なし。

スロー なし。

継承されるメソッド

以下のメソッドは、CatalogInfo クラスから継承され、ServerInfo クラスとともに使用できます。

- addOption
- getFirstOption
- getNextOption
- getOption

関連資料:

- 2 ページの『CatalogInfo クラス (Java)』
- 50 ページの『CatalogOption クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UserInfo クラス (Java)

このセクションでは、UserInfo クラスについて説明し、そのコンストラクターおよびメソッドについて解説します。

概要

UserInfo クラスは、CREATE USER MAPPING ステートメントおよび ALTER USER MAPPING ステートメントからマップするユーザーのカタログ情報をカプセル化します。

public UserInfo クラスは CatalogInfo クラスを拡張します。

UserInfo クラスは、Java API のカタログ・クラスの 1 つです。

使用法 UserInfo クラスはフェデレーテッド・サーバーによりインスタンス化され、CREATE USER MAPPING ステートメントまたは ALTER USER MAPPING ステートメント、あるいはフェデレーテッド・サーバーのシステム・カタログからの情報を持ちます。このクラスは、CREATE USER MAPPING ステートメントまたは ALTER MAPPING ステートメントの処理時に情報が追加された場合に、ラッパーによりインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表で、UserInfo クラスのコンストラクターおよびメソッドについて説明します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 54. UserInfo クラスのコンストラクター

コンストラクター	説明
UserInfo	デフォルトの (空) ユーザー情報オブジェクトを構成します。

表 55. UserInfo クラスのメソッド

メソッド	説明
addOption	オプションをオプション・チェーンに追加します。
getAuthID	このユーザー・マッピングに対する許可 ID を戻します。
getPassword	このユーザー・マッピングに対するパスワードを戻します。
getServerName	このユーザー・マッピングに対するデータ・ソース・サーバー名を戻します。
isAuthIDValid	許可 ID が指定されているかどうかを検証します。
isServerNameValid	データ・ソース・サーバー名が指定されているかどうかを検証します。
setAuthID	このユーザー・マッピングに対する許可 ID を設定します。
setServerName	このユーザー・マッピングに対するデータ・ソース・サーバー名を設定します。

UserInfo コンストラクター

目的 デフォルトの (空) ユーザー情報オブジェクトを構成します。

構文

```
public UserInfo()
```

パラメーター

なし。

addOption メソッド

目的 オプションをオプション・チェーンに追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      int action)
               throws WrapperException
```

パラメーター

表 56. addOption メソッドのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。
action	オプションのアクション・フラグ。オプションの有効アクションは、CatalogOption クラスで指定されます。

戻り値 なし。

スロー オプションがチェーンに既存の場合、またはアクションが無効の場合、WrapperException オブジェクト。

getAuthID メソッド

目的 このユーザー・マッピングに対する許可 ID を戻します。

構文

```
public java.lang.String getAuthID()
```

パラメーター

なし。

戻り値 許可 ID。

スロー なし。

getPassword メソッド

目的 このユーザー・マッピングに対するパスワードを戻します。このユーザー・マッピングに対して指定された REMOTE_PASSWORD オプションの値 (存在する場合) を入手します。

構文

```
public java.lang.String getPassword()
```

パラメーター

なし。

戻り値 このユーザー・マッピングに対する REMOTE_PASSWORD オプションの値として指定されたパスワード。あるいは、オプションが見つからない場合は NULL。

スロー なし。

getServerName メソッド

目的 このユーザー・マッピングに対するデータ・ソース・サーバー名を戻します。

構文

```
public java.lang.String getServerName()
```

パラメーター
なし。

戻り値 データ・ソース・サーバー名。

スロー なし。

isAuthIDValid メソッド

目的 許可 ID が指定されているかどうかを検証します。

構文

```
public boolean isAuthIDValid()
```

パラメーター
なし。

戻り値 許可 ID が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isServerNameValid メソッド

目的 データ・ソース・サーバー名が指定されているかどうかを検証します。

構文

```
public boolean isServerNameValid()
```

パラメーター
なし。

戻り値 データ・ソース・サーバー名値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

setAuthID メソッド

目的 このユーザー・マッピングに対する許可 ID を設定します。

構文

```
public void setAuthID(java.lang.String authID)
```

パラメーター

表 57. setAuthID メソッドのパラメーター

名前	説明
authID	許可 ID。

戻り値 なし。

スロー なし。

setServerName メソッド

目的 このユーザー・マッピングに対するデータ・ソース・サーバー名を設定します。

構文

```
public void setServerName(java.lang.String serverName)
```

パラメーター

表 58. setServerName メソッドのパラメーター

名前	説明
serverName	データ・ソース・サーバー名。

戻り値 なし。

スロー なし。

継承されるメソッド

以下のメソッドは、CatalogInfo クラスから継承され、UserInfo クラスとともに使用できます。

- addOption
- dropOption
- getFirstOption
- getNextOption
- getOption

関連資料:

- 2 ページの『CatalogInfo クラス (Java)』
- 50 ページの『CatalogOption クラス (Java)』

WrapperInfo クラス (Java)

このセクションでは、WrapperInfo クラスについて説明し、そのコンストラクターおよびメソッドについて解説します。

概要

WrapperInfo クラスは、CREATE WRAPPER ステートメントおよび ALTER WRAPPER ステートメントからのラッパーのカタログ情報をカプセル化します。

public WrapperInfo クラスは CatalogInfo クラスを拡張します。

WrapperInfo クラスは、Java API のカタログ・クラスの 1 つです。

使用法 WrapperInfo クラスはフェデレーテッド・サーバーによりインスタンス化され、 CREATE WRAPPER ステートメントまたは ALTER WRAPPER ステートメント、あるいはフェデレーテッド・サーバーのシステム・カタログか

らの情報を持ちます。このクラスは CREATE WRAPPER ステートメントまたは ALTER WRAPPER ステートメントの処理時に情報が追加された場合に、ラッパーによりインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表で、WrapperInfo クラスのコンストラクターおよびメソッドについて説明します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 59. WrapperInfo クラスのコンストラクター

コンストラクター	説明
WrapperInfo	新規の (空) ラッパー情報オブジェクトを構成します。

表 60. WrapperInfo クラスのメソッド

メソッド	説明
addOption	オプションをオプション・チェーンに追加します。
getCorelib	コア・ライブラリー名を戻します。
getType	ラッパー・タイプを戻します。
getVersion	ラッパー・バージョンを戻します。
getWrapperName	ラッパー名を戻します。
isCorelibValid	コア・ライブラリーが指定されているかどうかを検証します。
isNameValid	名前が指定されているかどうかを検証します。
isTypeValid	タイプが指定されているかどうかを検証します。
isVersionValid	バージョンが指定されているかどうかを検証します。
setType	ラッパー・タイプを設定します。
setVersion	ラッパー・バージョンを設定します。
setWrapperName	ラッパー名を設定します。

WrapperInfo コンストラクター

目的 新規の (空) ラッパー情報オブジェクトを構成します。

構文

```
public WrapperInfo()
```

パラメーター

なし。

addOption メソッド

目的 オプションをオプション・チェーンに追加します。

構文

```
public void addOption(java.lang.String optionName,
                      java.lang.String optionValue,
                      int action)
                throws WrapperException
```

パラメーター

表 61. addOption メソッドのパラメーター

名前	説明
optionName	オプションの名前。
optionValue	オプションの値。
action	オプションのアクション・フラグ。オプションの有効アクションは、CatalogOption クラスで指定されます。

戻り値 なし。

スロー オプションがチェーンに既存の場合、またはアクションが無効の場合、WrapperException オブジェクト。

getCorelib メソッド

目的 コア・ライブラリー名を戻します。

構文

```
public java.lang.String getCorelib()
```

パラメーター

なし。

戻り値 コア・ライブラリーの名前。

スロー なし。

getType メソッド

目的 ラッパー・タイプを戻します。

構文

```
public char getType()
```

パラメーター

なし。

戻り値 ラッパー・タイプ。

スロー なし。

getVersion メソッド

目的 ラッパー・バージョンを戻します。

構文

```
public int getVersion()
```

パラメーター

なし。

戻り値 ラッパー・バージョン。

スロー なし。

getWrapperName メソッド

目的 ラッパー名を戻します。

構文

```
public java.lang.String getWrapperName()
```

パラメーター
なし。

戻り値 ラッパーの名前。

スロー なし。

isCorelibValid メソッド

目的 コア・ライブラリーが指定されているかどうを検証します。

構文

```
public boolean isCorelibValid()
```

パラメーター
なし。

戻り値 コア・ライブラリー値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isNameValid メソッド

目的 名前が指定されているかどうかを検証します。

構文

```
public boolean isNameValid()
```

パラメーター
なし。

戻り値 名前値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isTypeValid メソッド

目的 タイプが指定されているかどうかを検証します。

構文

```
public boolean isTypeValid()
```

パラメーター
なし。

戻り値 タイプ値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

isVersionValid メソッド

目的 バージョンが指定されているかどうかを検証します。

構文

```
public boolean isVersionValid()
```

パラメーター

なし。

戻り値 バージョン値が指定されている場合は、真の値。指定されていない場合には、値は偽です。

スロー なし。

setType メソッド

目的 ラッパー・タイプを設定します。

構文

```
public void setType(char type)
```

パラメーター

表 62. setType メソッドのパラメーター

名前	説明
type	ラッパー・タイプ。

戻り値 なし。

スロー なし。

setVersion メソッド

目的 ラッパー・バージョンを設定します。

構文

```
public void setVersion(int version)
```

パラメーター

表 63. setVersion メソッドのパラメーター

名前	説明
version	ラッパー・バージョン。

戻り値 なし。

スロー なし。

setWrapperName メソッド

目的 ラッパー名を設定します。

構文

```
public void setWrapperName(java.lang.String name)
```

パラメーター

表 64. setWrapperName メソッドのパラメーター

名前	説明
name	ラッパー名。

戻り値 なし。

スロー なし。

継承されるメソッド

以下のメソッドは、Catalog クラスから継承され、WrapperInfo クラスとともに使用できます。

- addOption
- dropOption
- getFirstOption
- getNextOption
- getOption

関連資料:

- 2 ページの『CatalogInfo クラス (Java)』

CatalogOption クラス (Java)

このセクションでは、CatalogOption クラスについて説明し、その定数およびメソッドの詳細について解説します。

CatalogOption クラスには、コンストラクターは含まれません。

フェデレーテッド・サーバーは、このクラスをインスタンス化し、フェデレーテッド・サーバーのシステム・カタログからこのオプションを記述します。

概要

CatalogOption クラスは、カタログ・オブジェクトのオプションのベース・クラスを表します。このクラスは、1 つ以上の対の名前値をカプセル化し、次のオプションおよび前のオプションにリンクします。

CatalogOption クラスは、Java API のカタログ・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

定数

以下の表には、CatalogOption クラスで使用できる、このオプションの有効アクション定数が記載されています。

表 65. CatalogOption クラスの定数

定数	定義	説明
ADD	public static final int ADD	カタログにオプションが追加されることを示します。
DROP	public static final int DROP	カタログからオプションがドロップされることを示します。
NONE	public static final int NONE	オプションに対するアクションがないことを示します。
SET	public static final int SET	オプションが新規値に設定されることを示します。新規値に設定されるオプションは、既にカタログ内に存在していなければなりません。

メソッド

次の表は、CatalogOption クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 66. CatalogOption クラスのメソッド

メソッド	説明
getAction	オプションのアクションを戻します。
getName	オプションの名前を戻します。
getNextOption	オプションが属するチェーン内の次のオプションを戻します。
getPrevOption	オプションが属するチェーン内の前のオプションを戻します。
getValue	オプション値を戻します。
isReserved	フェデレーテッド・サーバーがオプションを予約するかどうかを示します。

getAction メソッド

目的 オプションのアクションを戻します。

構文

```
public final int getAction()
```

パラメーター

なし。

戻り値 アクション。

スロー なし。

getName メソッド

目的 オプションの名前を戻します。

構文

```
public final java.lang.String getName()
```

パラメーター
なし。

戻り値 オプション名。

スロー なし。

getNextOption メソッド

目的 オプションが属するチェーン内の次のオプションを戻します。

構文

```
public final CatalogOption getNextOption()
```

パラメーター
なし。

戻り値 チェーン内の次のオプション。

スロー なし。

getPrevOption メソッド

目的 オプションが属するチェーン内の前のオプションを戻します。

構文

```
public final CatalogOption getPrevOption()
```

パラメーター
なし。

戻り値 チェーン内の前のオプション。

スロー なし。

getValue メソッド

目的 オプション値を戻します。

構文

```
public java.lang.String getValue()
```

パラメーター
なし。

戻り値 ストリングとしてのオプション値。

isReserved メソッド

目的 フェデレーテッド・サーバーがオプションを予約するかどうかを示します。

構文

```
public final boolean isReserved()
```

パラメーター
なし。

戻り値 フェデレーテッド・サーバーがオプションを予約する場合には、真の値。予約しない場合には、偽の値をこのメソッドは戻します。

スロー なし。

関連資料:

- 186 ページの『WrapperException クラス (Java)』

Java API のラッパー・クラス

以下の表で、Java API の各ラッパー・クラスについて説明します。

表 67. ラッパー・クラス

クラス名	説明
Wrapper	データ・ソースのグループ用のベース・クラスを表すクラス。このクラスは、ライブラリーの構成と初期設定、およびラッパーによりサポートされるデータ・ソース・サーバーへのアクセスを提供します。
FencedWrapper	fenced (非トラステッド) プロセス・スペース内の抽象ラッパーを表すクラス。直接このクラスを使用しないでください。 FencedWrapper クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。FencedGenericWrapper クラスを使用してください。
FencedGenericWrapper	fenced (非トラステッド) プロセス・スペース内のラッパーを表すクラス。
UnfencedWrapper	unfenced (トラステッド) プロセス・スペース内の抽象ラッパーを表すクラス。直接このクラスを使用しないでください。 UnfencedWrapper クラスを直接インスタンス化またはサブクラス化すると、ラッパーが正しく動作しなくなります。 UnfencedGenericWrapper クラスを使用してください。
UnfencedGenericWrapper	unfenced (トラステッド) プロセス・スペース内のラッパーを表すクラス。

関連資料:

- 57 ページの『FencedGenericWrapper クラス (Java)』
- 57 ページの『FencedWrapper クラス (Java)』
- 62 ページの『UnfencedGenericWrapper クラス (Java)』
- 59 ページの『UnfencedWrapper クラス (Java)』
- 53 ページの『Wrapper クラス (Java)』

Wrapper クラス (Java)

このセクションでは、Wrapper クラスについて説明し、そのメソッドの詳細について解説します。

Wrapper クラスには、コンストラクターは含まれません。

概要

Wrapper クラスは、データ・ソースのセットのベース・クラスを表します。このクラスは、ライブラリー初期化サービス、およびラッパーがサポートするデータ・ソース・サーバーへのアクセスを提供します。 Wrapper クラスは、以下の情報を保守します。

- ラッパー名。
- ラッパーのコア・ライブラリー名。戻された名前は、ラッパーをロードした固有ライブラリーの名前です。
- このラッパーに関連したすべての情報が含まれる WrapperInfo オブジェクト。この情報は、DDL ステートメント (CREATE WRAPPER または ALTER WRAPPER) の発行の結果として、フェデレーテッド・サーバーのシステム・カタログに保管されます。

FencedWrapper クラスおよび UnfencedWrapper クラスは Wrapper クラスのサブクラスです。

Wrapper クラスは Java API のラッパー・クラスです。

使用法 このクラスを直接使用せずに、FencedGenericWrapper クラスおよび UnfencedGenericWrapper クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表に、Wrapper クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 68. Wrapper クラスのメソッド

メソッド	説明
createServer	ラッパーに対する適切なサーバーのサブクラスをインスタンス化します。
getCorelib	ラッパーのライブラリー名を戻します。
getInfo	DDL ステートメントの実行結果として、フェデレーテッド・サーバーのシステム・カタログに保管されたラッパー情報を戻します。
getName	ラッパーの名前を戻します。
getType	ラッパー・タイプを戻します。
getVersion	ラッパーのバージョンを戻します。
initializeMyWrapper	カタログ情報オブジェクトからのラッパー・オブジェクト状態を初期化します。

createServer メソッド

目的 ラッパーに対する適切なサーバーのサブクラスをインスタンス化します。ラッパー・ライターは、このメソッドをインプリメントして、ラッパー固有の Server サブクラスのインスタンスを作成する必要があります。

構文

```
protected Server createServer(java.lang.String serverName)
                        throws java.lang.Exception
```

パラメーター

表 69. createServer メソッドのパラメーター

名前	説明
serverName	CREATE SERVER ステートメント上で指定されたデータ・ソース・サーバーの名前。

戻り値 新規に作成された Server インスタンス。

スロー 新規の Server インスタンスを作成できない場合には、Exception オブジェクト。

getCorelib メソッド

目的 ラッパーのライブラリー名を戻します。

構文

```
public final java.lang.String getCorelib()
```

パラメーター

なし。

戻り値 ラッパーのロード元のライブラリーの名前。

スロー なし。

getInfo メソッド

目的 DDL ステートメントの実行結果として、フェデレーテッド・サーバーのシステム・カタログに保管されたラッパー情報を戻します。

構文

```
public final WrapperInfo getInfo()
```

パラメーター

なし。

戻り値 WrapperInfo オブジェクト。

スロー なし。

getName メソッド

目的 ラッパーの名前を戻します。

構文

```
public final java.lang.String getName()
```

パラメーター

なし。

戻り値 CREATE WRAPPER ステートメント上で指定されたラッパーの名前。

スロー なし。

getType メソッド

目的 ラッパー・タイプを戻します。ラッパーが非リレーショナル照会プランニングをサポートする場合、ラッパー・タイプは N でなければなりません。getType メソッドは N を戻します。N は唯一の有効値です。

構文

```
public char getType()
```

パラメーター
なし。

戻り値 ラッパーのタイプ。

スロー なし。

getVersion メソッド

目的 現在実行中のバージョンを表す、ラッパーのバージョンを戻します。この値は、ラッパーを DB2 Universal Database で登録し、互換性が確証された時点のバージョンと比較できます。

構文

```
public int getVersion()
```

パラメーター
なし。

戻り値 ラッパーのバージョン。

スロー なし。

initializeMyWrapper メソッド

目的 カタログ情報オブジェクトからのラッパー・オブジェクト状態を初期化します。デフォルトのインプリメンテーションは何もしません。ラッパー固有のラッパー・オプションがサポートされている場合、ラッパー・ライターは、このメソッドを UnfencedGenericWrapper および FencedGenericWrapper のラッパー固有サブクラスにインプリメントすることができます。

構文

```
protected void initializeMyWrapper(WrapperInfo wrapperInfo)
                                throws java.lang.Exception
```

パラメーター

表 70. initializeMyWrapper メソッドのパラメーター

名前	説明
wrapperInfo	ラッパーのカタログ情報を含む WrapperInfo インスタンス。

戻り値 なし。

スロー 初期化プロセスが失敗した場合には、Exception オブジェクト。

関連資料:

- 57 ページの『FencedWrapper クラス (Java)』

- 64 ページの『Server クラス (Java)』
- 59 ページの『UnfencedWrapper クラス (Java)』
- 45 ページの『WrapperInfo クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedWrapper クラス (Java)

このセクションでは、FencedWrapper クラスについて説明します。

FencedWrapper クラスには、コンストラクターは含まれません。

概要

FencedWrapper クラスは、fenced (非トラステッド) プロセス・スペース上の抽象ラッパーを表します。

public FencedWrapper クラスは、Wrapper クラスを拡張します。

FencedWrapper クラスは、Java API のラッパー・クラスの 1 つです。

使用法 直接このクラスを使用しないでください。 FencedWrapper クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 FencedGenericWrapper クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

継承されるメソッド

以下のメソッドは、Wrapper クラスから継承され、FencedWrapper クラスと共に使用できます。

- createServer
- getCorelib
- getInfo
- getName
- getType
- getVersion
- initializeMyWrapper

関連資料:

- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedGenericWrapper クラス (Java)

このセクションでは、FencedGenericWrapper クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

FencedGenericWrapper クラスは、 fenced (非トラステッド) プロセス・スペース上のラッパーを表します。

public FencedGenericWrapper クラスは、FencedWrapper クラスを拡張します。

FencedGenericWrapper クラスは、Java API のラッパー・クラスの 1 つです。

使用法 ラッパーは、FencedGenericWrapper のサブクラスをインプリメントする必要があります。ラッパー固有 fenced 汎用ラッパー・サブクラスの名前は、CREATE WRAPPER ステートメントの FENCED_WRAPPER_CLASS オプション値として指定されます。またこの名前は、UnfencedWrapper.verifyMyRegisterWrapperInfo(WrapperInfo) 関数のラッパー検証プロセスの際に、UnfencedWrapper.setFencedWrapperClass(com.ibm.db2.wrapper. WrapperInfo, java.lang.String) 関数を呼び出して指定することもできます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表では、FencedGenericWrapper クラスのコンストラクターおよびメソッドが記載されています。表の後に、コンストラクターとメソッドの詳細が説明されています。

表 71. FencedGenericWrapper クラスのコンストラクター

コンストラクター	説明
FencedGenericWrapper	新規 FencedGenricWrapper オブジェクトを構成します。

表 72. FencedGenericWrapper クラスのメソッド

メソッド	説明
getType	ラッパー・タイプを戻します。

FencedWrapper コンストラクター

目的 新規 FencedGenricWrapper オブジェクトを構成します。

構文

```
protected FencedGenericWrapper()
```

パラメーター

なし。

スロー なし。

getType メソッド

目的 ラッパー・タイプを戻します。デフォルトのインプリメンテーションでは、N の値を戻します。 N が唯一の有効値です。 このメソッドは、Wrapper クラスの getType メソッドをオーバーライドします。

構文

```
public char getType()
```

パラメーター
なし。

戻り値 ラッパー・タイプ。デフォルトでは、この関数は値 N を戻します。

スロー なし。

継承されるメソッド

以下のメソッドは、Wrapper クラスから継承され、FencedGenericWrapper クラスと共に使用できます。

- createServer
- getCorelib
- getInfo
- getName
- getVersion
- initializeMyWrapper

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ラッパー・クラス』

関連資料:

- 57 ページの『FencedWrapper クラス (Java)』
- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedWrapper クラス (Java)

このセクションでは、UnfencedWrapper クラスについて説明し、そのメソッドの詳細について解説します。

UnfencedWrapper クラスには、コンストラクターは含まれていません。

概要

UnfencedWrapper クラスは、unfenced (トラステッド) プロセス・スペース内の抽象ラッパーを表します。

public UnfencedWrapper クラスは Wrapper クラスを拡張します。

UnfencedWrapper クラスは、Java API のラッパー・クラスの 1 つです。

使用法 直接 UnfencedWrapper クラスを使用しないでください。 UnfencedWrapper クラスを直接インスタンス化またはサブクラス化すると、ラッパーが正しく動作しなくなります。 UnfencedGenericWrapper クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表に、UnfencedWrapper クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 73. UnfencedWrapper クラスのメソッド

メソッド	説明
setFencedWrapperClass	ラッパー固有の FencedWrapper サブクラス名を WrapperInfo オブジェクトに追加します。
verifyMyAlterWrapperInfo	ALTER WRAPPER ステートメント上で指定された情報を検証します。
verifyMyRegisterWrapperInfo	CREATE WRAPPER ステートメント上で指定された情報を検証します。

setFencedWrapperClass メソッド

目的 ラッパー固有の FencedWrapper サブクラス名を WrapperInfo オブジェクトに追加します。

使用法 このメソッドは、ラッパーの fenced 部分用にロードする必要のあるクラスを示し、その情報をフェデレーテッド・サーバーに渡します。

構文

```
public final void setFencedWrapperClass(WrapperInfo wrapperInfo,
                                         java.lang.String className)
                                  throws WrapperException
```

パラメーター

表 74. setFencedWrapperClass メソッドのパラメーター

名前	説明
wrapperInfo	オプションが追加される WrapperInfo オブジェクト。
className	FencedWrapper サブクラス名。

戻り値 なし。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

verifyMyAlterWrapperInfo メソッド

目的 ALTER WRAPPER ステートメント上で指定された情報を検証します。

使用法 このメソッドは検査用としてのみ使用してください。 ALTER WRAPPER ステートメントから入手した新規情報で Wrapper オブジェクトの内部状態を更新しないでください。ステートメントが正常に実行された場合、フェデレーテッド・サーバーは Wrapper オブジェクトが破棄され、新規情報で再作成されるようにします。予約済みオプションのみを許可するデフォルトのインプリメンテーションが提供され、追加情報は戻されません。

ラッパー固有のラッパー・オプションがサポートされる場合は、このメソッドをインプリメントする必要があります。このメソッドは、ラッパーにより、ラッパー固有の unfenced ラッパー・サブクラスにインプリメントされる場合があります。

構文

```
protected WrapperInfo verifyMyAlterWrapperInfo(WrapperInfo wrapperInfo)
                                               throws java.lang.Exception
```

パラメーター

表 75. verifyMyAlterWrapperInfo メソッドのパラメーター

名前	説明
wrapperInfo	ALTER WRAPPER ステートメント内で提供される情報を含む WrapperInfo オブジェクト。

戻り値 ラッパーにより追加される情報が含まれる WrapperInfo オブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

verifyMyRegisterWrapperInfo メソッド

目的 CREATE WRAPPER ステートメント上で指定された情報を検証します。

使用法 予約済みオプションのみを許可するデフォルトのインプリメンテーションが提供され、追加情報は戻されません。ラッパー固有のラッパー・オプションがサポートされる場合は、このメソッドをインプリメントする必要があります。このメソッドは、ラッパーにより、ラッパー固有の unfenced ラッパー・サブクラスにインプリメントされる場合があります。

構文

```
protected WrapperInfo verifyMyRegisterWrapperInfo(WrapperInfo wrapperInfo)
                                                  throws java.lang.Exception
```

パラメーター

表 76. verifyMyRegisterWrapperInfo メソッドのパラメーター

名前	説明
wrapperInfo	CREATE WRAPPER ステートメント内で提供される情報を含む WrapperInfo オブジェクト。

戻り値 ラッパーにより追加される情報が含まれる WrapperInfo オブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

継承されるメソッド

以下のメソッドは、Wrapper クラスから継承され、UnfencedWrapper クラスとともに使用できます。

- createServer
- getCorelib
- getInfo

- getName
- getVersion
- initializeMyWrapper

関連資料:

- 45 ページの『WrapperInfo クラス (Java)』
- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedGenericWrapper クラス (Java)

このセクションでは、UnfencedGenericWrapper クラスについて説明し、そのコンストラクターおよびメソッドについて解説します。

概要

UnfencedGenericWrapper クラスは、unfenced (トラステッド) プロセス・スペース内のラッパーを表します。

public UnfencedGenericWrapper クラスは UnfencedWrapper クラスを拡張します。

UnfencedGenericWrapper クラスは、Java API のラッパー・クラスの 1 つです。

使用法 ラッパーは、UnfencedGenericWrapper のサブクラスをインプリメントする必要があります。ユーザー固有の unfenced generic wrapper サブクラスは、CREATE WRAPPER ステートメント内の UNFENCED_WRAPPER_CLASS オプションの値として指定されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表で、UnfencedGenericWrapper クラスのコンストラクターおよびメソッドについて説明します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 77. UnfencedGenericWrapper クラスのコンストラクター

コンストラクター	説明
UnfencedGenericWrapper	新規の UnfencedGenericWrapper オブジェクトを構成します。

表 78. UnfencedGenericWrapper クラスのメソッド

メソッド	説明
getType	ラッパー・タイプを戻します。

UnfencedGenericWrapper コンストラクター

目的 新規の UnfencedGenericWrapper オブジェクトを構成します。

構文

```
protected UnfencedGenericWrapper()
```

パラメーター
なし。

スロー なし。

getType メソッド

目的 ラッパー・タイプを戻します。デフォルトのインプリメンテーションでは、N の値を戻します。 N が唯一の有効値です。

構文

```
public char getType()
```

パラメーター
なし。

戻り値 ラッパー・タイプ。デフォルトでは、このメソッドは値 N を戻します。

スロー なし。

継承されるメソッド

以下のメソッドは、Wrapper クラスから継承され、UnfencedGenericWrapper クラスとともに使用できます。

- createServer
- getCorelib
- getInfo
- getName
- getVersion
- initializeMyWrapper

以下のメソッドは、UnfencedWrapper クラスから継承され、UnfencedGenericWrapper クラスとともに使用できます。

- setFencedWrapperClass
- verifyMyAlterWrapperInfo
- verifyMyRegisterWrapperInfo

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ラッパー・クラス』

関連資料:

- 59 ページの『UnfencedWrapper クラス (Java)』
- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

Java API のサーバー・クラス

以下の表で、Java API の各サーバー・クラスについて説明します。

表 79. サーバー・クラス

クラス名	説明
サーバー	すべてのデータ・ソース・サーバー機能の抽象ベース・クラスを表し、ラッパーによってサポートされる特定のデータ・ソースをマップするクラスです。
FencedServer	Server クラスのサブクラスであり、 fenced (非トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。直接このクラスを使用しないでください。 FencedServer クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 FencedGenericServer クラスを使用してください。
FencedGenericServer	Server クラスのサブクラスであり、 fenced (非トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。
UnfencedServer	Server クラスのサブクラスであり、 unfenced (トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。直接このクラスを使用しないでください。 UnfencedServer クラスを直接インスタンス化またはサブクラス化すると、ラッパーが正しく動作しなくなります。 UnfencedGenericServer クラスを使用してください。
UnfencedGenericServer	Server クラスのサブクラスであり、 unfenced (トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。

関連資料:

- 71 ページの『FencedGenericServer クラス (Java)』
- 69 ページの『FencedServer クラス (Java)』
- 64 ページの『Server クラス (Java)』
- 76 ページの『UnfencedGenericServer クラス (Java)』
- 73 ページの『UnfencedServer クラス (Java)』

Server クラス (Java)

このセクションでは、Server クラスについて説明し、そのメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

Server クラスは、すべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。 Server クラスは、ラッパーによってサポートされる特定のデータ・ソースをマップし、以下の情報を保持します。

- データ・ソース・サーバー名。
- 収容ラッパー・オブジェクトの参照。
- このデータ・ソース・サーバーの情報が含まれる ServerInfo オブジェクト。この情報は、DDL ステートメントの発行後にフェデレーテッド・サーバーのシステム・カタログに保管されます。

FencedServer クラスおよび UnfencedServer クラスは Server クラスのサブクラスです。

Server クラスは、Java API のサーバー・クラスの 1 つです。

使用法 このクラスを直接使用せずに、FencedGenericServer クラスおよび UnfencedGenericServer クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表に、Server クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 80. Server クラスのメソッド

メソッド	説明
createNickname	このデータ・ソース・サーバーの該当する Nickname サブクラスをインスタンス化します。
createRemoteUser	このデータ・ソース・サーバーに適した RemoteUser サブクラスをインスタンス化します。
findRemoteUser	フェデレーテッド・サーバーのシステム・カタログで指定されたローカル名を持つ、リモート・ユーザー・マッピングを検索します。
getInfo	DDL ステートメントを実行した後、フェデレーテッド・サーバーのシステム・カタログに保管されたデータ・ソース・サーバー情報を戻します。
getName	データ・ソース・サーバーの名前を戻します。
getType	データ・ソース・サーバー・タイプを戻します。
getVersion	データ・ソース・サーバーのバージョンを戻します。
getWrapper	このデータ・ソース・サーバーが属するラッパー・オブジェクトを戻します。

表 80. Server クラスのメソッド (続き)

メソッド	説明
initializeMyServer	有効なフェデレーテッド・サーバーのシステム・カタログ情報を使用して、データ・ソース・サーバーを初期化します。

createNickname メソッド

目的 このデータ・ソース・サーバーの該当する Nickname サブクラスをインスタンス化します。ラッパー・ライターは、このメソッドをラッパー固有のデータ・ソース・サーバーのサブクラスにインプリメントする必要があります。

構文

```
protected Nickname createNickname(java.lang.String schemaName,
                                  java.lang.String nickname)
                           throws java.lang.Exception
```

パラメーター

表 81. createNickname メソッドのパラメーター

名前	説明
schemaName	作成するニックネームのローカル・スキーマ名。
nickname	作成するニックネームのローカル名。

戻り値 新規に作成された Nickname インスタンス。

スロー 新規の Nickname インスタンスを作成できない場合には、Exception オブジェクト。

createRemoteUser メソッド

目的 このデータ・ソース・サーバーに適した RemoteUser サブクラスをインスタンス化します。

使用法 ラッパーは、このメソッドをラッパー固有のデータ・ソース・サーバーのサブクラスにインプリメントすることができます。 RemoteUser クラスのラッパー固有サブクラスがインプリメントされている場合、このメソッドをインプリメントする必要があります。

構文

```
protected RemoteUser createRemoteUser(java.lang.String userName)
                                throws java.lang.Exception
```

パラメーター

表 82. createRemoteUser メソッドのパラメーター

名前	説明
userName	CREATE USER MAPPING ステートメントで指定された、作成するリモート・ユーザー・マッピングの名前。

戻り値 新規に作成された RemoteUser インスタンス。

スロー 新規 RemoteUser インスタンスを作成できない場合には、Exception オブジェクト。

findRemoteUser メソッド

目的 フェデレーテッド・サーバーのシステム・カタログで指定されたローカル名を持つ、リモート・ユーザー・マッピングを検索します。指定された名前を持つリモート・ユーザー・マッピングが見つからない場合、NULL を戻します。

構文

```
public final RemoteUser findRemoteUser(java.lang.String userName)
```

パラメーター

表 83. findRemoteUser メソッドのパラメーター

名前	説明
userName	リモート・ユーザー・マッピングの名前。

戻り値 指定された名前を持つ RemoteUser インスタンス。あるいはリモート・ユーザー・マッピングが見つからない場合は NULL。

スロー なし。

getInfo メソッド

目的 DDL ステートメントを実行した後、フェデレーテッド・サーバーのシステム・カタログに保管されたデータ・ソース・サーバー情報を戻します。

構文

```
public final ServerInfo getInfo()
```

パラメーター
なし。

戻り値 データ・ソース・サーバー情報が含まれるインスタンス。

スロー なし。

getName メソッド

目的 データ・ソース・サーバーの名前を戻します。

構文

```
public final java.lang.String getName()
```

パラメーター
なし。

戻り値 CREATE SERVER ステートメントで指定されたデータ・ソース・サーバーの名前。

スロー なし。

getType メソッド

目的 データ・ソース・サーバー・タイプを戻します。

構文

```
public java.lang.String getType()
```

パラメーター
なし。

戻り値 データ・ソース・サーバーのタイプ。

スロー なし。

getVersion メソッド

目的 データ・ソース・サーバーのバージョンを戻します。

構文

```
public java.lang.String getVersion()
```

パラメーター
なし。

戻り値 データ・ソース・サーバーのバージョン。

スロー なし。

getWrapper メソッド

目的 このデータ・ソース・サーバーが属するラッパー・オブジェクトを戻します。

構文

```
public final Wrapper getWrapper()
```

パラメーター
なし。

戻り値 ラッパー・オブジェクト。

スロー なし。

initializeMyServer メソッド

目的 有効なフェデレーテッド・サーバーのシステム・カタログ情報を使用して、データ・ソース・サーバーを初期化します。

使用法 ラッパーは、このメソッドをラッパー固有のデータ・ソース・サーバーのサブクラスにインプリメントすることができます。ラッパー・ライターは、ラッパー固有のデータ・ソース・サーバー・オプションがサポートされている場合、このメソッドをインプリメントする必要があります。

構文

```
protected void initializeMyServer(ServerInfo serverInfo)
                                  throws java.lang.Exception
```

パラメーター

表 84. initializeMyServer メソッドのパラメーター

名前	説明
serverInfo	データ・ソース・サーバー情報が含まれる ServerInfo インスタンス。

戻り値 なし。

スロー 初期化プロセスが失敗した場合には、Exception オブジェクト。

関連資料:

- 69 ページの『FencedServer クラス (Java)』
- 91 ページの『Nickname クラス (Java)』
- 80 ページの『RemoteUser クラス (Java)』
- 33 ページの『ServerInfo クラス (Java)』
- 73 ページの『UnfencedServer クラス (Java)』
- 45 ページの『WrapperInfo クラス (Java)』
- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedServer クラス (Java)

このセクションでは、FencedServer クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

FencedServer クラスは Server クラスのサブクラスを表します。 fenced (非トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。このクラスは、リモート接続およびニックネームを作成します。

FencedServer クラスは、Java API のサーバー・クラスの 1 つです。

使用法 直接このクラスを使用しないでください。 FencedServer クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 FencedGenericServer クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、FencedServer クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 85. FencedServer クラスのメソッド

メソッド	説明
createRemoteConnection	指定されたユーザー・マッピングに対して指定された種類の新規接続を作成します。
findConnection	データ・ソース・サーバーの現行アクティブ接続を戻します。
getRemoteConnectionKind	このデータ・ソース・サーバーがサポートする接続タイプを戻します。

createRemoteConnection メソッド

目的 指定されたユーザー・マッピングに対して指定された種類の新規接続を作成します。ラッパー・ライターは、このメソッドをインプリメントします。

構文

```
protected RemoteConnection createRemoteConnection(FencedRemoteUser user,
                                                  int kind,
                                                  long id)
                                           throws java.lang.Exception
```

パラメーター

表 86. createRemoteConnection メソッドのパラメーター

名前	説明
user	ユーザーの認証に使用するユーザー・マッピング・オブジェクト。
kind	接続タイプ。
id	RemoteConnection オブジェクトを表す整数値。

戻り値 新規作成された接続インスタンス。

スロー RemoteConnection インスタンスを作成できない場合には、Exception オブジェクト。

findConnection メソッド

目的 データ・ソース・サーバーの現行アクティブ接続を戻します。

構文

```
public final RemoteConnection findConnection()
```

戻り値 このデータ・ソース・サーバーのアクティブ接続。アクティブ接続がない場合には NULL。

スロー なし。

getRemoteConnectionKind メソッド

目的 定数 RemoteConnection.NO_PHASE_KIND または RemoteConnection.ONE_PHASE_KIND によって示される、このデータ・ソース・サーバーがサポートする接続タイプを戻します。デフォルトのインプリメンテーションでは、RemoteConnection.NO_PHASE_KIND 定数を戻します。必要な場合にはサブクラスは、このメソッドを上書きできます。

構文

```
protected int getRemoteConnectionKind()
```

パラメーター
なし。

戻り値 接続タイプ。

スロー なし。

継承されるメソッド

以下のメソッドは、Server クラスから継承され、FencedServer クラスと共に使用できます。

- createNickname
- createRemoteUser
- findRemoteUser
- getInfo
- getName
- getType
- getVersion
- getWrapper
- initializeMyServer

関連資料:

- 71 ページの『FencedGenericServer クラス (Java)』
- 100 ページの『RemoteConnection クラス (Java)』
- 64 ページの『Server クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedGenericServer クラス (Java)

このセクションでは、FencedGenericServer クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

FencedGenericServer クラスは Server クラスのサブクラスであり、 fenced (非トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。このクラスは、リモート接続およびニックネームを作成します。

FencedGenericServer クラスは、Java API のサーバー・クラスの 1 つです。

使用法 ラッパーは、FencedGenericServer クラスのサブクラスをインプリメントする必要があります。このクラスは、FencedGenericWrapper のラッパー固有サブクラスの createServer メソッドにより、ラッパーでインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表では、FencedGenericServer クラスのコンストラクターおよびメソッドについて説明されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 87. *FencedGenericServer* クラスのコンストラクター

コンストラクター	説明
FencedGenericServer	指定された名前のラッパーの FencedGenericServer オブジェクトを構成します。

表 88. *FencedGenericServer* クラスのメソッド

メソッド	説明
createRemoteUser	このデータ・ソース・サーバーに適した RemoteUser サブクラスをインスタンス化します。

FencedGenericServer コンストラクター

目的 指定された名前のラッパーの FencedGenericServer オブジェクトを構成します。

構文

```
protected FencedGenericServer(java.lang.String name,
                              FencedGenericWrapper wrapper)
```

パラメーター

表 89. *FencedGenericServer* コンストラクターのパラメーター

名前	説明
name	データ・ソース・サーバーの名前。
wrapper	データ・ソース・サーバーを含むラッパー・オブジェクト。

createRemoteUser メソッド

目的 指定されたデータ・ソース・サーバーに適した RemoteUser サブクラスをインスタンス化します。デフォルトでは、createRemoteUser メソッドは、FencedGenericRemoteUser クラスのインスタンスを作成します。

使用法 ラッパーは、このメソッドをラッパー固有のデータ・ソース・サーバーのサブクラスにインプリメントすることができます。

FencedGenericRemoteUser クラスのラッパー固有サブクラスをインプリメントする場合には、ラッパー・ライターはこのメソッドをインプリメントする必要があります。

構文

```
protected RemoteUser createRemoteUser(java.lang.String userName)
                                throws java.lang.Exception
```

パラメーター

表 90. createRemoteUser メソッドのパラメーター

名前	説明
userName	CREATE USER MAPPING ステートメントで指定した、作成されるリモート・ユーザー・マッピングの名前。

戻り 新規 FencedGenericRemoteUser インスタンス。

スロー 新規 RemoteUser インスタンスを作成できない場合には、Exception オブジェクト。

継承されるメソッド

以下のメソッドは、FencedServer クラスから継承され、FencedGenericServer クラスと共に使用できます。

- createRemoteConnection
- findConnection
- getRemoteConnectionKind

以下のメソッドは、Server クラスから継承され、FencedGenericServer クラスと共に使用できます。

- createNickname
- findRemoteUser
- getInfo
- getName
- getType
- getVersion
- getWrapper
- initializeMyServer

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『サーバー・クラス』

関連資料:

- 57 ページの『FencedGenericWrapper クラス (Java)』
- 69 ページの『FencedServer クラス (Java)』
- 64 ページの『Server クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedServer クラス (Java)

このセクションでは、UnfencedServer クラスについて説明し、そのメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

UnfencedServer クラスは、Server クラスのサブクラスであり、unfenced (トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。このクラスは照会プランニング・アクティビティーを実行します。 UnfencedServer クラスは、CREATE NICKNAME および ALTER NICKNAME ステートメントで指定された情報の検証も担当します。

UnfencedServer クラスは、Java API のサーバー・クラスの 1 つです。

使用法 直接このクラスを使用しないでください。 UnfencedServer クラスを直接インスタンス化またはサブクラス化すると、ラッパーが正しく動作しなくなります。 UnfencedGenericServer クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表に、UnfencedServer クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 91. UnfencedServer クラスのメソッド

メソッド	説明
findNickname	指定されたスキーマ名およびニックネームを持つニックネームを検索します。
verifyMyAlterServerInfo	ALTER SERVER ステートメント内で指定されたデータ・ソース・サーバー情報を検証します。
verifyMyRegisterServerInfo	CREATE SERVER ステートメント内で指定されたデータ・ソース・サーバー情報を検証します。

findNickname メソッド

目的 指定されたスキーマ名およびニックネームを持つニックネームを検索します。

構文

```
public final Nickname findNickname(java.lang.String schema,
                                   java.lang.String name)
```

パラメーター

表 92. findNickname メソッドのパラメーター

名前	説明
schema	探索するニックネームのスキーマ名。
name	探索するニックネームの名前。

戻り値 指定された名前を持つニックネーム。あるいはニックネームが見つからない場合は NULL。

スロー なし。

verifyMyAlterServerInfo メソッド

目的 ALTER SERVER ステートメント内で指定されたデータ・ソース・サーバー情報を検証します。デフォルトでは、このメソッドは予約済みオプションのみを許可し、追加情報を戻しません。ラッパーは、ラッパー固有の unfenced server サブクラスに verifyMyAlterServerInfo メソッドをインプリメントすることができます。ラッパーは、ラッパー固有のデータ・ソース・サーバー・オプションがサポートされている場合、このメソッドをインプリメントする必要があります。

構文

```
protected ServerInfo verifyMyAlterServerInfo(ServerInfo serverInfo)
                                        throws java.lang.Exception
```

パラメーター

表 93. verifyMyAlterServerInfo メソッドのパラメーター

名前	説明
serverInfo	ALTER SERVER ステートメント内で提供される情報を含むオブジェクト。

戻り値 データ・ソース・サーバーにより追加される情報が含まれるオブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

verifyMyRegisterServerInfo メソッド

目的 CREATE SERVER ステートメント内で指定されたサーバー情報を検証します。デフォルトでは、このメソッドは予約済みオプションのみを許可し、追加情報を戻しません。ラッパーは、ラッパー固有の unfenced server サブクラスに verifyMyRegisterServerInfo メソッドをインプリメントすることができます。ラッパーは、ラッパー固有のデータ・ソース・サーバー・オプションがサポートされている場合、このメソッドをインプリメントする必要があります。

構文

```
protected ServerInfo verifyMyRegisterServerInfo(ServerInfo serverInfo)
                                            throws java.lang.Exception
```

パラメーター

表 94. verifyMyRegisterServerInfo メソッドのパラメーター

名前	説明
serverInfo	CREATE SERVER ステートメント内で提供される情報を含むオブジェクト。

戻り値 データ・ソース・サーバーにより追加される情報が含まれるオブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

継承されるメソッド

以下のメソッドは、Server クラスから継承され、UnfencedServer クラスとともに使用できます。

- createNickname
- createRemoteUser
- findRemoteUser
- getInfo
- getName
- getType
- getVersion
- getWrapper
- initializeMyServer

関連資料:

- 33 ページの『ServerInfo クラス (Java)』
- 64 ページの『Server クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedGenericServer クラス (Java)

このセクションでは、UnfencedGenericServer クラスについて説明し、そのコンストラクターおよびメソッドについて解説します。

概要

UnfencedGenericServer クラスは、Server クラスのサブクラスであり、unfenced (トラステッド) プロセス・スペースで操作されるすべてのデータ・ソース・サーバー機能の抽象ベース・クラスです。このクラスは照会プランニング・アクティビティーを実行します。 UnfencedGenericServer クラスは、CREATE NICKNAME および ALTER NICKNAME ステートメントで指定された情報の検証も担当します。

UnfencedGenericServer クラスは、Java API のサーバー・クラスの 1 つです。

使用法 ラッパーは、UnfencedGenericServer クラスのサブクラスをインプリメントする必要があります。このクラスは、UnfencedGenericWrapper クラスのラッパー固有サブクラスの createServer メソッドにより、ラッパーでインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表で、UnfencedGenericServer クラスのコンストラクターおよびメソッドについて説明します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 95. UnfencedGenericServer クラスのコンストラクター

コンストラクター	説明
UnfencedGenericServer	指定された名前を持つ指定されたラッパーの UnfencedGenericServer オブジェクトを構成します。

表 96. UnfencedGenericServer クラスのメソッド

メソッド	説明
createRemoteUser	このデータ・ソース・サーバーに適した RemoteUser サブクラスをインスタンス化します。
createReply	指定された Request に対する新規の空の Reply オブジェクトを作成します。
getSelectivity	述部リストの選択性を計算します。
planRequest	提案されたプランを分析し、どの程度のプラン (存在する場合) をリモート・データ・ソースにプッシュダウンするか決定します。

UnfencedGenericServer コンストラクター

目的 指定された名前を持つ指定されたラッパーの UnfencedGenericServer オブジェクトを構成します。

構文

```
protected UnfencedGenericServer(java.lang.String name,
                                UnfencedGenericWrapper wrapper)
```

パラメーター

表 97. UnfencedGenericServer コンストラクターのパラメーター

名前	説明
name	データ・ソース・サーバーの名前。
wrapper	データ・ソース・サーバーを含むラッパー・オブジェクト。

スロー なし。

createRemoteUser メソッド

目的 このデータ・ソース・サーバーに適した RemoteUser サブクラスをインスタンス化します。デフォルトで、このメソッドは UnfencedGenericRemoteUser クラスのインスタンスを作成します。

使用法 ラッパーは、このメソッドをラッパー固有のサーバーのサブクラスにインプリメントすることができます。 UnfencedGenericRemoteUser クラスのラッパー固有サブクラスをインプリメントする場合、このメソッドもインプリメントする必要があります。

構文

```
protected RemoteUser createRemoteUser(java.lang.String userName)
                                throws java.lang.Exception
```

パラメーター

表 98. createRemoteUser メソッドのパラメーター

名前	説明
userName	CREATE USER MAPPING ステートメントで指定された、作成するリモート・ユーザー・マッピングの名前。

戻り値 新規に作成された RemoteUser インスタンス。

スロー RemoteUser インスタンスが作成できない場合には、Exception オブジェクト。

createReply メソッド

目的 指定された Request に対する新規の空の Reply オブジェクトを作成します。 planRequest メソッドはこのメソッドを呼び出し、Reply オブジェクトを作成します。

構文

```
public final Reply createReply(Request request)
                        throws java.lang.Exception
```

パラメーター

表 99. createReply メソッドのパラメーター

名前	説明
request	Reply オブジェクトを作成する対象となる Request オブジェクト。

戻り値 新規に作成された Reply インスタンス。

スロー メソッドが失敗した場合には、Exception オブジェクト。

getSelectivity メソッド

目的 述部リストの選択性を計算します。

構文

```
public float getSelectivity(PredicateList predicateList)
                    throws java.lang.Exception
```

パラメーター

表 100. getSelectivity メソッドのパラメーター

名前	説明
PredicateList	PredicateList インスタンス。

戻り値 選択性の値。

スロー メソッドが失敗した場合には、Exception オブジェクト。

planRequest メソッド

目的 提案されたプランを分析し、どの程度のプラン (存在する場合) をリモー

ト・データ・ソースにプッシュダウンするか決定します。ラッパー・ライターは、このメソッドをインプリメントします。

構文

```
public Reply planRequest(Request request)
                  throws java.lang.Excepttion
```

パラメーター

表 101. planRequest メソッドのパラメーター

名前	説明
Request	計画済み照会が含まれる Request オブジェクト。

戻り値 Reply オブジェクト、または Reply のリストにある最初の Reply。それぞれの Reply は、ラッパーがリモート・データ・ソースにプッシュダウンできる照会フラグメントについて記述します。

スロー メソッドが失敗した場合には、Exception オブジェクト。

継承されるメソッド

以下のメソッドは、UnfencedServer クラスから継承され、UnfencedGenericServer クラスとともに使用できます。

- findNickname
- verifyMyAlterServerInfo
- verifyMyRegisterServerInfo

以下のメソッドは、Server クラスから継承され、UnfencedGenericServer クラスとともに使用できます。

- createNickname
- dropAllNicknames
- findRemoteUser
- getInfo
- getName
- getType
- getVersion
- getWrapper
- initializeMyServer

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『サーバー・クラス』

関連資料:

- 151 ページの『PredicateList クラス (Java)』
- 130 ページの『Reply クラス (Java)』
- 129 ページの『Request クラス (Java)』
- 80 ページの『RemoteUser クラス (Java)』

- 64 ページの『Server クラス (Java)』
- 73 ページの『UnfencedServer クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

Java API のユーザー・クラス

以下の表で、Java API の各ユーザー・クラスについて説明します。

表 102. ユーザー・クラス

クラス名	説明
RemoteUser	ラッパーが通信するデータ・ソース・サーバー上での使用許可を表すクラス。
FencedRemoteUser	fenced (非トラステッド) プロセス・スペース内のユーザー・マッピングを表す抽象ベース・クラス。直接このクラスを使用しないでください。 FencedRemoteUser クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 FencedGenericRemoteUser クラスを使用してください。
FencedGenericRemoteUser	fenced (非トラステッド) プロセス・スペース内のデータ・ソースのユーザー・マッピングを表すクラス。
UnfencedRemoteUser	unfenced (トラステッド) プロセス・スペース内のユーザー・マッピングを表す抽象ベース・クラス。直接このクラスを使用しないでください。 UnfencedRemoteUser クラスを直接インスタンス化またはサブクラス化すると、ラッパーが正しく動作しなくなります。 UnfencedGenericRemoteUser クラスを使用してください。
UnfencedGenericRemoteUser	unfenced (トラステッド) プロセス・スペース内のデータ・ソースのユーザー・マッピングを表し、 CREATE USER MAPPING および ALTER USER MAPPING ステートメント内で指定された情報を検証するためにインスタンス化されるクラス。

関連資料:

- 84 ページの『FencedGenericRemoteUser クラス (Java)』
- 83 ページの『FencedRemoteUser クラス (Java)』
- 80 ページの『RemoteUser クラス (Java)』
- 88 ページの『UnfencedGenericRemoteUser クラス (Java)』
- 86 ページの『UnfencedRemoteUser クラス (Java)』

RemoteUser クラス (Java)

このセクションでは、RemoteUser クラスについて説明し、そのメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RemoteUser クラスは、データ・ソース・サーバー上での使用許可を表します。このクラスの各インスタンスまたはそのサブクラスは、ラッパーが作業するデータ・ソース・サーバー上での使用許可を表します。 RemoteUser 基本クラスのインプリメンテーションは、以下の情報を保守します。

- ユーザーのローカル許可 ID
- データ・ソース・サーバー情報、および DDL ステートメントを実行した後、フェデレーテッド・サーバーのシステム・カタログに保管されたユーザー・ペアに関する情報が含まれる UserInfo オブジェクト
- ユーザー情報が含まれるデータ・ソース・サーバー・オブジェクトの参照

FencedRemoteUser クラスおよび UnfencedRemoteUser クラスは RemoteUser クラスのサブクラスです。

RemoteUser クラスは、Java API のユーザー・クラスの 1 つです。

使用法 直接このクラスを使用しないでください。 FencedGenericRemoteUser クラスおよび UnfencedGenericRemoteUser クラスをインスタンス化またはサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下の表では、RemoteUser クラスで使用できるオプションの有効な定数およびクラス・フィールドについて説明します。

表 103. RemoteUser クラスの定数およびクラス・フィールド

定数またはクラス・フィールド	定義	説明
REMOTE_AUTHID_ OPTION	public static final java.lang.String REMOTE_AUTHID_OPTION	リモート許可 ID オプションの名前を表す定数。
REMOTE_ PASSWORD_OPTION	public static final java.lang.String REMOTE_PASSWORD_OPTION	リモート許可パスワード・オプションの名前を表す定数。

メソッド

次の表に、RemoteUser クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 104. RemoteUser クラスのメソッド

メソッド	説明
getInfo	DDL ステートメントを実行した後、フェデレーテッド・サーバーのシステム・カタログに保管されたユーザー・マッピング情報を戻します。
getLocalName	ローカル・データベース上のユーザーの名前を戻します。

表 104. RemoteUser クラスのメソッド (続き)

メソッド	説明
getWrapper	このユーザー・マッピングが属するラッパー・インスタンスを戻します。
initializeMyUser	必要なユーザー・マッピングの初期化を実行します。

getInfo メソッド

目的 DDL ステートメントを実行した後、フェデレーテッド・サーバーのシステム・カタログに保管されたユーザー・マッピング情報を戻します。

構文

```
public final UserInfo getInfo()
```

パラメーター
なし。

戻り値 ユーザー・マッピング情報が含まれる UserInfo のインスタンス。

スロー なし。

getLocalName メソッド

目的 ローカル・データベース上のユーザーの名前を戻します。

構文

```
public final java.lang.String getLocalName()
```

パラメーター
なし。

戻り値 ローカル・ユーザー名。

スロー なし。

getWrapper メソッド

目的 このユーザーが属するラッパー・インスタンスを戻します。

構文

```
public final Wrapper getWrapper()
```

パラメーター
なし。

戻り値 ラッパー・オブジェクト。

スロー なし。

initializeMyUser メソッド

目的 必要なユーザー・マッピングの初期化を実行します。このメソッドは、ユーザー・マッピングの作成時あるいはそのオプションの変更時に呼び出されます。

このメソッドは、ラッパー・ライターにより、ユーザー・マッピングの初期化を実行する、ラッパー固有の RemoteUser サブクラスにインプリメントされる場合があります。

構文

```
protected void initializeMyUser(UserInfo userInfo)
                          throws java.lang.Exception
```

パラメーター

表 105. initializeMyUser メソッドのパラメーター

名前	説明
userInfo	ユーザー・マッピング情報を表す UserInfo のインスタンス。

戻り値 なし。

スロー 初期化プロセスが失敗した場合には、Exception オブジェクト。

関連資料:

- 64 ページの『Server クラス (Java)』
- 41 ページの『UserInfo クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedRemoteUser クラス (Java)

このセクションでは、FencedRemoteUser クラスについて説明します。

概要

FencedRemoteUser クラスは抽象ベース・クラスで、 fenced (非トラステッド) プロセス・スペース内のユーザー・マッピングを表します。

public FencedRemoteUser クラスは、RemoteUser クラスを拡張します。
FencedGenericRemoteUser クラスは、FencedRemoteUser クラスのサブクラスです。

FencedRemoteUser クラスは、Java API のユーザー・クラスの 1 つです。

使用法 直接 FencedRemoteUser クラスを使用しないでください。
FencedRemoteUser クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。
FencedGenericRemoteUser クラスをインスタンス化するか、サブクラス化してください。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下の表には、FencedRemoteUser クラスで使用できる、このオプションの有効定数とクラス・フィールドが記載されています。これらの定数およびクラス・フィールドは、RemoteUser クラスから継承されます。

表 106. FencedRemoteUser クラスの定数およびクラス・フィールド

定数またはクラス・フィールド	定義	説明
REMOTE_AUTHID_ OPTION	public static final java.lang.String REMOTE_AUTHID_OPTION	リモート許可 ID オプションの名前を表す定数。
REMOTE_ PASSWORD_OPTION	public static final java.lang.String REMOTE_PASSWORD_OPTION	リモート許可パスワード・オプションの名前を表す定数。

継承されるメソッド

以下のメソッドは、RemoteUser クラスから継承され、FencedRemoteUser クラスと共に使用できます。

- getInfo
- getLocalName
- getWrapper
- initializeMyUser

関連資料:

- 69 ページの『FencedServer クラス (Java)』
- 80 ページの『RemoteUser クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedGenericRemoteUser クラス (Java)

このセクションでは、FencedGenericRemoteUser クラスについて説明します。

概要

FencedGenericRemoteUser クラスは、 fenced (非トラステッド) プロセス・スペース内のデータ・ソースのユーザー・マッピングを表します。

FencedGenericRemoteUser クラスは、Java API のユーザー・クラスの 1 つです。

使用法 このクラスは、FencedGenericServer のラッパー固有サブクラスの createRemoteUser メソッドにより、ラッパーでインスタンス化されます。ラッパー固有のユーザー・マッピング・オプションがサポートされる場合、ラッパーは FencedGenericRemoteUser のサブクラスをインプリメントします。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下の表には、FencedGenericRemoteUser クラスで使用できる、このオプションの有効な定数とクラス・フィールドが記載されています。これらの定数およびクラス・フィールドは、RemoteUser クラスから継承されます。

表 107. FencedGenericRemoteUser クラスの定数およびクラス・フィールド

定数またはクラス・フィールド	定義	説明
REMOTE_AUTHID_ OPTION	public static final java.lang.String REMOTE_AUTHID_OPTION	リモート許可 ID オプションの名前を表す定数。
REMOTE_ PASSWORD_OPTION	public static final java.lang.String REMOTE_PASSWORD_OPTION	リモート許可パスワード・オプションの名前を表す定数。

コンストラクター

次の表は、FencedGenericRemoteUser クラスのコンストラクターについて説明しています。表の後に、コンストラクターの詳細が記載されています。

表 108. FencedGenericRemoteUser クラスのコンストラクター

コンストラクター	説明
FencedGenericRemoteUser	指定されたデータ・ソース・サーバーの指定された名前で、ユーザー・マッピングを構成します。

FencedGenericRemoteUser コンストラクター

目的 指定されたデータ・ソース・サーバーの指定された名前で、ユーザー・マッピングを構成します。

構文

```
public FencedGenericRemoteUser(java.lang.String localName,
                               FencedGenericServer server)
```

パラメーター

表 109. FencedGenericRemoteUser コンストラクターのパラメーター

名前	説明
localName	ユーザー・マッピングのローカル DB2 Information Integrator 名。
server	ユーザー・マッピングを含むサーバー・オブジェクト。

スロー なし。

継承されるメソッド

以下のメソッドは、RemoteUser クラスから継承され、FencedGenericRemoteUser クラスと共に使用できます。

- getInfo
- getLocalName
- getWrapper
- initializeMyUser

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ユーザー・クラス』

関連資料:

- 71 ページの『FencedGenericServer クラス (Java)』
- 83 ページの『FencedRemoteUser クラス (Java)』
- 80 ページの『RemoteUser クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedRemoteUser クラス (Java)

このセクションでは、UnfencedRemoteUser クラスについて説明し、そのメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

UnfencedRemoteUser クラスは、unfenced (トラステッド) プロセス・スペース内のユーザー・マッピングを表す抽象ベース・クラスです。このクラスはインスタンス化され、CREATE USER MAPPING および ALTER USER MAPPING ステートメント内で指定された情報を検証します。

public UnfencedRemoteUser クラスは RemoteUser クラスを拡張します。UnfencedGenericRemoteUser クラスは UnfencedRemoteUser クラスのサブクラスです。

UnfencedRemoteUser クラスは、Java API のユーザー・クラスの 1 つです。

使用法 直接 UnfencedRemoteUser クラスを使用しないでください。UnfencedRemoteUser クラスを直接インスタンス化またはサブクラス化すると、ラッパーが正しく動作しなくなります。 UnfencedGenericRemoteUser クラスをインスタンス化またはサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下の表では、UnfencedRemoteUser クラスで使用できるオプションの有効な定数およびクラス・フィールドについて説明します。これらの定数およびクラス・フィールドは、RemoteUser クラスから継承されます。

表 110. UnfencedRemoteUser クラスの定数およびクラス・フィールド

定数またはクラス・フィールド	定義	説明
REMOTE_AUTHID_ OPTION	public static final java.lang.String REMOTE_AUTHID_OPTION	リモート許可 ID オプションの名前を表す定数。
REMOTE_ PASSWORD_OPTION	public static final java.lang.String REMOTE_PASSWORD_OPTION	リモート許可パスワード・オプションの名前を表す定数。

メソッド

次の表に、UnfencedRemoteUser クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 111. UnfencedRemoteUser クラスのメソッド

メソッド	説明
verifyMyAlterUserInfo	ALTER USER MAPPING ステートメント内で指定されたユーザー・マッピング情報を検証します。
verifyMyRegisterUserInfo	CREATE USER MAPPING ステートメント内で指定されたユーザー・マッピング情報を検証します。

verifyMyAlterUserInfo メソッド

目的 ALTER USER MAPPING ステートメント内で指定されたユーザー・マッピング情報を検証します。デフォルトでは、このメソッドは予約済みオプションのみ許容し、他のオプションは戻しません。ラッパー固有のユーザー・マッピング・オプションがサポートされる場合は、このメソッドをインプリメントする必要があります。

構文

```
protected UserInfo verifyMyAlterUserInfo(UserInfo userInfo)
                                         throws java.lang.Exception
```

パラメーター

表 112. verifyMyAlterUserInfo メソッドのパラメーター

名前	説明
userInfo	ALTER USER MAPPING ステートメント内で提供される情報を含む UserInfo オブジェクト。

戻り値 ラッパーにより追加される情報が含まれる UserInfo オブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

verifyMyRegisterUserInfo メソッド

目的 CREATE USER MAPPING ステートメント内で指定されたユーザー・マッピング情報を検証します。デフォルトでは、このメソッドは予約済みオプションのみ許容し、他のオプションは戻しません。ラッパー固有のユーザー・マッピング・オプションがサポートされる場合は、このメソッドをインプリメントする必要があります。

構文

```
protected UserInfo verifyMyRegisterUserInfo(UserInfo userInfo)
                                            throws java.lang.Exception
```

パラメーター

表 113. verifyMyRegisterUserInfo メソッドのパラメーター

名前	説明
userInfo	CREATE USER MAPPING ステートメント内で提供される情報を含む UserInfo オブジェクト。

戻り値 ラッパーにより追加される情報が含まれる UserInfo オブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

継承されるメソッド

以下のメソッドは、RemoteUser クラスから継承され、UnfencedRemoteUser クラスとともに使用できます。

- getInfo
- getLocalName
- getWrapper
- initializeMyUser

関連資料:

- 80 ページの『RemoteUser クラス (Java)』
- 41 ページの『UserInfo クラス (Java)』
- 45 ページの『WrapperInfo クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedGenericRemoteUser クラス (Java)

このセクションでは、UnfencedGenericRemoteUser クラスについて説明します。

概要

UnfencedGenericRemoteUser クラスは、 unfenced (トラステッド) プロセス・スペース内のデータ・ソースのユーザー・マッピングを表します。

UnfencedGenericRemoteUser クラスは、Java API のユーザー・クラスの 1 つです。

使用法 このクラスは、UnfencedGenericServer のラッパー固有サブクラスの createRemoteUser メソッドにより、ラッパーでインスタンス化されます。CREATE USER MAPPING ステートメントまたは ALTER USER MAPPING ステートメントのラッパー固有のユーザー・マッピング・オプションを使用する場合、ラッパーは UnfencedGenericRemoteUser のサブクラスをインプリメントする必要があります。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下の表では、UnfencedGenericRemoteUser クラスで使用できるオプションの有効な定数およびクラス・フィールドについて説明します。これらの定数

およびクラス・フィールドは、RemoteUser クラスから継承されます。

表 114. UnfencedGenericRemoteUser クラスの定数およびクラス・フィールド

定数またはクラス・フィールド	定義	説明
REMOTE_AUTHID_ OPTION	public static final java.lang.String REMOTE_AUTHID_OPTION	リモート許可 ID オプションの名前を表す定数。
REMOTE_ PASSWORD_OPTION	public static final java.lang.String REMOTE_PASSWORD_OPTION	リモート許可パスワード・オプションの名前を表す定数。

コンストラクター

次の表は、UnfencedGenericRemoteUser クラスのコンストラクターについて説明しています。表の後に、コンストラクターの詳細が記載されています。

表 115. UnfencedGenericRemoteUser クラスのコンストラクター

コンストラクター	説明
UnfencedGenericRemoteUser	指定されたデータ・ソース・サーバーの指定された名前で、ユーザー・マッピングを構成します。

UnfencedGenericRemoteUser コンストラクター

目的 指定されたデータ・ソース・サーバーの指定された名前で、ユーザー・マッピングを構成します。

構文

```
public UnfencedGenericRemoteUser(java.lang.String localName,
                                 UnfencedGenericServer server)
```

パラメーター

表 116. UnfencedGenericRemoteUser コンストラクターのパラメーター

名前	説明
localName	ユーザー・マッピングのローカル DB2 Information Integrator 名。
server	ユーザー・マッピングを含むサーバー・オブジェクト。

スロー なし。

継承されるメソッド

以下のメソッドは、UnfencedRemoteUser クラスから継承され、UnfencedGenericRemoteUser クラスとともに使用できます。

- verifyMyAlterUserInfo
- verifyMyRegisterUserInfo

以下のメソッドは、RemoteUser クラスから継承され、UnfencedGenericRemoteUser クラスとともに使用できます。

- getInfo
- getLocalName
- getWrapper
- initializeMyUser

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ユーザー・クラス』

関連資料:

- 80 ページの『RemoteUser クラス (Java)』
- 76 ページの『UnfencedGenericServer クラス (Java)』
- 86 ページの『UnfencedRemoteUser クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

Java API のニックネーム・クラス

以下の表で、Java API の各ニックネーム・クラスについて説明します。

表 117. サーバー・クラス

クラス名	説明
Nickname	ラッパーが処理する、データ・ソース・サーバーによって管理されるデータの集合を表すクラス。
FencedNickname	fenced (非トラステッド) プロセス・スペース内のニックネームを表す抽象ベース・クラス。直接このクラスを使用しないでください。 FencedNickname クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 FencedGenericNickname クラスをサブクラス化してください。
FencedGenericNickname	fenced (非トラステッド) プロセス・スペース内のニックネームを表し、 CREATE NICKNAME ステートメントからの情報の検証を行うクラス。
UnfencedNickname	unfenced (トラステッド) プロセス・スペース内のニックネームを表す抽象ベース・クラス。直接このクラスを使用しないでください。 UnfencedNickname クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 UnfencedGenericNickname クラスをサブクラス化してください。
UnfencedGenericNickname	unfenced (トラステッド) プロセス・スペース内のニックネームを表すクラス。このクラスは、CREATE NICKNAME ステートメントおよび ALTER NICKNAME ステートメントからの情報を検証します。

関連資料:

- 95 ページの『FencedGenericNickname クラス (Java)』

- 94 ページの『FencedNickname クラス (Java)』
- 91 ページの『Nickname クラス (Java)』
- 98 ページの『UnfencedGenericNickname クラス (Java)』
- 97 ページの『UnfencedNickname クラス (Java)』

Nickname クラス (Java)

このセクションでは、Nickname クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

Nickname クラスは、データ・ソース・サーバーが管理するデータの集合をモデル化します。このクラスまたはそのサブクラスの各インスタンスは、ラッパーが処理する、データ・ソース・サーバーによって管理されるデータの集合を表します。Nickname ベース・クラスのインプリメンテーションによって、以下の情報が保守されます。

- このニックネームに対して定義されている、リモート・データ・セットのローカル名。
- このニックネームに対して定義されている、リモート・データ・セットのローカル・スキーマ。
- NicknameInfo オブジェクト。これには、CREATE NICKNAME または ALTER NICKNAME ステートメントの実行後に、フェデレーテッド・サーバーのシステム・カタログに保管されたニックネームに関する情報が含まれます。NicknameInfo オブジェクトは、各列のローカル名および追加情報が含まれる ColumnInfo オブジェクトを参照します。こうした情報は、DDL ステートメントの実行結果として、フェデレーテッド・サーバーのシステム・カタログに保管されています。
- このニックネームが含まれるデータ・ソース・サーバー・オブジェクトへの参照。

FencedNickname クラスおよび UnfencedNickname クラスは、Nickname クラスのサブクラスです。

Nickname クラスは、Java API のユーザー・クラスの 1 つです。

使用法 直接 Nickname クラスを使用するのではなく、サブクラスの FencedGenericNickname クラスおよび UnfencedGenericNickname クラスを使用してください。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、Nickname クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 118. Nickname クラスのメソッド

メソッド	説明
getInfo	フェデレーテッド・サーバーのシステム・カタログに、 DDL ステートメントの実行結果として保管されているニックネーム情報を戻します。
getLocalName	このニックネームのローカル名を戻します。
getLocalSchema	このニックネームのローカル・スキーマを戻します。
getServer	このニックネームが含まれるデータ・ソース・サーバーを戻します。
getWrapper	このニックネームが属するラッパー・インスタンスを戻します。
initializeMyNickname	必要なニックネーム初期化を実行します。
verifyMyRegisterNicknameInfo	CREATE NICKNAME ステートメントで指定されたニックネーム情報を検証します。

getInfo メソッド

目的 フェデレーテッド・サーバーのシステム・カタログに、 DDL ステートメントの実行結果として保管されているニックネーム情報を戻します。

構文

```
public final NicknameInfo getInfo()
```

パラメーター

なし。

戻り値 ニックネーム情報を含む NicknameInfo インスタンス。

スロー なし。

getLocalName メソッド

目的 このニックネームのローカル名を戻します。

構文

```
public final java.lang.String getLocalName()
```

パラメーター

なし。

戻り値 ニックネームのローカル名。

スロー なし。

getLocalSchema メソッド

目的 このニックネームのローカル・スキーマを戻します。

構文

```
public final java.lang.String getLocalSchema()
```

パラメーター

なし。

戻り値 ニックネームのローカル・スキーマ。

スロー なし。

getServer メソッド

目的 このニックネームが含まれるデータ・ソース・サーバーを戻します。

構文

```
public final Server getServer()
```

パラメーター
なし。

戻り値 データ・ソース・サーバー。

スロー なし。

getWrapper メソッド

目的 このニックネームが属するラッパー・インスタンスを戻します。

構文

```
public final Wrapper getWrapper()
```

パラメーター
なし。

戻り値 ラッパー・オブジェクト。

スロー なし。

initializeMyNickname メソッド

目的 必要なニックネーム初期化を実行します。このメソッドは、ニックネームが作成されると、またはオプションが変更されると呼び出されます。 ラッパー・ライターは、ラッパー固有ニックネームでこのメソッドをインプリメントし、ニックネーム固有初期化プロセスを呼び出せます。

構文

```
protected void initializeMyNickname(NicknameInfo nicknameInfo)
                                    throws java.lang.Exception
```

パラメーター

表 119. initializeMyNickname メソッドのパラメーター

名前	説明
nicknameInfo	ニックネーム・カタログ情報。

戻り値 なし。

スロー 初期化プロセスが失敗した場合には、Exception オブジェクト。

verifyMyRegisterNicknameInfo メソッド

目的 CREATE NICKNAME ステートメントで指定されたニックネーム情報を検証します。デフォルトでは、このメソッドは予約済みオプションのみ許容し、他のオプションは戻しません。

使用法 ラッパーは、ラッパー固有ニックネーム・サブクラスで

verifyMyRegisterNicknameInfo メソッドをインプリメントできます。 unfenced クラスであれ、fenced クラスであれ、ラッパー固有ニックネームまたは列オプションがサポートされる場合には、このメソッドをインプリメントする必要があります。 UnfencedGenericNickname クラスの verifyMyRegisterNicknameInfo メソッドはトラステッド・プロセス・スペースの一部であるため、このメソッドはリモート・データ・ソースと対話できません。 ニックネーム情報を検証するためにリモート・データ・ソースとの対話が必要な場合には、 FencedGenericNickname クラスの verifyMyRegisterNicknameInfo メソッドをインプリメントしなければなりません。

構文

```
protected NicknameInfo verifyMyRegisterNicknameInfo(NicknameInfo nicknameInfo)
                                                    throws java.lang.Exception
```

パラメーター

表 120. verifyMyRegisterNicknameInfo メソッドのパラメーター

名前	説明
nicknameInfo	CREATE NICKNAME ステートメントからの情報。

戻り値 ニックネーム・オブジェクトより追加される情報。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

関連資料:

- 22 ページの『NicknameInfo クラス (Java)』
- 64 ページの『Server クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedNickname クラス (Java)

このセクションでは、FencedNickname クラスについて説明します。

概要

FencedNickname クラスは抽象ベース・クラスで、 fenced (非トラステッド) プロセス・スペース内のニックネームを表します。このクラスは、CREATE NICKNAME ステートメントからの情報を検証します。

public FencedGenericServer クラスは、FencedServer クラスを拡張します。 FencedGenericNickname クラスは、FencedNickname クラスのサブクラスです。

FencedNickname クラスは、Java API のサーバー・クラスの 1 つです。

使用法 直接 FencedNickname クラスを使用しないでください。 FencedNickname クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。 FencedGenericNickname クラスをサブクラス化してください。

パッケージ

com.ibm.db2.wrapper

継承されるメソッド

以下のメソッドは、Nickname クラスから継承され、FencedNickname クラスと共に使用できます。

- getInfo
- getLocalName
- getLocalSchema
- getServer
- getWrapper
- initializeMyNickname
- verifyMyRegisterNicknameInfo

関連資料:

- 22 ページの『NicknameInfo クラス (Java)』
- 91 ページの『Nickname クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

FencedGenericNickname クラス (Java)

このセクションでは、FencedGenericNickname クラスについて説明します。

概要

FencedGenericNickname クラスは、 fenced (非トラステッド) プロセス・スペース内のデータ・ソースのニックネームを表します。このクラスは、CREATE NICKNAME ステートメントからの情報を検証します。

FencedGenericNickname クラスは、FencedNickname クラスを拡張します。

FencedGenericNickname クラスは、Java API のユーザー・クラスの 1 つです。

使用法 ラッパーは、FencedGenericNickname クラスのサブクラスでなければなりません。 FencedGenericNickname クラスは、 FencedGenericServer のラッパー固有サブクラスの createNickname メソッドによって、ラッパーでインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクター

次の表は、FencedGenericNickname クラスのコンストラクターについて説明しています。表の後に、コンストラクターの詳細が記載されています。

表 121. FencedGenericNickname クラスのコンストラクター

コンストラクター	説明
FencedGenericNickname	指定したデータ・ソース・サーバーの指定したスキーマおよび名前で、ニックネームを構成します。

FencedGenericNickname コンストラクター

目的 指定したデータ・ソース・サーバーの指定したスキーマおよび名前で、ニックネームを構成します。

構文

```
protected FencedGenericNickname(java.lang.String schema,
                                java.lang.String name,
                                FencedGenericServer server)
```

パラメーター

表 122. FencedGenericNickname コンストラクターのパラメーター

名前	説明
schema	このニックネームに対して定義されている、リモート・データ・セットのローカル DB2 Information Integrator スキーマ。
name	このニックネームに対して定義されている、リモート・データ・セットのローカル DB2 Information Integrator 名。
server	ニックネームを含むサーバー・オブジェクト。

スロー なし。

継承されるメソッド

以下のメソッドは、Nickname クラスから継承され、FencedGenericNickname クラスと共に使用できます。

- getInfo
- getLocalName
- getLocalSchema
- getServer
- getWrapper
- initializeMyNickname
- verifyMyRegisterNicknameInfo

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ニックネーム・クラス』

関連資料:

- 71 ページの『FencedGenericServer クラス (Java)』
- 94 ページの『FencedNickname クラス (Java)』

- 91 ページの『Nickname クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedNickname クラス (Java)

このセクションでは、UnfencedNickname クラスについて説明し、そのコンストラクターおよびメソッドについて解説します。

このクラスには、コンストラクターは含まれません。

概要

UnfencedNickname クラスは、unfenced (トラステッド) プロセス・スペース内のニックネームを表す抽象ベース・クラスです。このクラスは、CREATE NICKNAME ステートメントおよび ALTER NICKNAME ステートメントからの情報を検証します。

UnfencedNickname クラスは Nickname クラスを拡張します。

UnfencedNickname クラスは、Java API のユーザー・クラスの 1 つです。

使用法 直接 UnfencedNickname クラスを使用しないでください。
UnfencedNickname クラスを直接インスタンス化したりサブクラス化したりすると、ラッパーの動作が正常ではなくなります。
UnfencedGenericNickname クラスをサブクラス化してください。

パッケージ
com.ibm.db2.wrapper

メソッド

次の表に、UnfencedNickname クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 123. UnfencedNickname クラスのメソッド

メソッド	説明
verifyMyAlterNicknameInfo	ALTER NICKNAME ステートメント内で指定されたニックネーム情報を検証します。

verifyMyAlterNicknameInfo メソッド

目的 ALTER NICKNAME ステートメント内で指定されたニックネーム情報を検証します。デフォルトでは、このメソッドは予約済みオプションのみ許容し、他のオプションは戻しません。

使用法 ラッパーは、ラッパー固有の nickname サブクラスに verifyMyAlterNicknameInfo メソッドをインプリメントすることができます。ラッパー固有のニックネーム・オプションまたは列オプションがサポートされる場合は、verifyMyAlterNicknameInfo メソッドをインプリメントする必要があります。 verifyMyAlterNicknameInfo メソッドはトラステッド・プロセス・スペースの一部であるため、このメソッドはリモート・データ・ソースと対話できません。

構文

```
protected NicknameInfo verifyMyAlterNicknameInfo(NicknameInfo nicknameInfo)
                                                 throws java.lang.Exception
```

パラメーター

表 124. verifyMyAlterNicknameInfo メソッドのパラメーター

名前	説明
nicknameInfo	ALTER NICKNAME ステートメント内で提供される情報を含むオブジェクト。

戻り値 ニックネームにより追加される情報が含まれるオブジェクト。

スロー 検証プロセスが失敗した場合には、Exception オブジェクト。

継承されるメソッド

以下のメソッドは、Nickname クラスから継承され、UnfencedNickname クラスとともに使用できます。

- getInfo
- getLocalName
- getLocalSchema
- getServer
- getWrapper
- initializeMyNickname
- verifyMyRegisterNicknameInfo

関連資料:

- 22 ページの『NicknameInfo クラス (Java)』
- 91 ページの『Nickname クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

UnfencedGenericNickname クラス (Java)

このセクションでは、UnfencedGenericNickname クラスについて説明します。

概要

UnfencedGenericNickname クラスは、 unfenced (トラステッド) プロセス・スペース内のデータ・ソースのニックネームを表します。このクラスは、CREATE NICKNAME ステートメントおよび ALTER NICKNAME ステートメントからの情報を検証します。

UnfencedGenericNickname クラスは UnfencedNickname クラスを拡張します。

UnfencedGenericNickname クラスは、Java API のユーザー・クラスの 1 つです。

使用法 UnfencedGenericNickname クラスは、ラッパーによりサブクラス化され、UnfencedGenericServer のラッパー固有サブクラスの createNickname メソッドにインスタンス化されます。

パッケージ

com.ibm.db2.wrapper

コンストラクター

次の表に、UnfencedGenericNickname クラスのコンストラクターを示します。表の後に、コンストラクターの詳細が記載されています。

表 125. UnfencedGenericNickname クラスのコンストラクター

コンストラクター	説明
UnfencedGenericNickname	指定したデータ・ソース・サーバーの指定したスキーマおよび名前で、ニックネームを構成します。

UnfencedGenericNickname コンストラクター

目的 指定したデータ・ソース・サーバーの指定したスキーマおよび名前で、ニックネームを構成します。

構文

```
protected UnfencedGenericNickname(java.lang.String schema,
                                  java.lang.String name,
                                  UnfencedGenericServer server)
```

パラメーター

表 126. UnfencedGenericNickname コンストラクターのパラメーター

名前	説明
schema	このニックネームに対して定義されている、リモート・データ・セットのローカル DB2 Information Integrator スキーマ。
name	このニックネームに対して定義されている、リモート・データ・セットのローカル DB2 Information Integrator 名。
server	ニックネームを含むサーバー・オブジェクト。

スロー なし。

継承されるメソッド

verifyMyAlterNicknameInfo メソッドは、UnfencedNickname クラスから継承され、UnfencedGenericNickname クラスとともに使用することもできます。

以下のメソッドは、Nickname クラスから継承され、UnfencedGenericNickname クラスとともに使用できます。

- getInfo
- getLocalName
- getLocalSchema
- getServer
- getWrapper

- initializeMyNickname
- verifyMyRegisterNicknameInfo

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ニックネーム・クラス』

関連資料:

- 91 ページの『Nickname クラス (Java)』
- 97 ページの『UnfencedNickname クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

RemoteConnection クラス (Java)

このセクションでは、RemoteConnection クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

RemoteConnection クラスは、データ・ソース・サーバーとの接続 (セッション) を表します。データ・ソース・サーバー (データベース) 上の操作は、接続を介して処理されます。接続は、非トラステッドかつ fenced データ・ソース・サーバーの場合のみ作成されます。 RemoteConnection ベース・クラスのインプリメンテーションによって、以下の情報が保守されます。

- ご使用のデータ・ソースへの接続状況 (接続または切断)。
- 適切な FencedServer オブジェクトへの参照。
- 適切な FencedRemoteUser オブジェクトへの参照。
- 接続に使用するコード・ページ。
- 接続タイプ。接続タイプの詳細については、 101 ページの表 127 を参照してください。

RemoteConnection クラスは、Java API の接続クラスです。

使用法 FencedServer.createRemoteConnection メソッドを適切な fenced サーバー・サブクラスのインスタンス上に呼び出すと、 RemoteConnection サブクラスのインスタンスを作成できます。フェデレーテッド・サーバーは、関連するデータ・ソース・サーバーで最初のリモート操作を処理する前に、このメソッドを呼び出します。フェデレーテッド・サーバーは、データ・ソース・サーバーを使用しないで、事前定義された数のトランザクションをアプリケーションで処理した後に、 RemoteConnection インスタンスを破棄します。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下の表には、RemoteConnection クラスで使用できる、このオプションの有効な定数とクラス・フィールドが記載されています。

表 127. RemoteConnection クラスの定数およびクラス・フィールド

定数またはクラス・フィールド	定義	説明
NO_PHASE_KIND	public static final int NO_PHASE_KIND	トランザクションがサポートされないことを示します。
ONE_PHASE_KIND	public static final int ONE_PHASE_KIND	1 フェーズ・コミットのトランザクションがサポートされることを示します。

コンストラクターおよびメソッド

以下の表では、RemoteConnection クラスのコンストラクターおよびメソッドが記載されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 128. RemoteConnection クラスのコンストラクター

コンストラクター	説明
RemoteConnection	指定されたユーザー許可とトランザクションを使用する、指定のデータ・ソース・サーバーの接続を構成します。

表 129. RemoteConnection クラスのメソッド

メソッド	説明
commit	トランザクションが正常に完了し、リモート・データ・ソースがトランザクションをコミットすることを示します。
connect	データ・ソース・サーバーに接続するためのコードを呼び出します。
createRemotePassthru	RemotePassthru オブジェクトを作成して、パススルー・ステートメントを実行します。
createRemoteQuery	SQL ステートメントを実行して、RemoteQuery オブジェクトを作成します。
disconnect	データ・ソース・サーバーから切断するためのコードを呼び出します。
getCodepage	接続用のコード・ページを戻します。
getKind	接続タイプを戻します。
getServer	この接続が含まれるデータ・ソース・サーバー・オブジェクトを戻します。
getUser	接続を認証するためのユーザー・マッピングを戻します。
getWrapper	ラッパー・オブジェクトを戻します。
isConnected	データ・ソース・サーバーとの接続が存在するかどうかを検証します。
markDisconnected	データ・ソース・サーバーとの接続が終了したことを示すフラグを設定します。

表 129. RemoteConnection クラスのメソッド (続き)

メソッド	説明
rollback	トランザクションが失敗し、リモート・データ・ソースがトランザクションをロールバックすることを示します。

RemoteConnection コンストラクター

目的 指定されたユーザー許可とタイプを使用する、指定のデータ・ソース・サーバーの接続を構成します。

構文

```
protected RemoteConnection(FencedServer remoteServer,
                           FencedRemoteUser remoteUser,
                           int connectionKind,
                           long id)
```

パラメーター

表 130. RemoteConnection コンストラクターのパラメーター

名前	説明
remoteServer	接続が含まれるデータ・ソース・サーバー。
remoteUser	認証に使用するユーザー・マッピング。
connectionKind	接続のタイプ。サポートされるトランザクション・タイプを指定します。 1 フェーズ・コミットのトランザクションをサポートできる接続は、ONE_PHASE_KIND 定数で示します。トランザクションをサポートしない接続は、NO_PHASE_KIND 定数で示します。
id	RemoteConnection オブジェクトを表す整数値。

スロー なし。

commit メソッド

目的 トランザクションが正常に完了し、リモート・データ・ソースがトランザクションをコミットすることを示します。デフォルトのインプリメンテーションは何もしません。ラッパーはこのメソッドをオーバーライドして、リモート・データ・ソースで必要なコミット・ロジックをインプリメントできます。

構文

```
protected void commit()
          throws java.lang.Exception
```

パラメーター
なし。

戻り値 なし。

スロー メソッドが失敗した場合には、Exception オブジェクト。

connect メソッド

目的 データ・ソース・サーバーに接続するためのコードを呼び出します。デフォルトでは、このメソッドは何もしません。リモート・データ・ソースとの接続を確立するためにラッパー固有の接続クラスが何らかの操作を実行する必要がある場合、この connect メソッドをインプリメントできます。

構文

```
protected void connect()
            java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー メソッドが失敗した場合には、Exception オブジェクト。

createRemotePassthru メソッド

目的 RemotePassthru オブジェクトを作成して、パススルー・ステートメントを実行します。このメソッドは、デフォルトでは NULL を戻します。 しかしラッパーがパススルー・ステートメントを実行する能力をサポートする場合には、ラッパー固有の接続クラスでこのメソッドをインプリメントすることが必要です。こうした状況で createRemotePassthru メソッドを使用する場合、NULL 以外の値を戻さなければなりません。

構文

```
protected RemotePassthru createRemotePassthru(long id)
                                      throws java.lang.Exception
```

パラメーター

表 131. createRemotePassthru メソッドのパラメーター

名前	説明
id	RemotePassthru オブジェクトを表す整数値。

戻り値 RemotePassthru インスタンス、またはラッパーがこの操作をサポートしていない場合には NULL。

スロー オブジェクト作成に失敗した場合には、Exception オブジェクト。

createRemoteQuery メソッド

目的 SQL ステートメントを実行して、RemoteQuery オブジェクトを作成します。このメソッドは、デフォルトでは NULL を戻します。 しかしラッパーが SQL ステートメントを実行する能力をサポートする場合には、ラッパー固有の接続クラスでこのメソッドをインプリメントすることが必要です。こうした状況で createRemoteQuery メソッドを使用する場合、 NULL 以外の値を戻さなければなりません。

構文

```
protected RemoteQuery createRemoteQuery(long id)
                                 throws java.lang.Exception
```

パラメーター

表 132. createRemoteQuery メソッドのパラメーター

名前	説明
id	RemoteQuery オブジェクトを表す整数値。

戻り値 RemoteQuery インスタンス、またはラッパーがこの操作をサポートしていない場合には NULL。

スロー オブジェクト作成に失敗した場合には、Exception オブジェクト。

disconnect メソッド

目的 データ・ソース・サーバーから切断するためのコードを呼び出します。リモート・データ・ソースとの接続を終了するためにラッパー固有の接続クラスが何らかの操作を実行する必要がある場合、この disconnect メソッドをインプリメントできます。

構文

```
protected void disconnect()
                throws java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー メソッドが失敗した場合には、Exception オブジェクト。

getCodepage メソッド

目的 接続用のコード・ページを戻します。

構文

```
public final short getCodepage()
```

パラメーター

なし。

戻り値 コード・ページ。

スロー なし。

getKind メソッド

目的 接続タイプを戻します。接続は、1 フェーズ・コミットのトランザクションをサポートできるか (ONE_PHASE_KIND 定数で示される)、またはトランザクションをサポートできない (NO_PHASE_KIND 定数で示される) のいずれかです。

構文

```
public final int getKind()
```

パラメーター

なし。

戻り値 接続タイプ。

スロー なし。

getServer メソッド

目的 この接続が含まれるデータ・ソース・サーバー・オブジェクトを戻します。

構文

```
public final FencedServer getServer()
```

パラメーター
なし。

戻り値 接続用のデータ・ソース・サーバー。

スロー なし。

getUser メソッド

目的 この接続を認証するためのユーザー・マッピングを戻します。

構文

```
public final FencedRemoteUser getUser()
```

パラメーター
なし。

戻り値 この接続のユーザー・マッピング。

スロー なし。

getWrapper メソッド

目的 ラッパー・オブジェクトを戻します。

構文

```
public final Wrapper getWrapper()
```

パラメーター
なし。

戻り値 ラッパー・オブジェクト。

スロー なし。

isConnected メソッド

目的 データ・ソース・サーバーとの接続が存在するかどうかを検証します。

構文

```
public final boolean isConnected()
```

パラメーター
なし。

戻り値 データ・ソース・サーバーとの接続が存在するかどうかを示す値。

スロー なし。

markDisconnected メソッド

目的 データ・ソース・サーバーとの接続が終了したことを示すフラグを設定します。

構文

```
public final void markDisconnected()
```

パラメーター
なし。

戻り値 なし。

スロー なし。

rollback メソッド

目的 トランザクションが失敗し、リモート・データ・ソースがトランザクションをロールバックすることを示します。デフォルトのインプリメンテーションは何もしません。ラッパーはこのメソッドをオーバーライドして、リモート・データ・ソースで必要なロールバック・ロジックをインプリメントできます。

構文

```
protected void rollback()
              throws java.lang.Exception
```

パラメーター
なし。

戻り値 なし。

スロー メソッドが失敗した場合には、Exception オブジェクト。

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『リモート接続クラス』

関連資料:

- 83 ページの『FencedRemoteUser クラス (Java)』
- 69 ページの『FencedServer クラス (Java)』
- 110 ページの『RemotePassthru クラス (Java)』
- 114 ページの『RemoteQuery クラス (Java)』
- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

Java API の操作クラス

以下の表で、Java API の各操作クラスについて説明します。

表 133. 操作クラス

クラス名	説明
RemoteOperation	種々のリモート操作を処理する複数のクラスのベース・クラスを表すクラス。 そうしたリモート操作には、照会およびパススルー・セッションが含まれます。
RemotePassthru	リモート・データ・ソース上のパススルー・セッションを表すクラス。
RemoteQuery	リモート・データ・ソース上の SELECT ステートメント操作を表すクラス。

関連資料:

- 107 ページの『RemoteOperation クラス (Java)』
- 110 ページの『RemotePassthru クラス (Java)』
- 114 ページの『RemoteQuery クラス (Java)』

RemoteOperation クラス (Java)

このセクションでは、RemoteOperation クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RemoteOperation クラスは、種々のリモート操作を表すクラスのベース・クラスです。そうしたリモート操作には、照会およびパススルー・セッションが含まれます。 RemoteOperation クラスを直接インスタンス化したり、カスタマイズすることはできません。 RemoteOperation ベース・クラスのインプリメンテーションによって、以下の情報が保守されます。

- オブジェクトが表すリモート操作のタイプ。
- RemoteConnection オブジェクトへの参照。
- RuntimeDataList への参照。それには、SQL ステートメント内のパラメーター・マーカー (ある場合) にバインドされる値のタイプと位置が記述されています。
- RuntimeDataList への参照。それには、予想タイプが記述されていて、(ある場合には) SQL ステートメントによって戻される値のバッファーを提供します。

RemotePassthru および RemoteQuery クラスは、RemoteOperation クラスのサブクラスです。

RemoteOperation クラスは、Java API の操作クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、RemoteOperation クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 134. RemoteOperation クラスのメソッド

メソッド	説明
getConnection	リモート操作用の接続を戻します。
getExecDesc	リモート操作用の実行記述子を戻します。
getInputData	リモート操作用の入力値のリストを戻します。
getOutputData	リモート操作からの出力データ・バッファーのリストを戻します。

表 134. RemoteOperation クラスのメソッド (続き)

メソッド	説明
getServer	リモート操作を所有するデータ・ソース・サーバーを戻します。
getWrapper	ラッパー・オブジェクトを戻します。
reportEof	取り出し操作中にファイルの終わり状態をレポートします。

getConnection メソッド

目的 リモート操作用の接続を戻します。

構文

```
public final RemoteConnection getConnection()
```

パラメーター

なし。

戻り値 RemoteConnection インスタンス。

スロー なし。

getExecDesc メソッド

目的 リモート操作用の実行記述子を戻します。

構文

```
public final java.lang.Object getExecDesc()
                                    throws WrapperException,
                                           java.io.IOException,
                                           java.lang.ClassNotFoundException
```

パラメーター

なし。

戻り値 実行記述子。

スロー 以下の例外のいずれかです。

- 実行記述子オブジェクトの検索が失敗した場合には、WrapperException オブジェクト。
- 実行記述子のストリーミングが失敗した場合には、java.io.IOException。
- 実行記述子クラスが見つからない場合には、java.lang.ClassNotFoundException。

getInputData メソッド

目的 リモート操作用の入力値のリストを戻します。

構文

```
public final RuntimeDataList getInputData()
```

パラメーター

なし。

戻り値 入力値のリスト。

スロー なし。

getOutputData メソッド

目的 リモート操作用の出力データ・バッファーのリストを戻します。

構文

```
public final RuntimeDataList getOutputData()
```

パラメーター
なし。

戻り値 出力データ。

スロー なし。

getServer メソッド

目的 リモート操作用のデータ・ソース・サーバーを戻します。

構文

```
public final FencedServer getServer()
```

パラメーター
なし。

戻り値 リモート操作を実行する FencedServer インスタンス。

スロー なし。

getWrapper メソッド

目的 ラッパー・オブジェクトを戻します。

構文

```
public final Wrapper getWrapper()
```

パラメーター
なし。

戻り値 ラッパー・オブジェクト。

スロー なし。

reportEof メソッド

目的 取り出し操作中にファイルの終わり状態をレポートします。

構文

```
protected void reportEof()
                    throws WrapperException
```

パラメーター
なし。

戻り値 なし。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

関連資料:

- 69 ページの『FencedServer クラス (Java)』
- 176 ページの『RuntimeDataList クラス (Java)』
- 100 ページの『RemoteConnection クラス (Java)』

- 53 ページの『Wrapper クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

RemotePassthru クラス (Java)

このセクションでは、RemotePassthru クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

RemotePassthru クラスは、リモート・データ・ソース上のパススルー・セッションを表します。フェデレーテッド・サーバーが、適切な RemoteConnection サブクラス上に createRemotePassthru メソッドを呼び出すと、RemotePassthru サブクラスのインスタンスが作成されます。 ご使用のデータ・ソースがパススルー操作をサポートしていない場合、 RemotePassthru サブクラスをインプリメントしたり、このメソッドのデフォルトのインプリメンテーションをオーバーライドしたりしないでください。

public RemotePassthru クラスは、RemoteOperation クラスを拡張します。

RemotePassthru クラスは、Java API の操作クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表では、RemotePassthru クラスのコンストラクターおよびメソッドが記載されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 135. RemotePassthru クラスのコンストラクター

コンストラクター	説明
RemotePassthru	指定された接続のための新規パススルー・オブジェクトを構成します。

表 136. RemotePassthru クラスのメソッド

メソッド	説明
close	パススルー・カーソルをクローズし、パススルー・ステートメント処理後にデータ・ソースをクリーンアップできるようにします。
describe	パススルー操作が処理するステートメントの結果セットを記述します。
execute	パススルー操作によってコードは戻すものの、結果セットは戻さないステートメントを実行します。
fetch	単一結果行を出力データ・バッファーにコピーして、パススルー・カーソルから行を取り出します。

表 136. RemotePassthru クラスのメソッド (続き)

メソッド	説明
getSQLStatement	リモート・パススルー操作用の SQL ステートメントを戻します。
open	パススルー操作用のカーソルをオープンします。
prepare	パススルー操作を準備します。
reportEof	取り出し操作中にファイルの終わり状態をレポートします。

RemotePassthru コンストラクター

目的 指定された接続のための新規オブジェクトを構成します。

構文

```
protected RemotePassthru(RemoteConnection activeConnection,
                         long id)
```

パラメーター

表 137. RemotePassthru コンストラクターのパラメーター

名前	説明
activeConnection	パススルー・ステートメントをデータ・ソースにサブミットする接続。
id	RemotePassthru オブジェクトを表す整数値。

スロー なし。

close メソッド

目的 パススルー・カーソルをクローズし、パススルー・ステートメント処理後にデータ・ソースをクリーンアップできるようにします。

構文

```
protected void close()
          throws java.lang.Exception
```

パラメーター
なし。

戻り値 なし。

スロー クローズ操作が失敗した場合には、Exception オブジェクト。

describe メソッド

目的 パススルー操作が処理するステートメントの結果セットを記述します。各行の結果列の数およびタイプを記述する RuntimeDataDescList クラスにデータを追加します。

構文

```
protected void describe(RuntimeDataDescList dataDescriptionList)
              throws java.lang.Exception
```

パラメーター

表 138. describe メソッドのパラメーター

名前	説明
dataDescriptionList	操作用のデータ記述子のリスト。

戻り値 なし。

スロー メソッドが失敗した場合には、Exception オブジェクト。

execute メソッド

目的 パススルー操作によってコードは戻すものの、結果セットは戻さないステートメントを実行します。

構文

```
protected void execute()
            throws java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー 実行プロセスが失敗した場合には、Exception オブジェクト。

fetch メソッド

目的 単一結果行を出力データ・バッファーにコピーして、パススルー・カーソルから行を取り出します。

構文

```
protected void fetch()
          throws java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー 取り出しプロセスが失敗した場合には、Exception オブジェクト。

getSQLStatement メソッド

目的 リモート・パススルー操作用の SQL ステートメントを戻します。

構文

```
public final java.lang.String getSQLStatement()
                                        throws WrapperException
```

パラメーター

なし。

戻り値 SQL ステートメントを表すストリング値。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

open メソッド

目的 パススルー操作用のカーソルをオープンし、データ・ソースが照会の最初の結果行を戻せるようにします。

構文

```
protected void open()
          throws java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー オープン・プロセスが失敗した場合には、Exception オブジェクト。

prepare メソッド

目的 パススルー操作を準備します。パススルー・ストリングをデータ・ソースに送信し、各結果行の列数およびタイプを判別します。

構文

```
protected void prepare(RuntimeDataDescList dataDescriptionList)
             throws java.lang.Exception
```

パラメーター

表 139. prepare メソッドのパラメーター

名前	説明
dataDescriptionList	操作用のデータ記述子のリスト。

戻り値 なし。

スロー 準備操作が失敗した場合には、Exception オブジェクト。

reportEof メソッド

目的 取り出し操作中にファイルの終わり状態をレポートします。ラッパーは、戻す行がもうない場合には fetch() メソッドからこのメソッドを呼び出す必要があります。

構文

```
protected final void reportEof()
                     throws WrapperException
```

オーバーライド

RemoteOperation クラスの reportEof メソッド。

パラメーター

なし。

戻り値 なし。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

継承されるメソッド

以下のメソッドは、RemoteOperation クラスから継承され、RemotePassthru クラスと共に使用できます。

- getConnection
- getExecDesc
- getInputData
- getOutputData
- getServer
- getWrapper

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『リモート passthru クラス』

関連資料:

- 184 ページの『RuntimeDataDescList クラス (Java)』
- 100 ページの『RemoteConnection クラス (Java)』
- 107 ページの『RemoteOperation クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

RemoteQuery クラス (Java)

このセクションでは、RemoteQuery クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

RemoteQuery クラスは、リモート・データ・ソース上の SELECT ステートメント操作を扱います。 RemoteQuery サブクラスのインスタンスは、フェデレーテッド・サーバーが該当する RemoteConnection サブクラスのインスタンス上で RemoteConnection.createRemoteQuery(long) メソッドを呼び出す際に作成されます。フェデレーテッド・サーバーは、フェデレーテッド・サーバーへの照会をサブミットしたアプリケーションが照会の結果セット上でカーソルをクローズした後、RemoteQuery オブジェクトを破棄します。

public RemoteQuery クラスは RemoteOperation クラスを拡張します。

RemoteQuery クラスは、Java API の操作クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

クラス・フィールド

以下の表では、RemoteQuery クラスで使用できるオプションの有効なクラス・フィールドについて説明します。

表 140. *RemoteQuery* クラスのクラス・フィールド

クラス・フィールド	定義	説明
CLOSE_EOA	public static final byte CLOSE_EOA	ラッパーが必要なクリーンアップを完了すること、および close メソッドが戻された後に RemoteQuery オブジェクトが破棄されることを示します。この定数は、close メソッドのみで使用されます。
CLOSE_EOS	public static final byte CLOSE_EOS	ラッパーが RemoteQuery オブジェクトを、 reopen メソッドまたは reopenInputLob メソッドを呼び出せる状態にしておかなければならないことを示します。この定数は、close メソッドのみで使用されます。
EOF	public static final byte EOF	RemoteQuery オブジェクトが結果セットのすべての行を取り出した (戻した) ことを示しています。この定数は、getStatus メソッドおよび setStatus メソッドのみで使用されます。
FETCH_LOB_LAST	public static final int FETCH_LOB_LAST	現在の LOB 列のデータの最後のバッファーが取り出された (戻された) ことを示します。
FETCH_LOB_MORE	public static final int FETCH_LOB_MORE	データがさらに LOB 列で使用できることを示します。
FETCH_LOB_NEXT	public static final int FETCH_LOB_NEXT	次に取り出される (戻される) 項目が LOB 列であることを示します。
FETCH_NON_LOB_NEXT	public static final int FETCH_NON_LOB_NEXT	次に取り出される (戻される) 項目が非 LOB 列であることを示します。
FETCH_ROW_DONE	public static final int FETCH_ROW_DONE	現在行のすべての値が取り出された (戻された) ことを示します。
OPEN	public static final byte OPEN	RemoteQuery オブジェクトがオープン状態であることを示します。この定数は、getStatus メソッドおよび setStatus メソッドのみで使用されます。
OPEN_LOB_INIT	public static final int OPEN_LOB_INIT	入力 LOB 値 の処理が開始することを示します。

表 140. RemoteQuery クラスのクラス・フィールド (続き)

クラス・フィールド	定義	説明
OPEN_LOB_LAST	public static final int OPEN_LOB_LAST	これが現行の入力 LOB 値の最後のフラグメントであることを示します。
OPEN_LOB_MORE	public static final int OPEN_LOB_MORE	フラグメントがさらに現在の入力 LOB 値で使用できることを示します。
OPEN_NON_LOB_NEXT	public static final int OPEN_NON_LOB_NEXT	入力の非 LOB 値のみが処理されることを示します。
OPEN_LOB_NEW	public static final int OPEN_LOB_NEW	ラッパーが新規の入力 LOB 値のデータ・フラグメントを予期することを示します。
OPEN_ROW_DONE	public static final int OPEN_ROW_DONE	すべての入力値が処理されたことを示します。
READY	public static final byte READY	RemoteQuery オブジェクトが次の行を取り出す (取り戻す) 準備ができていることを示します。この定数は、getStatus メソッドおよび setStatus メソッドのみで使用されます。
UNREADY	public static final byte UNREADY	RemoteQuery オブジェクトが次の行を取り出す準備ができていないことを示します。この定数は、getStatus メソッドおよび setStatus メソッドのみで使用されます。

コンストラクターおよびメソッド

以下の表で、RemoteQuery クラスのコンストラクターおよびメソッドについて説明します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 141. RemoteQuery クラスのコンストラクター

コンストラクター	説明
RemoteQuery	指定された接続のための新規照会オブジェクトを構成します。

表 142. RemoteQuery クラスのメソッド

メソッド	説明
close	ラッパーおよびデータ・ソースが照会ステートメントの処理後にクリーンアップできるようにします。

表 142. RemoteQuery クラスのメソッド (続き)

メソッド	説明
fetch	リモート・データ・ソース照会から単一の結果行を取り出し (取り戻し)、出力データ・バッファーに入れます。
fetchLob	リモート・データ・ソース照会からラージ・オブジェクト (LOB) フラグメントを戻します。
getRowStatus	現在の行状況を戻します。
getStatus	照会の状況を戻します。
lobDataReady	LOB 列からデータのフラグメントが取り出される (戻される) ことをフェデレーテッド・サーバーに通知します。
open	ラッパーがデータ・ソースを準備して照会の最初の結果行を戻せるようにします。
openInputLob	ラッパーがリモート・データ・ソースを準備して、入力 LOB パラメーター照会の最初の結果行を戻せるようにします。
reopen	以前に開かれた結果ストリームをリセットし、さらに結果を戻すためにデータ・ソースを準備します。これらの結果は、パラメーター・バインディングによって異なる場合があります。
reopenInputLob	以前に開かれた結果ストリームをリセットし、さらに結果セットを戻すためにデータ・ソースを準備します。これらの結果は、入力 LOB パラメーターを使用した照会のパラメーター・バインディングによって異なる場合があります。
reportEof	取り出し (取り戻し) 操作中にファイルの終わり状態をレポートします。
setLobNext	現在の取り出された (戻された) 行に LOB が含まれるとき、および fetchLob メソッドが呼び出されることをフェデレーテッド・サーバーに通知します。
setRowStatus	現在の行状況を設定します。
setStatus	照会状況を設定します。

RemoteQuery コンストラクター

目的 指定された接続のための新規照会オブジェクトを構成します。

構文

```
protected RemoteQuery(RemoteConnection activeConnection,
                      long id)
```

パラメーター

表 143. RemoteQuery コンストラクターのパラメーター

名前	説明
activeConnection	照会を実行するために使用する接続。
id	RemoteQuery オブジェクトを表す整数値。

スロー なし。

close メソッド

目的 ラッパーおよびデータ・ソースが照会ステートメントの処理後にクリーンアップできるようにします。ラッパーは、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントする必要があります。

使用法 フェデレーテッド・サーバーに対してサブミットされる単一 SQL ステートメントは、ラッパーへの複数の照会要求につながる可能性があります。ラッパーがデータ・ソースへの照会の変換およびサブミットをさらに簡単に最適化できるように、 close メソッドは、クローズが end-of-query (CLOSE_EOS) と end-of-statement (CLOSE_EOA) 状況のどちらを表すかを示す状況フラグを受け取ります。 close メソッドが end-of-query 状況とともに呼び出されると、RemoteQuery オブジェクトは reopen メソッドを正常に実行することができます。 close メソッドが end-of-statement 状況とともに呼び出されると、フェデレーテッド・サーバーは、close メソッドの完了時に RemoteQuery オブジェクトを破棄します。

構文

```
protected void close(short status)
          throws java.lang.Exception
```

パラメーター

表 144. close メソッドのパラメーター

名前	説明
status	操作の状況。

戻り値 なし。

スロー クローズ操作が失敗した場合には、Exception オブジェクト。

fetch メソッド

目的 リモート・データ・ソース照会から単一の結果行を取り出し (取り戻し)、出力データ・バッファーに入れます。ラッパーは、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントする必要があります。

構文

```
protected void fetch()
          throws java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー 取り出し (取り戻し) プロセスが失敗した場合には、Exception オブジェクト。

fetchLob メソッド

目的 リモート・データ・ソース照会からラージ・オブジェクト (LOB) フラグメントを戻します。ラッパーは、LOB を転送した場合に、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントすることができます。

使用法 fetchLob メソッドには、取り出される (取り戻される) データ・フラグメントの最大サイズを指定する関数引き数が含まれます。 fetchLob メソッドは、lobDataReady メソッドを呼び出し、フェデレーテッド・サーバーに取り出し状況を通知する必要があります。

構文

```
protected java.lang.Object fetchLob(int bufferSize,
                                    int bytesWritten)
                            throws java.lang.Exception
```

パラメーター

表 145. fetchLob メソッドのパラメーター

名前	説明
bufferSize	取り出す (取り戻す) データ・フラグメントの最大サイズ。
bytesWritten	現行 LOB 用にすでに取り出された (取り戻された) バイト数。

戻り値 データ・フラグメント。このデータ・フラグメントは CLOB データではストリングであり、BLOB データではバイト配列です。

スロー fetchLob メソッドが失敗した場合は、WrapperException オブジェクト。

getRowStatus メソッド

目的 現在の行状況を戻します。

構文

```
protected final int getRowStatus()
```

パラメーター

なし。

戻り値 現在の行状況。

スロー なし。

getStatus メソッド

目的 照会の状況を戻します。

構文

```
protected final byte getStatus()
```

パラメーター

なし。

戻り値 照会状況。

スロー なし。

lobDataReady メソッド

目的 LOB 列からデータのフラグメントが取り出される (戻される) ことをフェデレーテッド・サーバーに通知します。このメソッドを fetchLob メソッドから呼び出します。

構文

```
protected final void lobDataReady(int columnNumber,
                                  int bytesReady,
                                  int status,
                                  int intent)
                           throws WrapperException
```

パラメーター

表 146. lobDataReady メソッドのパラメーター

名前	説明
columnNumber	このデータ・フラグメントが属する列索引。
bytesReady	フラグメントの長さ (バイト)。
status	LOB 操作の状況。状況値は、FETCH_LOB_MORE または FETCH_LOB_LAST になります。
intent	さらに取り出す (取り戻す) LOB または非 LOB 列があるか、あるいは完全な行が取り出されたかを示すフラグ。

戻り値 なし。

スロー 取り出し (取り戻し) プロセスが失敗した場合には、WrapperException オブジェクト。

open メソッド

目的 ラッパーがデータ・ソースを準備して照会の最初の結果行を戻せるようにします。ラッパーは、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントする必要があります。

構文

```
protected void open()
          throws java.lang.Exception
```

パラメーター

なし。

戻り値 なし。

スロー オープン・プロセスが失敗した場合には、Exception オブジェクト。

openInputLob メソッド

目的 ラッパーがデータ・ソースを準備して、LOB が含まれる照会の最初の結果

行を戻せるようにします。ラッパーは、入力 LOB パラメーターをサポートする場合、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントする必要があります。

使用法 フェデレーテッド・サーバーは、入力 LOB ホスト変数が見つかった場合にこのメソッドを呼び出します。最初に、openInputLob メソッドが、空のバッファーおよび -1 に設定された columnNumber パラメーターとともに呼び出されます。ラッパーは、次の LOB フラグメントを受け取るホスト変数の索引を戻し、入力 LOB パラメーターが存在することを示す setRowStatus メソッドを呼び出す必要があります。処理する LOB 入力パラメーターがさらに存在することをラッパーが示した場合、フェデレーテッド・サーバーは openInputLob メソッドを別の LOB フラグメントとともに呼び出します。LOB 入力値の合計サイズ (バイト) が matSize パラメーターとして指定されます。現在の LOB フラグメントのサイズ (バイト) が xferBytes パラメーターとして指定されます。ラッパーはこれらのパラメーター値を使用して、次の入力変数に進むか、あるいは全体の入力値が読み取られたことを知らせる必要があります。

構文

```
protected int openInputLob(int columnNumber,
                           int matSize,
                           int xferBytes,
                           java.lang.Object buffer)
                    throws java.lang.Exception
```

パラメーター

表 147. lobDataReady メソッドのパラメーター

名前	説明
columnNumber	現在の LOB フラグメントを受け取る入力ホスト変数のインデックス。
matSize	入力データのマテリアライズ・サイズ (バイト)。
xferBytes	現在のデータ・フラグメントのサイズ (バイト)。
buffer	現在の LOB フラグメント・オブジェクト。LOB フラグメント・オブジェクトは、CLOB 変数ではストリングであり、BLOB 変数ではバイト配列です。

戻り値 次の LOB フラグメントを受け取る入力ホスト変数のインデックス。

スロー openInputLob プロセスが失敗した場合には、Exception オブジェクト。

reopen メソッド

目的 以前に開かれた結果ストリームをリセットし、さらに結果を戻すためにデータ・ソースを準備します。これらの結果は、パラメーター・バインディングによって異なる場合があります。ラッパーは、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントする必要があります。照会が end-of-query の状態で以前にクローズされた場合を除いて、reopen メソッドは呼び出されません。

構文

```
protected void reopen(short action)
              throws java.lang.Exception
```

パラメーター

表 148. reopen メソッドのパラメーター

名前	説明
action	使用されません。

戻り値 なし。

スロー reopen プロセスが失敗した場合には、Exception オブジェクト。

reopenInputLob メソッド

目的 以前に開かれた結果ストリームをリセットし、さらに結果セットを戻すためにデータ・ソースを準備します。これらの結果は、入力 LOB パラメーターを使用した照会のパラメーター・バインディングによって異なる場合があります。ラッパーは、入力 LOB パラメーターをサポートする場合、このメソッドをラッパー固有の RemoteQuery サブクラスにインプリメントする必要があります。

使用法 照会が end-of-query の状態で以前にクローズされた場合を除いて、このメソッドは呼び出されません。フェデレーテッド・サーバーは、入力 LOB ホスト変数が見つかった場合にこのメソッドを呼び出します。最初に、reopenInputLob メソッドが、空のバッファーおよび -1 に設定された columnNumber パラメーターとともに呼び出されます。ラッパーは、次の LOB フラグメントを受け取るホスト変数の索引を戻し、入力 LOB パラメーターが存在することを示す setRowStatus メソッドを呼び出す必要があります。処理する LOB 入力パラメーターがさらに存在することをラッパーが示した場合、フェデレーテッド・サーバーは reopenInputLob メソッドを別の LOB フラグメントとともに呼び出します。 LOB 入力値の合計サイズ (バイト) が matSize パラメーターとして指定されます。現在の LOB フラグメントのサイズ (バイト) が xferBytes パラメーターとして指定されます。ラッパーはこれらのパラメーター値を使用して、次の入力変数に進むか、あるいは全体の入力値が読み取られたことを知らせる必要があります。

構文

```
protected int reopenInputLob(short action,
                             int columnNumber,
                             int matSize,
                             int xferBytes,
                             java.lang.Object buffer)
                      throws java.lang.Exception
```

パラメーター

表 149. reopenInputLob メソッドのパラメーター

名前	説明
action	使用されません。
columnNumber	現在の LOB フラグメントを受け取る入力ホスト変数のインデックス。

表 149. reopenInputLob メソッドのパラメーター (続き)

名前	説明
matSize	入力データのマテリアライズ・サイズ (バイト)。
xferBytes	現在のデータ・フラグメントのサイズ (バイト)。
buffer	現在の LOB フラグメント・オブジェクト。LOB フラグメント・オブジェクトは、CLOB 変数ではストリングであり、BLOB 変数ではバイト配列です。

戻り値 次の LOB フラグメントを受け取る入力ホスト変数のインデックス。

スロー reopenInputLob プロセスが失敗した場合には、Exception オブジェクト。

reportEof メソッド

目的 取り出し (取り戻し) 操作中にファイルの終わり状態をレポートします。ラッパー固有の RemoteQuery サブクラスは、最後の行が取り出されたことを示す取り出し操作中にこのメソッドを呼び出します。

構文

```
protected final void reportEof()
                     throws WrapperException
```

オーバーライド
RemoteOperation クラスの reportEof メソッド。

パラメーター
なし。

戻り値 なし。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

setLobNext メソッド

目的 現在の取り出された (戻された) 行に LOB が含まれるとき、および fetchLob メソッドが呼び出されることをフェデレーテッド・サーバーに通知します。

使用法 転送する LOB データがある場合、fetch() メソッドからこのメソッドを呼び出します。 fetchLob メソッド内からは setLobNext メソッドを呼び出しません。 fetch メソッドが戻され、setLobNext が呼び出された後、フェデレーテッド・サーバーは fetchLob を呼び出して、LOB 列値を取り戻します。

構文

```
protected final void setLobNext()
```

パラメーター
なし。

戻り値 なし。

スロー なし。

setRowStatus メソッド

目的 現在の行状況を設定します。

構文

```
pprotected final void setRowStatus(int status)
```

パラメーター

表 150. setStatus メソッドのパラメーター

名前	説明
status	設定する新規の行状況。

戻り値 なし。

スロー なし。

setStatus メソッド

目的 照会状況を設定します。

構文

```
protected final void setStatus(byte status)
```

パラメーター

表 151. setStatus メソッドのパラメーター

名前	説明
status	設定する新規の照会状況。

戻り値 なし。

スロー なし。

継承されるメソッド

以下のメソッドは、RemoteOperation クラスから継承され、RemoteQuery クラスとともに使用できます。

- getConnection
- getExecDesc
- getInputData
- getOutputData
- getServer
- getWrapper

関連資料:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『リモート照会クラス』
- 100 ページの『RemoteConnection クラス (Java)』
- 107 ページの『RemoteOperation クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

Java API のプランニング・クラス

以下の表で、Java API の各プランニング・クラスについて説明します。

表 152. 要求クラス

クラス名	説明
ParsedQueryFragment	照会フラグメントを記述するクラス。
Request	ラッパーにより分析および処理される照会フラグメントをカプセル化するクラス。
Reply	ラッパーが処理する要求の一部を扱うクラス。
RequestExp	式ツリー内のノードを扱うクラス。
RequestExpType	式ツリー内のノードのタイプを記述するクラス。
RequestConstant	照会プランニングの際に照会式で使用される定数を記述するクラス。
Quantifier	要求の数量詞の情報をカプセル化するクラス。
PredicateList	選択性要因を推定するために述部の 2 つのリスト (選択性を請求する述部のリスト、および以前に適用された述部のリスト) を記述するクラス。

関連資料:

- 151 ページの『PredicateList クラス (Java)』
- 125 ページの『ParsedQueryFragment クラス (Java)』
- 150 ページの『Quantifier クラス (Java)』
- 130 ページの『Reply クラス (Java)』
- 145 ページの『RequestConstant クラス (Java)』
- 129 ページの『Request クラス (Java)』
- 142 ページの『RequestExpType クラス (Java)』
- 138 ページの『RequestExp クラス (Java)』

ParsedQueryFragment クラス (Java)

このセクションでは、ParsedQueryFragment クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

ParsedQueryFragment クラスは、Request クラスと Reply クラスの両方のベース・クラスを表します。 ParsedQueryFragment クラスは、照会フラグメントを記述します。照会フラグメントは、SELECT、FROM、WHERE、および ORDER BY 文節で構成されています。各文節は、オブジェクトの配列で表されます。このクラスのメソッドは、配列のサイズ、項目、および関連するデータ構造を戻します。

Reply クラスと Request クラスは、ParsedQueryFragment クラスのサブクラスです。

ParsedQueryFragment クラスは、Java API のプランニング・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、ParsedQueryFragment クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 153. ParsedQueryFragment クラスのメソッド

メソッド	説明
getDistinct	DISTINCT 標識をテストします (SELECT DISTINCT 文節)。
getHeadExp	SELECT 文節内の指定された位置の式を戻します。
getNickname	FROM 文節内の指定された位置に対応するニックネームを戻します。
getNumberOfHeadExp	SELECT 文節内のエレメントの数を戻します。
getNumberOfPredicates	WHERE 文節内のエレメント (述部) の数を戻します。
getNumberOfQuantifiers	FROM 文節内のエレメントの数 (数量詞) を戻します。
getPredicate	WHERE 文節内の指定された位置の述部式を戻します。
getQuantifier	FROM 文節内の指定された位置の数量詞を戻します。
getQuantiferByHandle	FROM 文節内の指定されたハンドルを持つ数量詞を戻します。

getDistinct メソッド

目的 DISTINCT 標識をテストします (SELECT DISTINCT 文節)。

構文

```
public final boolean getDistinct()
```

パラメーター

なし。

戻り値 DISTINCT 標識。

スロー なし。

getHeadExp メソッド

目的 SELECT 文節内の指定された位置の式を戻します。

構文

```
public final RequestExp getHeadExp(int position)
```

パラメーター

表 154. getHeadExp メソッドのパラメーター

名前	説明
position	SELECT 文節内のヘッド式の位置。最初のヘッド式の位置は 1 です。

戻り値 指定された位置のヘッド式。

スロー なし。

getNickname メソッド

目的 FROM 文節内の指定された位置に対応するニックネームを戻します。

構文

```
public final UnfencedGenericNickname getNickname(int position)
```

パラメーター

表 155. getNickname メソッドのパラメーター

名前	説明
position	FROM 文節内の数量詞の位置。最初の数量詞の位置は 1 です。

戻り値 指定された位置のニックネーム・オブジェクト。

スロー なし。

getNumberOfHeadExp メソッド

目的 SELECT 文節内のエレメントの数を戻します。

構文

```
public final int getNumberOfHeadExp()
```

パラメーター

なし。

戻り値 ヘッド式の数。

スロー なし。

getNumberOfPredicates メソッド

目的 WHERE 文節内のエレメント (述部) の数を戻します。

構文

```
public final int getNumberOfPredicates()
```

パラメーター

なし。

戻り値 述部の数。

スロー なし。

getNumberOfQuantifiers メソッド

目的 FROM 文節内のエレメントの数 (数量詞) を戻します。

構文

```
public final int getNumberOfQuantifiers()
```

パラメーター

なし。

戻り値 数量詞の数。

スロー なし。

getPredicate メソッド

目的 WHERE 文節内の指定された位置の述部式を戻します。

構文

```
public final RequestExp getPredicate(int position)
```

パラメーター

表 156. getPredicate メソッドのパラメーター

名前	説明
position	WHERE 文節内の述部の位置。最初の述部の位置は 1 です。

戻り値 指定された位置の述部式。

スロー なし。

getQuantifier メソッド

目的 FROM 文節内の指定された位置の数量詞を戻します。

構文

```
public final Quantifier getQuantifier(int position)
```

パラメーター

表 157. getQuantifier メソッドのパラメーター

名前	説明
position	FROM 文節内の数量詞位置。最初の数量詞の位置は 1 です。

戻り値 指定された位置の数量詞。

スロー なし。

getQuantiferByHandle メソッド

目的 FROM 文節内の指定されたハンドルを持つ数量詞を戻します。

構文

```
public final Quantifier getQuantiferByHandle(int quantifierHandle)
```

パラメーター

表 158. getQuantiferByHandle メソッドのパラメーター

名前	説明
quantifierHandle	数量詞ハンドル。

戻り値 指定されたハンドルの数量詞オブジェクト。

スロー なし。

関連資料:

- 150 ページの『Quantifier クラス (Java)』
- 130 ページの『Reply クラス (Java)』
- 129 ページの『Request クラス (Java)』
- 138 ページの『RequestExp クラス (Java)』
- 98 ページの『UnfencedGenericNickname クラス (Java)』

Request クラス (Java)

このセクションでは、Request クラスについて説明します。

概要

Request クラスは、ラッパーにより分析および処理される照会フラグメントをカプセル化します。フェデレーテッド・サーバーは、適切な Reply オブジェクトを戻す UnfencedGenericServer.planRequest メソッドにこのクラスのメソッドのインスタンスを提供します。

public Request クラスは、ParsedQueryFragment クラスを拡張します。

Request クラスは、Java API のプランニング・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

継承されるメソッド

以下のメソッドは、ParsedQueryFragment クラスから継承され、Request クラスと共に使用できます。

- getDistinct
- getHeadExp

- getNickname
- getNumberOfHeadExp
- getNumberOfPredicates
- getNumberOfQuantifiers
- getPredicate
- getQuantifier
- getQuantiferByHandle

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『Request クラス』

関連資料:

- 125 ページの『ParsedQueryFragment クラス (Java)』
- 130 ページの『Reply クラス (Java)』
- 76 ページの『UnfencedGenericServer クラス (Java)』

Reply クラス (Java)

このセクションでは、Reply クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

Reply クラスは、ラッパーが処理する要求の一部を扱います。 ParsedQueryFragment クラスのアクションのほかに、 Reply クラスはメソッドおよびデータ・メンバーを加えて、以下のアクションを開始します。

- 文節に項目を追加して、応答にデータを設定する。
- 実行記述子構造を保管する。
- ラッパーが同一の要求に対して複数のプランを戻す場合、応答をチェーニングする。
- オーダー情報を保管する。より適したプランを構成するため、オプティマイザーは戻されたデータを別の順序で使用する可能性があります。

public Request クラスは、ParsedQueryFragment クラスを拡張します。

Reply クラスは、Java API のプランニング・クラスの 1 つです。

使用法 応答照会フラグメントに実行コスト (推定実行時間) を使用すると、 Reply クラスで高機能カスタマイズを利用できます。ラッパー・ライターは、ソースの実行モデルをより正確に記述したメソッドによって、コスト計算のデフォルト・インプリメンテーションをオーバーライドできます。デフォルトのコスト計算のオーバーライドは、照会フラグメントの述部に対するデフォルトの選択性推定に依存します。 UnfencedGenericServer.getSelectivity メソッドを呼び出すと、選択性推定値を取得できますが、ラッパー・ライターによって多重定義される可能性もあります。

パッケージ

com.ibm.db2.wrapper

定数

以下の表には、Reply クラスで使用できる有効な定数が記載されています。

表 159. Reply クラスの定数

定数	定義	説明
ASC	static final int ASC	昇順のオーダー・エントリーを指定する定数。
DESC	public static final int DESC	降順のオーダー・エントリーを指定する定数。

コンストラクターおよびメソッド

以下の表では、Reply クラスのコンストラクターおよびメソッドが記載されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 160. Reply クラスのコンストラクター

コンストラクター	説明
Reply	指定されたデータ・ソース (サーバー) の応答および要求をインスタンス化します。

表 161. Reply クラスのメソッド

メソッド	説明
addHeadExp	照会の SELECT 文節内にヘッド式を追加します。
addPredicate	照会の WHERE 文節内に述部を追加します。
addQuantifier	照会の FROM 文節内に数量詞を追加します。
addOrderEntry	照会の ORDER BY 文節内にオーダー・エントリーを追加します。
cardinality	応答が処理する照会フラグメントの実行時に、結果のカーディナリティーを戻します。
firstTupleCost	応答によって表される、照会フラグメントの最初の結果タプルを取得するのに必要なコストを戻します。
getExecDesc	実行記述子オブジェクトを戻します。
getNumberOfOrderEntries	オーダー項目の数を戻します。
getOrderEntry	ORDER BY 文節内の指定された位置のオーダー・エントリーを戻します。
getServer	この応答が属するデータ・ソース・サーバーを戻します。
nextReply	チェーン内の次の応答を戻します。
reExecCost	応答が処理する照会フラグメントを再実行するのに必要なコストを戻します。

表 161. Reply クラスのメソッド (続き)

メソッド	説明
setDistinct	SELECT DISTINCT ステートメントで指定された DISTINCT 標識を設定します。
setExecDesc	実行記述子オブジェクトを設定します。
setNextReply	チェーン内の次の応答を指定します。
totalCost	応答が処理する照会フラグメントを実行するのに必要な合計実行コストを戻します。

Reply コンストラクター

目的 指定されたデータ・ソース (サーバー) の応答および要求をインスタンス化します。

構文

```
public Reply(Request request,
             UnfencedGenericServer server)
```

パラメーター

表 162. Reply コンストラクターのパラメーター

名前	説明
request	Reply が生成される Request オブジェクト。 Request オブジェクトは、ラッパーにより分析および処理される照会フラグメントをカプセル化します。
server	要求を処理し、応答を生成する Server オブジェクト。

addHeadExp メソッド

目的 照会の SELECT 文節内にヘッド式を追加します。

構文

```
public final void addHeadExp(RequestExp headExp)
```

パラメーター

表 163. addHeadExp メソッドのパラメーター

名前	説明
headExp	ヘッド式。

戻り値 なし。

スロー なし。

addPredicate メソッド

目的 照会の WHERE 文節内に述部を追加します。

構文

```
public final void addPredicate(RequestExp predicate)
```

パラメーター

表 164. addPredicate メソッドのパラメーター

名前	説明
predicate	追加する述部。

戻り値 なし。

スロー なし。

addQuantifier メソッド

目的 照会の FROM 文節内に数量詞を追加します。

構文

```
public final void addQuantifier(Quantifier quantifier)
```

パラメーター

表 165. addQuantifier メソッドのパラメーター

名前	説明
quantifier	追加する数量詞。

戻り値 なし。

スロー なし。

addOrderEntry メソッド

目的 照会の ORDER BY 文節内にオーダー・エントリーを追加します。オーダー・エントリーは、ヘッド式の位置および方向 (ASC または DESC) で構成されています。

構文

```
public final void addOrderEntry(int position,
                                 int direction)
                          throws WrapperException
```

パラメーター

表 166. addOrderEntry メソッドのパラメーター

名前	説明
position	SELECT 文節内のヘッド式の位置。
direction	方向。昇順 (ASC) オーダーまたは降順 (DESC) オーダー。

戻り値 なし。

スロー 方向が無効の場合には、WrapperException オブジェクト。

cardinality メソッド

目的 応答が処理する照会フラグメントの実行時に、結果のカーディナリティーを戻します。

構文

```
public double cardinality()
```

パラメーター
なし。

戻り値 カーディナリティー。

スロー なし。

firstTupleCost メソッド

目的 応答によって表される、照会フラグメントの最初の結果タプルを取得するのに必要なコストを戻します。

構文

```
public double firstTupleCost()
```

パラメーター
なし。

戻り値 最初のタプルのコスト。

スロー なし。

getExecDesc メソッド

目的 実行記述子オブジェクトを戻します。

構文

```
public final java.io.Serializable getExecDesc()
```

パラメーター
なし。

戻り値 実行記述子。

スロー なし。

getNumberOfOrderEntries メソッド

目的 オーダー項目の数を戻します。

構文

```
public final int getNumberOfOrderEntries()
```

パラメーター
なし。

戻り値 オーダー項目の数。

スロー なし。

getOrderEntry メソッド

目的 ORDER BY 文節内の指定された位置のオーダー・エントリーを戻します。オーダー・エントリーは、ヘッド式の位置および方向 (ASC または DESC) で構成されています。

構文

```
public final int[] getOrderEntry(int position)
                            throws WrapperException
```

パラメーター

表 167. getOrderEntry メソッドのパラメーター

名前	説明
position	ORDER BY 文節内のオーダー・エントリーの位置。最初の項目の位置は 1 です。

戻り値 2 つの整数値の配列です。 2 つの整数値は、SELECT 文節のヘッド式とオーダー・タイプで、ASC (昇順) または DESC (降順) です。

スロー 要求された位置が無効の場合には、WrapperException オブジェクト。

getServer メソッド

目的 この応答が属するデータ・ソース・サーバーを戻します。

構文

```
public final UnfencedGenericServer getServer()
```

パラメーター
なし。

戻り値 応答のデータ・ソース・サーバー。

スロー なし。

nextReply メソッド

目的 チェーン内の次の応答を戻します。

構文

```
public final Reply nextReply()
```

パラメーター
なし。

戻り値 チェーン内の次の応答、または応答がない場合には NULL。

スロー なし。

reExecCost メソッド

目的 応答が処理する照会フラグメントを再実行するのに必要なコストを戻します。

構文

```
public double reExecCost()
```

パラメーター

なし。

戻り値 再実行コスト。

スロー なし。

setDistinct メソッド

目的 SELECT DISTINCT 文節で指定された DISTINCT 標識を設定します。

構文

```
public final void setDistinct(boolean distinct)
```

パラメーター

表 168. setDistinct メソッドのパラメーター

名前	説明
distinct	DISTINCT 標識。

戻り値 なし。

スロー なし。

setExecDesc メソッド

目的 実行記述子オブジェクトを設定します。

構文

```
public final void setExecDesc(java.io.Serializable execDesc)
```

パラメーター

表 169. setExecDesc メソッドのパラメーター

名前	説明
execDesc	実行記述子。

戻り値 なし。

スロー なし。

setNextReply メソッド

目的 チェーン内の次の応答を設定します。ラッパーが要求に複数の応答を戻す場合には、応答をチェーニングします。

構文

```
public final void setNextReply(Reply next)
```

パラメーター

表 170. setNextReply メソッドのパラメーター

名前	説明
next	チェーンに追加される Reply オブジェクト。

戻り値 なし。

スロー なし。

totalCost メソッド

目的 応答が処理する照会フラグメントを実行するのに必要な合計実行コストを戻します。

構文

```
public double totalCost()
```

パラメーター

戻り値 合計コスト。

スロー なし。

継承されるメソッド

以下のメソッドは、ParsedQueryFragment クラスから継承され、Reply クラスと共に使用できます。

- getDistinct
- getHeadExp
- getNickname
- getNumberOfHeadExp
- getNumberOfPredicates
- getNumberOfQuantifiers
- getPredicate
- getQuantifier
- getQuantifierByHandle

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『Reply クラス』

関連資料:

- 125 ページの『ParsedQueryFragment クラス (Java)』
- 129 ページの『Request クラス (Java)』
- 76 ページの『UnfencedGenericServer クラス (Java)』

RequestExp クラス (Java)

このセクションでは、RequestExp クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RequestExp クラスは、式ツリー内のノードを扱います。ノードは列参照、定数値、アンバウンド・パラメーター、または演算子のいずれかです。アンバウンド・パラメーターは、定数に似ています。 しかし、フェデレーテッド・サーバーがラッパーにアンバウンド・パラメーター値を渡す実行時フェーズまでは、この値は不明です。

RequestExp クラスは、Java API のプランニング・クラスの 1 つです。

使用法 RequestExp オブジェクトは、ラッパーによりインスタンス化されません。フェデレーテッド・サーバーは、こうしたオブジェクトを作成し、照会プランニングの際にラッパーに渡します。

パッケージ

com.ibm.db2.wrapper

定数

以下の表には、RequestExp クラスで使用できる、このオプションの有効な定数が記載されています。

表 171. RequestExp クラスの定数

定数	定義	説明
BADKIND	public static final int BADKIND	式が不明であることを示します。
COLUMN	public static final int COLUMN	式が列であることを示します。
CONSTANT	public static final int CONSTANT	式が定数であることを示します。
OPERATOR	public static final int OPERATOR	式が演算子であることを示します。
UNBOUND	public static final int UNBOUND	式がアンバウンド・パラメーターであることを示します。

メソッド

次の表は、RequestExp クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 172. RequestExp クラスのメソッド

メソッド	説明
getColumnName	列名を戻します。
getDataType	式のデータ・タイプを記述するオブジェクトを戻します。

表 172. *RequestExp* クラスのメソッド *(続き)*

メソッド	説明
getFirstChild	演算子の最初の子を戻します。
getHandle	式のハンドルを戻します。
getKind	式の種類を戻します。
getNextChild	式の兄弟を戻します。
getNumberofChildren	演算子の子の数を戻します。
getParent	式の親ノードを戻します。
getQuantifier	この列に属する数量詞を戻します。
getSignature	演算子のシグニチャーを戻します。
getToken	演算子のトークンを戻します。
getValue	定数の値を表す RequestConstant オブジェクトを戻します。

getColumnName メソッド

目的 列名 (列式ノード) を戻します。

構文

```
public java.lang.String getColumnName()
          throws WrapperException
```

パラメーター
なし。

戻り値 列名。

スロー 列以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

getDataType メソッド

目的 式のデータ・タイプを記述するオブジェクトを戻します。

構文

```
public RequestExpType getDataType()
```

パラメーター
なし。

戻り値 式タイプ。

スロー なし。

getFirstChild メソッド

目的 演算子 (演算子式ノード) の最初の子を戻します。

構文

```
public RequestExp getFirstChild()
          throws WrapperException
```

パラメーター
なし。

戻り値 最初の子。

スロー 演算子以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

getHandle メソッド

目的 式のハンドルを戻します。

構文

```
public int getHandle()
```

パラメーター
なし。

戻り値 ハンドル。

スロー なし。

getKind メソッド

目的 式の種類を戻します。

構文

```
public int getKind()
```

パラメーター
なし。

戻り値 式の種類。

スロー なし。

getNextChild メソッド

目的 式の兄弟を戻します。

構文

```
public RequestExp getNextChild()
```

パラメーター
なし。

戻り値 次の兄弟。または、式に兄弟がない場合には NULL。

スロー なし。

getNumberOfChildren メソッド

目的 演算子 (演算子式ノード) の子の数を戻します。

構文

```
public int getNumberOfChildren()
          throws WrapperException
```

パラメーター
なし。

戻り値 子の数。

スロー 演算子以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

getParent メソッド

目的 式の親ノードを戻します。

構文

```
public RequestExp getParent()
```

パラメーター
なし。

戻り値 親式。

スロー なし。

getQuantifier メソッド

目的 この列に属する列式ノード (数量詞) を戻します。

構文

```
public Quantifier getQuantifier()
        throws WrapperException
```

パラメーター
なし。

戻り値 数量詞。

スロー 列以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

getSignature メソッド

目的 演算子 (演算子式ノード) のシグニチャーを戻します。

構文

```
public java.lang.String getSignature()
        throws WrapperException
```

パラメーター
なし。

戻り値 シグニチャー。

スロー 演算子以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

getToken メソッド

目的 演算子 (演算子式ノード) のトークンを戻します。

構文

```
public java.lang.String getToken()
        throws WrapperException
```

パラメーター
なし。

戻り値 演算子のトークン。

スロー 演算子以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

getValue メソッド

目的 定数 (定数式ノード) の値を表す RequestConstant オブジェクトを戻します。

構文

```
public RequestConstant getValue()
        throws WrapperException
```

パラメーター
なし。

戻り値 RequestConstant インスタンスの定数の値。

スロー 定数以外の式ノードでメソッドが呼び出された場合、WrapperException オブジェクト。

関連資料:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『式の要求クラス』
- 142 ページの『RequestExpType クラス (Java)』

RequestExpType クラス (Java)

このセクションでは、RequestExpType クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RequestExpType クラスは、式ツリー内のノードのタイプを記述します。ノードは列参照、定数値、アンバウンド・パラメーター、または演算子のいずれかです。

RequestExpType クラスは、Java API のプランニング・クラスの 1 つです。

使用法 ラッパーは、RequestExpType オブジェクトを決してインスタンス化しません。フェデレーテッド・サーバーは、こうしたオブジェクトを作成し、照会プランニングの際にラッパーに渡します。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、RequestExpType クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 173. RequestExpType クラスのメソッド

メソッド	説明
getCodepage	文字データ・タイプのコード・ページを戻します。
getDataType	データ・タイプを戻します。
getForBitData	データがバイナリー形式かどうかを示す FOR BIT DATA フラグを戻します。
getMaximumLength	データ・タイプの最大長を戻します。
getName	データ・タイプの名前を戻します。
getNullIndicator	NULL 標識値を戻します。
getPrecision	数値データ・タイプの精度を戻します。
getScale	数値データ・タイプの位取りを戻します。

getCodepage 関数

目的 文字データ・タイプのコード・ページを戻します。

構文

```
public short getCodepage()
```

パラメーター

なし。

戻り値 コード・ページ。

スロー なし。

getDataType 関数

目的 データ・タイプを戻します。

構文

```
public short getDataType()
```

パラメーター

なし。

戻り値 データ・タイプ。

スロー なし。

getForBitData 関数

目的 データがバイナリー形式かどうかを示す FOR BIT DATA フラグを戻します。

構文

```
public boolean getForBitData()
```

パラメーター
なし。

戻り値 FOR BIT DATA フラグ。

スロー なし。

getMaximumLength 関数

目的 データ・タイプの最大長を戻します。

構文

```
public int getMaximumLength()
```

パラメーター
なし。

戻り値 データ・タイプの最大長。

スロー なし。

getName 関数

目的 データ・タイプの名前を戻します。

構文

```
public java.lang.String getName()
```

パラメーター
なし。

戻り値 名前。

スロー なし。

getNullIndicator 関数

目的 NULL 標識値を戻します。

構文

```
public short getNullIndicator()
```

パラメーター
なし。

戻り値 NULL 標識。

スロー なし。

getPrecision 関数

目的 数値データ・タイプの精度を戻します。

構文

```
public byte getPrecision()
```

パラメーター
なし。

戻り値 精度。

スロー なし。

getScale 関数

目的 数値データ・タイプの位取りを戻します。

構文

```
public byte getScale()
```

パラメーター
なし。

戻り値 位取り。

スロー なし。

関連資料:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『式タイプの要求クラス』
- 138 ページの『RequestExp クラス (Java)』

RequestConstant クラス (Java)

このセクションでは、RequestConstant クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RequestConstant クラスは、照会プランニングの際に照会式で使用される定数を記述します。 RequestExp.getValue() メソッドは、RequestConstant クラスのインスタンスを戻します。

public RequestConstant クラスは、Data クラスを拡張します。

RequestConstant クラスは、Java API のプランニング・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

定数

以下の定数は、Data クラスから継承され、RequestConstant クラスと共に使用できます。

表 174. RequestConstant クラスの定数

定数	定義	説明
BLOB	public static final short BLOB	BLOB データ・タイプを表す定数。
CHAR	public static final short CHAR	CHAR データ・タイプを表す定数。
CLOB	public static final short CLOB	CLOB データ・タイプを表す定数。
DATE	public static final short DATE	DATE データ・タイプを表す定数。
DECIMAL	public static final short DECIMAL	DECIMAL または NUMERIC データ・タイプを表す定数。
DOUBLE	public static final short DOUBLE	DOUBLE データ・タイプを表す定数。
FLOAT	public static final short FLOAT	FLOAT データ・タイプを表す定数。
INT	public static final short INT	INTEGER (INT) データ・タイプを表す定数。
LONG	public static final short LONG	LONG データ・タイプを表す定数。
NONE	public static final short NONE	不明なデータ・タイプを表す定数。
SHORT	public static final short SHORT	SHORT データ・タイプを表す定数。
TIME	public static final short TIME	TIME データ・タイプを表す定数。
TIMESTAMP	public static final short TIMESTAMP	TIMESTAMP データ・タイプを表す定数。
VARCHAR	public static final short VARCHAR	VARCHAR データ・タイプを表す定数。

メソッド

次の表は、RequestConstant クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 175. RequestConstant クラスのメソッド

メソッド	説明
getCodepage	文字データ・タイプのコード・ページを戻します。

表 175. *RequestConstant* クラスのメソッド (続き)

メソッド	説明
getData	内部形式のデータ・バッファーを戻します。
getDataType	データ・タイプを戻します。
getForBitData	FOR BIT DATA フラグを戻します。
getMaximumLength	データ・タイプの最大長を戻します。
getName	定義されている場合には、定数の名前を戻します。
getNullIndicator	NULL 標識を戻します。
getPrecision	数値データ・タイプの精度を戻します。
getScale	数値データ・タイプの位取りを戻します。
isDataNull	定数が NULL かどうかを示します。

getCodepage メソッド

目的 文字データ・タイプのコード・ページを戻します。

構文

```
public short getCodepage()
```

パラメーター
なし。

戻り値 コード・ページ。

スロー なし。

getData メソッド

目的 内部形式のデータ・バッファーを戻します。

構文

```
protected byte[] getData()
```

オーバーライド
Data クラスの getData メソッド。

パラメーター
なし。

戻り値 データ。

スロー なし。

getDataType メソッド

目的 データ・タイプを戻します。

構文

```
public short getDataType()
```

オーバーライド
Data クラスの getDataType メソッド。

パラメーター
なし。

戻り値 データ・タイプ。

スロー なし。

getForBitData メソッド

目的 FOR BIT DATA フラグを戻します。

構文

```
public boolean getForBitData()
```

オーバーライド
Data クラスの getForBitData メソッド。

パラメーター
なし。

戻り値 FOR BIT DATA フラグ。

スロー なし。

getMaximumLength メソッド

目的 データ・タイプの最大長を戻します。

構文

```
public int getMaximumLength()
```

パラメーター
なし。

戻り値 最大長。

スロー なし。

getName メソッド

目的 定義されている場合には、定数の名前を戻します。

構文

```
public java.lang.String getName()
```

パラメーター
なし。

戻り値 名前。

スロー なし。

getNullIndicator メソッド

目的 NULL 標識を戻します。

構文

```
public short getNullIndicator()
```

パラメーター
なし。

戻り値 NULL 標識。

スロー なし。

getPrecision メソッド

目的 数値データ・タイプの精度を戻します。

構文

```
public byte getPrecision()
```

パラメーター
なし。

戻り値 精度。

スロー なし。

getScale メソッド

目的 数値データ・タイプの位取りを戻します。

構文

```
public byte getScale()
```

パラメーター
なし。

戻り値 位取り。

スロー なし。

isDataNull メソッド

目的 定数が NULL かどうかを示します。

構文

```
public boolean isDataNull()
```

パラメーター
なし。

戻り値 定数が NULL の場合には、真の値。指定されていない場合には、値は偽です。

スロー なし。

継承されるメソッド

以下のメソッドは、Data クラスから継承され、RequestConstant クラスと共に使用できます。

- getBigDecimal
- getByte
- getDate
- getDouble
- getFloat
- getInt
- getLong
- getObject
- getShort
- getString
- getTime
- getTimestamp

関連資料:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『定数の要求クラス』
- 154 ページの『Data クラス (Java)』
- 138 ページの『RequestExp クラス (Java)』

Quantifier クラス (Java)

このセクションでは、Quantifier クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

Quantifier クラスは、Request の数量詞の情報をカプセル化します。

Quantifier クラスは、Java API のプランニング・クラスの 1 つです。

使用法 このクラスはフェデレーテッド・サーバーによりインスタンス化され、照会プランニングの際に Request オブジェクトの情報を持ちます。ラッパー固有の unfenced サーバー・サブクラスの planRequest メソッドは、数量詞情報を調べます。ラッパーがこの数量詞情報を処理できる場合には、planRequest メソッドは数量詞情報を Reply オブジェクトに追加します。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、Quantifier クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 176. Quantifier クラスのメソッド

メソッド	説明
getHandle	数量詞のハンドルを戻します。
getNickname	この数量詞が参照するニックネームを戻します。

getHandle メソッド

目的 数量詞のハンドルを戻します。ハンドルは、フェデレーテッド・サーバーの照会プランニング・コンポーネントが使用する整数値です。

構文

```
public int getHandle()
```

パラメーター
なし。

戻り値 数量詞ハンドル。

スロー なし。

getNickname メソッド

目的 この数量詞が参照するニックネームを戻します。

構文

```
public UnfencedGenericNickname getNickname()
```

パラメーター
なし。

戻り値 数量詞が参照するニックネーム。

スロー なし。

関連資料:

- 130 ページの『Reply クラス (Java)』
- 129 ページの『Request クラス (Java)』
- 98 ページの『UnfencedGenericNickname クラス (Java)』

PredicateList クラス (Java)

このセクションでは、PredicateList クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

PredicateList クラスは述部の 2 つのリストを記述して、選択性要因を推定します。このクラスのインスタンスは、選択性推定関数 UnfencedGenericServer.getSelectivity に対する入力として使用されます。 2 つの述部リストは次のとおりです。

- 選択性を請求する述部のリスト (P)
- 以前に適用された述部のリスト (AP)

結果の選択性は、条件付き選択性 (P/AP) です。無条件の選択性を必要とする場合には、AP を NULL にすることができます。インプリメンテーションがこのフレームワークをサポートできない場合には、インプリメンテーションは AP を無視するように設定できます。この述部のリストは、Request クラスの述部のリストに類似していて、同様のメソッドで取り扱われます。

PredicateList クラスは、Java API のプランニング・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、PredicateList クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 177. PredicateList クラスのメソッド

メソッド	説明
getAppliedPredicate	指定された位置の適用済み述部を記述する RequestExp オブジェクトを戻します。
getNumberOfAppliedPredicates	適用済み述部の数を戻します。
getNumberOfPredicates	リスト内の述部の数を戻します。
getPredicate	指定された位置の述部を記述する RequestExp オブジェクトを戻します。

getAppliedPredicate メソッド

目的 指定された位置の適用済み述部を記述する RequestExp オブジェクトを戻します。

構文

```
public RequestExp getAppliedPredicate(int position)
```

パラメーター

表 178. getAppliedPredicate メソッドのパラメーター

名前	説明
position	述部の位置。最初の述部の位置は 1 です。

戻り値 適用済み述部の式。

スロー なし。

getNumberOfAppliedPredicates メソッド

目的 適用済み述部の数を戻します。

構文

```
public int getNumberOfAppliedPredicates()
```

パラメーター

なし。

戻り値 適用済み述部の数。

スロー なし。

getNumberOfPredicates メソッド

目的 リスト内の述部の数を戻します。

構文

```
public int getNumberOfPredicates()
```

パラメーター

なし。

戻り値 述部の数。

スロー なし。

getPredicate メソッド

目的 指定された位置の述部を記述する RequestExp オブジェクトを戻します。

構文

```
public RequestExp getPredicate(int position)
```

パラメーター

表 179. getPredicate メソッドのパラメーター

名前	説明
position	述部の位置。最初の述部の位置は 1 です。

戻り値 述部の式。

スロー なし。

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『述部リスト・クラス』

関連資料:

- 138 ページの『RequestExp クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

Java API のデータ・クラス

以下の表で、Java API の各データ・クラスについて説明します。

表 180. データ・クラス

クラス名	説明
Data	セル値、定数、およびパラメーター値を管理するクラスで、値を内部形式から Java 標準タイプおよびオブジェクトへと変換するメソッドを提供します。
RuntimeData	フェデレーテッド・サーバーとラッパー間で転送される各列値を処理するクラス。
RuntimeDataList	入力パラメーターのリストまたは結果セットの行を処理する RuntimeData オブジェクトのリストをカプセル化するクラス。
RuntimeDataDesc	フェデレーテッド・サーバーによって作成されるクラスで、フェデレーテッド・サーバーとラッパー間で転送される各列値を記述します。
RuntimeDataDescList	RuntimeDataDesc オブジェクトのリストをカプセル化するクラスで、結果セットのパラメーター値または行のリストを記述します。

関連資料:

- 177 ページの『RuntimeDataDesc クラス (Java)』
- 184 ページの『RuntimeDataDescList クラス (Java)』
- 176 ページの『RuntimeDataList クラス (Java)』
- 160 ページの『RuntimeData クラス (Java)』

Data クラス (Java)

このセクションでは、Data クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

Data クラスはベース・クラスで、セル値、定数、およびパラメーター値を管理します。 このクラスは、値を内部形式から標準 Java タイプおよびオブジェクトに変換するためのメソッドを提供します。

RequestConstant クラスと RuntimeData クラスは、Data クラスのサブクラスです。

Data クラスは、Java API のデータ・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

定数

以下の表には、Data クラスで使用できる、このオプションの有効な定数が記載されています。

表 181. Data クラスの定数

定数	定義	説明
BLOB	public static final short BLOB	BLOB データ・タイプを表す定数。
CHAR	public static final short CHAR	CHAR データ・タイプを表す定数。
CLOB	public static final short CLOB	CLOB データ・タイプを表す定数。
DATE	public static final short DATE	DATE データ・タイプを表す定数。
DECIMAL	public static final short DECIMAL	DECIMAL または NUMERIC データ・タイプを表す定数。
DOUBLE	public static final short DOUBLE	DOUBLE データ・タイプを表す定数。
FLOAT	public static final short FLOAT	FLOAT データ・タイプを表す定数。
INT	public static final short INT	INTEGER (INT) データ・タイプを表す定数。
LONG	public static final short LONG	LONG データ・タイプを表す定数。
NONE	public static final short NONE	不明なデータ・タイプを表す定数。
SHORT	public static final short SHORT	SHORT データ・タイプを表す定数。
TIME	public static final short TIME	TIME データ・タイプを表す定数。
TIMESTAMP	public static final short TIMESTAMP	TIMESTAMP データ・タイプを表す定数。
VARCHAR	public static final short VARCHAR	VARCHAR データ・タイプを表す定数。

メソッド

次の表は、Data クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 182. Data クラスのメソッド

メソッド	説明
getBigDecimal	BigDecimal インスタンスとしてデータを戻します。
getByte	バイト値としてデータを戻します。
getData	バイト配列としてデータを戻します。
getDataType	データ・タイプを戻します。
getDate	Date インスタンスとしてデータを戻します。

表 182. *Data* クラスのメソッド *(続き)*

メソッド	説明
getDouble	DOUBLE データ・タイプ値としてデータを戻します。
getFloat	FLOAT データ・タイプ値としてデータを戻します。
getForBitData	FOR BIT DATA フラグを戻します。
getInt	整数 (INT) データ・タイプ値としてデータを戻します。
getLong	LONG データ・タイプ値としてデータを戻します。
getObject	Object インスタンスとしてデータを戻します。
getShort	SHORT データ・タイプ値としてデータを戻します。
getString	ストリングとしてデータを戻します。
getTime	Time インスタンスとしてデータを戻します。
getTimestamp	Timestamp インスタンスとしてデータを戻します。

getBigDecimal メソッド

目的 BigDecimal インスタンスとしてデータを戻します。

構文

```
public java.math.BigDecimal getBigDecimal()
                                    throws WrapperException
```

パラメーター

なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getByte メソッド

目的 バイト値としてデータを戻します。

構文

```
public byte getByte()
            throws WrapperException
```

パラメーター

なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getData メソッド

目的 バイト配列としてデータを戻します。値は内部形式です。

構文

```
protected abstract byte[] getData()
```

パラメーター

なし。

戻り値 データ。

スロー なし。

getDataType メソッド

目的 データ・タイプを戻します。

構文

```
public abstract short getDataType()
```

パラメーター

なし。

戻り値 データ・タイプ。

スロー なし。

getDate メソッド

目的 Date インスタンスとしてデータを戻します。

構文

```
public java.sql.Date getDate()
                        throws WrapperException
```

パラメーター

なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getDouble メソッド

目的 DOUBLE データ・タイプ値としてデータを戻します。

構文

```
public double getDouble()
                 throws WrapperException
```

パラメーター

なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getFloat メソッド

目的 FLOAT データ・タイプ値としてデータを戻します。

構文

```
public float getFloat()
              throws WrapperException
```

パラメーター
なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getForBitData メソッド

目的 FOR BIT DATA フラグを戻します。バイナリー形式でデータを保管するかどうかを示すには、このメソッドを CHAR または LONG VARCHAR データ・タイプにのみ使用してください。

構文

```
public abstract boolean getForBitData()
```

パラメーター
なし。

戻り値 FOR BIT DATA フラグ値。

スロー なし。

getInt メソッド

目的 整数 (INT) データ・タイプ値としてデータを戻します。

構文

```
public int getInt()
           throws WrapperException
```

パラメーター
なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getLong メソッド

目的 LONG データ・タイプ値としてデータを戻します。

構文

```
public long getLong()
             throws WrapperException
```

パラメーター
なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getObject メソッド

目的 Object インスタンスとしてデータを戻します。

構文

```
public java.lang.Object getObject()
                              throws WrapperException
```

パラメーター
なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getShort メソッド

目的 SHORT データ・タイプ値としてデータを戻します。

構文

```
public short getShort()
               throws WrapperException
```

パラメーター
なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getString メソッド

目的 ストリングとしてデータを戻します。

構文

```
public java.lang.String getString()
                              throws WrapperException
```

パラメーター
なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getTime メソッド

目的 Time インスタンスとしてデータを戻します。

構文

```
public java.sql.Time getTime()
                     throws WrapperException
```

パラメーター

なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

getTimestamp メソッド

目的 Timestamp インスタンスとしてデータを戻します。

構文

```
public java.sql.Timestamp getTimestamp()
                                  throws WrapperException
```

パラメーター

なし。

戻り値 データ。

スロー データ・タイプに互換性がない場合には、WrapperException オブジェクト。

関連資料:

- 145 ページの『RequestConstant クラス (Java)』
- 160 ページの『RuntimeData クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

RuntimeData クラス (Java)

このセクションでは、RuntimeData クラスについて説明し、そのメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RuntimeData クラスは、フェデレーテッド・サーバーとラッパーの間で転送されるそれぞれの列値を表します。列値は、ラッパーからフェデレーテッド・サーバーに転送される結果行の一部になります。あるいは、フェデレーテッド・サーバーによりラッパーにサブミットされる照会において実行時パラメーターにバインドする値になります。

public RuntimeData クラスは Data クラスを拡張します。

RuntimeData クラスは、Java API のデータ・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

定数およびクラス・フィールド

以下のメソッド定数およびクラス・フィールドは、Data クラスから継承され、RuntimeData クラスと共に使用できます。

- BLOB
- CHAR
- CLOB
- DATE
- DECIMAL
- DOUBLE
- FLOAT
- INT
- LONG
- NONE
- SHORT
- TIME
- TIMESTAMP
- VARCHAR

メソッド

次の表に、RuntimeData クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 183. RuntimeData クラスのメソッド

メソッド	説明
checkFriendlyDivBy	NULL 標識の理由が 0 除算エラーであることを識別します。
checkFriendlyException	NULL 標識の理由が数値例外であることを識別します。
clearNullIndicator	データ値に対する NULL 標識をクリアします。
getActualLength	データ値の実際の長さを戻します。
getCodepage	文字データ・タイプ値のコード・ページを戻します。
getData	内部形式のデータ値を戻します。
getDataIndex	データ値の列番号を戻します。
getDataType	データ・タイプを戻します。
getForBitData	バイナリー・データを示す FOR BIT DATA フラグを戻します。
getInvariant	不変値を戻します。
getMaximumLength	データの最大長を戻します。
getName	データの名前を戻します。
getNullIndicator	データ値の NULL 標識を戻します。
getPrecision	数値データ・タイプ値の精度を戻します。

表 183. RuntimeData クラスのメソッド (続き)

メソッド	説明
getRemoteLength	データ値のリモートの長さを戻します。
getRemotePrecision	数値データ・タイプ値のリモート精度を戻します。
getRemoteScale	数値データ・タイプ値のリモートの位取りを戻します。
getRemoteType	データ値のリモート・タイプを戻します。
getScale	数値データ・タイプ値の位取りを戻します。
isDataNull	データ値が NULL かどうか示します。
isDataNullable	データ値が NULL 可能かどうか示します。
isSemanticNull	データ値がセマンティック NULL かどうか示します。
setActualLength	データ値の実際の長さを設定します。
setBigDecimal	データ値を BigDecimal インスタンスとして設定します。
setBinary	データ値をバイナリー値として設定します。
setByte	データ値をバイトとして設定します。
setDataNull	データ値を NULL としてマークします。
setDate	データ値を Date インスタンスとして設定します。
setDouble	データ値を double データ・タイプ値として設定します。
setFloat	データ値を float データ・タイプ値として設定します。
setFriendlyDivBy0	0 除算エラーが発生したため、値が NULL であることを示します。
setFriendlyException	数値例外のため、値が NULL であることを示します。
setInt	データ値を int データ・タイプ値として設定します。
setLong	データ値を long データ・タイプ値として設定します。
setNullIndicator	データ値に NULL 標識を設定します。
setObject	データ値を Object インスタンスとして設定します。
setPrecision	数値データ・タイプ値の精度を設定します。
setScale	数値データ・タイプ値の位取りを設定します。
setShort	データ値を short データ・タイプ値として設定します。
setString	データをストリング値として設定します。
setTime	データを Time インスタンス値として設定します。

表 183. RuntimeData クラスのメソッド (続き)

メソッド	説明
setTimestamp	データを Timestamp インスタンス値として設定します。

checkFriendlyDivBy メソッド

目的 NULL 標識の理由が 0 除算エラーであることを識別します。

構文

```
public boolean checkFriendlyDivBy0()
```

パラメーター

なし。

戻り値 0 除算エラーが発生した場合は、真の値。それ以外の場合は、値は偽です。

スロー なし。

checkFriendlyException メソッド

目的 NULL 標識の理由が数値例外であることを識別します。

構文

```
public boolean checkFriendlyException()
```

パラメーター

なし。

戻り値 NULL 標識の理由が数値例外である場合は真の値。それ以外の場合は、値は偽です。

スロー なし。

clearNullIndicator メソッド

目的 データ値に対する NULL 標識をクリアします。

構文

```
public void clearNullIndicator()
                          throws WrapperException
```

パラメーター

なし。

戻り値 なし。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

getActualLength メソッド

目的 データ値の実際の長さを戻します。

構文

```
public int getActualLength()
```

パラメーター
なし。

戻り値 なし。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

getData メソッド

目的 フェデレーテッド・サーバーの内部形式のデータ値を戻します。

構文

```
protected byte[] getData()
```

オーバーライド
Data クラスの getData メソッド。

パラメーター
なし。

戻り値 フェデレーテッド・サーバーの内部形式のデータ。

スロー なし。

getDataIndex メソッド

目的 データ値の列番号を戻します。

構文

```
public int getDataIndex()
```

パラメーター
なし。

戻り値 列番号。

スロー なし。

getDataType メソッド

目的 データ・タイプを戻します。

構文

```
public short getDataType()
```

オーバーライド
Data クラスの getDataType メソッド。

パラメーター
なし。

戻り値 データ・タイプ。

スロー なし。

getForBitData メソッド

目的 バイナリー・データを示す FOR BIT DATA フラグを戻します。

構文

```
public short getDataType()
```

オーバーライド
Data クラスの getForBitData メソッド。

パラメーター
なし。

戻り値 FOR BIT DATA データ・フラグ。

スロー なし。

getInvariant メソッド

目的 不変値を戻します。 RemoteQuery.reopen(short) メソッドのアクション・パラメーターに対する変更がラッパーに通知される場合を除いて、不変パラメーターの値は変更されません。

構文

```
public byte getInvariant()
```

パラメーター
なし。

戻り値 不変値。

スロー なし。

getMaximumLength メソッド

目的 データの最大長を戻します。

構文

```
public int getMaximumLength()
```

パラメーター
なし。

戻り値 最大長。

スロー なし。

getName メソッド

目的 データの名前を戻します。

構文

```
public java.lang.String getName()
```

パラメーター
なし。

戻り値 名前。

スロー なし。

getNullIndicator メソッド

目的 データ値の NULL 標識を戻します。

構文

```
public short getNullIndicator()
```

パラメーター

なし。

戻り値 NULL 標識。

スロー なし。

getPrecision メソッド

目的 数値データ・タイプ値の精度を戻します。

構文

```
public byte getPrecision()
```

パラメーター

なし。

戻り値 精度。

スロー なし。

getRemoteLength メソッド

目的 データ値のリモートの長さを戻します。

構文

```
public int getRemoteLength()
```

パラメーター

なし。

戻り値 リモートの長さ。

スロー なし。

getRemotePrecision メソッド

目的 数値データ・タイプ値のリモート精度を戻します。

構文

```
public byte getRemotePrecision()
```

パラメーター

なし。

戻り値 リモートの精度。

スロー なし。

getRemoteScale メソッド

目的 数値データ・タイプ値のリモートの位取りを戻します。

構文

```
public byte getRemoteScale()
```

パラメーター

なし。

戻り値 リモートのスケール。

スロー なし。

getRemoteType メソッド

目的 データ値のリモート・タイプを戻します。

構文

```
public short getRemoteType()
```

パラメーター

なし。

戻り値 リモート・タイプ。

スロー なし。

getScale メソッド

目的 数値データ・タイプ値の位取りを戻します。

構文

```
public byte getScale()
```

パラメーター

なし。

戻り値 位取り。

スロー なし。

isDataNull メソッド

目的 データ値が NULL かどうか示します。

構文

```
public boolean isDataNull()
```

パラメーター

なし。

戻り値 データが NULL の場合は、真の値。それ以外の場合は、値は偽です。

スロー なし。

isDataNullable メソッド

目的 データ値が NULL 可能かどうか示します。

構文

```
public boolean isDataNullable()
```

パラメーター

なし。

戻り値 データが NULL 可能の場合は、真の値。それ以外の場合は、値は偽です。

スロー なし。

isSemanticNull メソッド

目的 データ値がセマンティック NULL かどうか示します。

構文

```
public boolean isSemanticNull()
```

パラメーター

なし。

戻り値 データがセマンティック NULL の場合は、真の値。それ以外の場合は、値は偽です。

スロー なし。

setActualLength メソッド

目的 データ値の実際の長さを設定します。

構文

```
public void setActualLength(int length)
```

パラメーター

表 184. setActualLength メソッドのパラメーター

名前	説明
length	実際の長さ。

戻り値 なし。

スロー なし。

setBigDecimal メソッド

目的 データ値を BigDecimal インスタンスとして設定します。

構文

```
public void setBigDecimal(java.math.BigDecimal value)
                  throws WrapperException
```

パラメーター

表 185. setBigDecimal メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setBinary メソッド

目的 データ値をバイナリー値として設定します。

構文

```
public void setBinary(byte[] value)
                throws WrapperException
```

パラメーター

表 186. setBinary メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setByte メソッド

目的 データ値をバイトとして設定します。

構文

```
public void setByte(byte value)
             throws WrapperException
```

パラメーター

表 187. setByte メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setDataNull メソッド

目的 データ値を NULL としてマークします。

構文

```
public void setDataNull()
                throws WrapperException
```

パラメーター

なし。

戻り値 なし。

スロー データが NULL にできない場合には、WrapperException オブジェクト。

setDate メソッド

目的 データ値を Date インスタンスとして設定します。

構文

```
public void setDate(java.sql.Date value)
             throws WrapperException
```

パラメーター

表 188. setDate メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setDouble メソッド

目的 データ値を double データ・タイプ値として設定します。

構文

```
public void setDouble(double value)
               throws WrapperException
```

パラメーター

表 189. setDouble メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setFloat メソッド

目的 データ値を float データ・タイプ値として設定します。

構文

```
public void setFloat(float value)
             throws WrapperException
```

パラメーター

表 190. setFloat メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setFriendlyDivBy0 メソッド

目的 0 除算エラーが発生したため、値が NULL であることを示します。

使用法 ラッパーが、setFriendlyDivBy0 メソッドを呼び出す前に setDataNull() メソッドを呼び出すかどうかを検証します。

構文

```
public void setFriendlyDivBy0()
                      throws WrapperException
```

パラメーター

なし。

戻り値 なし。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

setFriendlyException メソッド

目的 数値例外のため、値が NULL であることを示します。

使用法 ラッパーが、setFriendlyException メソッドを呼び出す前に setDataNull() メソッドを呼び出すかどうかを検証します。

構文

```
public void setFriendlyException()
                          throws WrapperException
```

パラメーター

なし。

戻り値 なし。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

setInt メソッド

目的 データ値を int データ・タイプ値として設定します。

構文

```
public void setInt(int value)
            throws WrapperException
```

パラメーター

表 191. setInt メソッドのパラメーター

名前	説明
value	データ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setLong メソッド

目的 データ値を long データ・タイプ値として設定します。

構文

```
public void setLong(long value)
             throws WrapperException
```

パラメーター

表 192. setLong メソッドのパラメーター

名前	説明
value	設定するデータ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setNullIndicator メソッド

目的 データ値に NULL 標識を設定します。

構文

```
public void setNullIndicator(short indicator)
```

パラメーター

表 193. setNullIndicator メソッドのパラメーター

名前	説明
indicator	NULL 標識。

戻り値 なし。

スロー なし。

setObject メソッド

目的 データ値を Object インスタンスとして設定します。

構文

```
public void setObject(java.lang.Object value)
              throws WrapperException
```

パラメーター

表 194. setObject メソッドのパラメーター

名前	説明
value	設定するデータ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setPrecision メソッド

目的 数値データ・タイプ値の精度を設定します。

構文

```
public void setPrecision(byte precision)
```

パラメーター

表 195. setPrecision メソッドのパラメーター

名前	説明
precision	設定する精度。

戻り値 なし。

スロー なし。

setScale メソッド

目的 数値データ・タイプ値の位取りを設定します。

構文

```
public void setScale(byte scale)
```

パラメーター

表 196. setScale メソッドのパラメーター

名前	説明
scale	設定する位取り。

戻り値 なし。

スロー なし。

setShort メソッド

目的 データ値を short データ・タイプ値として設定します。

構文

```
public void setShort(short value)
             throws WrapperException
```

パラメーター

表 197. setShort メソッドのパラメーター

名前	説明
value	設定するデータ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setString メソッド

目的 データ値をストリングとして設定します。

構文

```
public void setString(java.lang.String value)
              throws WrapperException
```

パラメーター

表 198. setString メソッドのパラメーター

名前	説明
value	設定するデータ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setTime メソッド

目的 データ値を Time インスタンスとして設定します。

構文

```
public void setTime(java.sql.Time value)
             throws WrapperException
```

パラメーター

表 199. setTime メソッドのパラメーター

名前	説明
value	設定するデータ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

setTimestamp メソッド

目的 データ値を Timestamp インスタンスとして設定します。

構文

```
public void setTimestamp(java.sql.Timestamp value)
                  throws WrapperException
```

パラメーター

表 200. setTimestamp メソッドのパラメーター

名前	説明
value	設定するデータ値。

戻り値 なし。

スロー データがデータ・タイプと非互換の場合には、WrapperException オブジェクト。

継承されるメソッド

以下のメソッドは、Data クラスから継承され、RuntimeData クラスとともに使用できます。

- getBigDecimal
- getByte
- getDate
- getDouble
- getFloat
- getInt
- getLong
- getObject
- getShort
- getString
- getTime
- getTimestamp

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ランタイム・データ・クラス』

関連資料:

- 154 ページの『Data クラス (Java)』
- 114 ページの『RemoteQuery クラス (Java)』

• 186 ページの『WrapperException クラス (Java)』

RuntimeDataList クラス (Java)

このセクションでは、RuntimeDataList クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RuntimeDataList クラスは、結果セット内の行または入力パラメーターのリストのいずれかを表す RuntimeData オブジェクトのリストをカプセル化します。

RuntimeDataList クラスは、Java API のデータ・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、RuntimeDataList クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 201. RuntimeDataList クラスのメソッド

メソッド	説明
getNumberOfValues	リスト内の値の数を戻します。
getValue	リスト内の指定された位置の値を戻します。

getNumberOfValues メソッド

目的 リスト内の値の数を戻します。

構文

```
public int getNumberOfValues()
```

パラメーター

なし。

戻り値 値の数。

スロー なし。

getValue メソッド

目的 リスト内の指定された位置の値を戻します。

構文

```
public RuntimeData getValue(int position)
```

パラメーター

表 202. getValue メソッドのパラメーター

名前	説明
position	戻される値の位置。最初の値の位置は、リスト内でゼロです。

戻り値 データ値オブジェクト。

スロー なし。

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ランタイム・データ・クラス』

関連資料:

- 177 ページの『RuntimeDataDesc クラス (Java)』
- 160 ページの『RuntimeData クラス (Java)』

RuntimeDataDesc クラス (Java)

このセクションでは、RuntimeDataDesc クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

フェデレーテッド・サーバーは RuntimeDataDesc クラスを作成して、フェデレーテッド・サーバーとラッパー間で転送される各列値を記述します。列値は、以下のいずれかの項目になります。

- ラッパーからフェデレーテッド・サーバーに転送される結果行の一部。
- 照会内でフェデレーテッド・サーバーがラッパーにサブミットする、実行時パラメーターにバインドされる値。

RuntimeDataDesc クラスは、Java API のデータ・クラスの 1 つです。

使用法 RemoteQuery と RemotePassthru オブジェクトのいずれの場合も、フェデレーテッド・サーバーは、パラメーター値を表す入力データ・リスト内の各 RuntimeData オブジェクトに RuntimeDataDesc インスタンスを作成します。フェデレーテッド・サーバーは列説明を提供します。 RemoteQuery オブジェクトの場合、フェデレーテッド・サーバーは、結果行の値を表す出力データ・リスト内の各 RuntimeData オブジェクトに RuntimeDataDesc インスタンスも作成します。 フェデレーテッド・サーバーは、それらの列説明も提供します。 ラッパー・ライターは、RuntimeDataDesc クラスのインスタンスを作成し、 RemotePassthru.describe メソッド内のパススルー・セッションの結果セットを記述します。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表では、RuntimeDataDesc クラスのコンストラクターおよびメソッドについて説明されています。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 203. RuntimeDataDesc クラスのコンストラクター

コンストラクター	説明
RuntimeDataDesc	指定された属性 (タイプ ID、最大長、コード・ページ、および NULL 標識) を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。
RuntimeDataDesc	指定された属性 (タイプ ID、最大長、コード・ページ、NULL 標識、および精度) を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。
RuntimeDataDesc	指定された属性 (タイプ ID、最大長、コード・ページ、NULL 標識、精度、および位取り) を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。
RuntimeDataDesc	指定された属性 (タイプ ID、最大長、コード・ページ、NULL 標識、精度、位取り、およびデータ列名) を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。
RuntimeDataDesc	指定された属性 (タイプ ID、最大長、コード・ページ、NULL 標識、精度、位取り、データ列名、およびリモート・データのタイプ) を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。

表 204. RuntimeDataDesc クラスのメソッド

メソッド	説明
getCodepage	文字データ・タイプ値のコード・ページを戻します。
getDataType	データ・タイプを戻します。
getForBitData	バイナリー・データを示す FOR BIT DATA フラグを戻します。
getMaximumLength	データの最大長を戻します。
getName	データの名前を戻します。
getNullIndicator	データ値の NULL 標識を戻します。
getPrecision	数値データ・タイプ値の精度を戻します。
getRemoteType	データのリモート・タイプを戻します。
getScale	数値データ・タイプ値の位取りを戻します。

RuntimeDataDesc コンストラクター

目的 指定された属性を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。

構文

```
public RuntimeDataDesc(short type,
                       int maxLength,
                       short codepage,
                       short nullIndicator)
                throws WrapperException
```

パラメーター

表 205. RuntimeDataDesc コンストラクターのパラメーター

名前	説明
type	データのタイプ ID。
maxLength	データの最大長。
codepage	データを表現するコード・ページ。このパラメーターは、文字データ・タイプのみに有効です。
nullIndicator	NULL 値の標識。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

RuntimeDataDesc コンストラクター

目的 指定された属性を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。

構文

```
public RuntimeDataDesc(short type,
                       int maxLength,
                       short codepage,
                       short nullIndicator,
                       byte precision)
                throws WrapperException
```

パラメーター

表 206. RuntimeDataDesc コンストラクターのパラメーター

名前	説明
type	データのタイプ ID。
maxLength	データの最大長。
codepage	データを表現するコード・ページ。このパラメーターは、文字データ・タイプのみに有効です。
nullIndicator	NULL 値の標識。
precision	データの精度。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

RuntimeDataDesc コンストラクター

目的 指定された属性を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。

構文

```
public RuntimeDataDesc(short type,
                       int maxLength,
                       short codepage,
                       short nullIndicator,
                       byte precision,
                       byte scale)
                 throws WrapperException
```

パラメーター

表 207. RuntimeDataDesc コンストラクターのパラメーター

名前	説明
type	データのタイプ ID。
maxLength	データの最大長。
codepage	データを表現するコード・ページ。このパラメーターは、文字データ・タイプのみに有効です。
nullIndicator	NULL 値の標識。
precision	データの精度。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。
scale	データの位取り。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

RuntimeDataDesc コンストラクター

目的 指定された属性を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。

構文

```
public RuntimeDataDesc(short type,
                       int maxLength,
                       short codepage,
                       short nullIndicator,
                       byte precision,
                       byte scale,
                       java.lang.String name)
                 throws WrapperException
```

パラメーター

表 208. RuntimeDataDesc コンストラクターのパラメーター

名前	説明
type	データのタイプ ID。
maxLength	データの最大長。
codepage	データを表現するコード・ページ。このパラメーターは、文字データ・タイプのみに有効です。
nullIndicator	NULL 値の標識。
precision	データの精度。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。
scale	データの位取り。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。
name	データ列の名前。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

RuntimeDataDesc コンストラクター

目的 指定された属性を持ち、結果セットの列を記述する新規 RuntimeDataDesc オブジェクトを構成します。

構文

```
public RuntimeDataDesc(short type,
                       int maxLength,
                       short codepage,
                       short nullIndicator,
                       byte precision,
                       byte scale,
                       java.lang.String name,
                       short remoteType)
                 throws WrapperException
```

パラメーター

表 209. RuntimeDataDesc コンストラクターのパラメーター

名前	説明
type	データのタイプ ID。
maxLength	データの最大長。
codepage	データを表現するコード・ページ。このパラメーターは、文字データ・タイプのみに有効です。
nullIndicator	NULL 値の標識。
precision	データの精度。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。

表 209. RuntimeDataDesc コンストラクターのパラメーター (続き)

名前	説明
scale	データの位取り。このパラメーターは、数値データ・タイプおよび 10 進数データ・タイプのみに有効です。
name	データ列の名前。
remoteType	リモート・データ・ソースでのデータ・タイプ。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

getCodepage メソッド

目的 文字データ・タイプ値のコード・ページを戻します。

構文

```
public short getCodepage()
```

パラメーター
なし。

戻り値 コード・ページ。

スロー なし。

getDataType メソッド

目的 データ・タイプを戻します。

構文

```
public short getDataType()
```

パラメーター
なし。

戻り値 データ・タイプ。

スロー なし。

getForBitData メソッド

目的 バイナリー・データを示す FOR BIT DATA フラグを戻します。

構文

```
public boolean getForBitData()
```

オーバーライド
Data クラスの getForBitData メソッド。

パラメーター
なし。

戻り値 FOR BIT DATA データ・フラグ。

スロー なし。

getMaximumLength メソッド

目的 データの最大長を戻します。

構文

```
public int getMaximumLength()
```

パラメーター
なし。

戻り値 最大長。

スロー なし。

getName メソッド

目的 データの名前を戻します。

構文

```
public java.lang.String getName()
                          throws WrapperException
```

パラメーター
なし。

戻り値 名前。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

getNullIndicator メソッド

目的 データ値の NULL 標識を戻します。

構文

```
public short getNullIndicator()
```

パラメーター
なし。

戻り値 NULL 標識。

スロー なし。

getPrecision メソッド

目的 数値データ・タイプ値の精度を戻します。

構文

```
public byte getPrecision()
```

パラメーター
なし。

戻り値 精度。

スロー なし。

getRemoteType メソッド

目的 データ値のリモート・タイプを戻します。

構文

```
public short getRemoteType()
```

パラメーター
なし。

戻り値 リモート・タイプ。

スロー なし。

getScale メソッド

目的 数値データ・タイプ値の位取りを戻します。

構文

```
public byte getScale()
```

パラメーター
なし。

戻り値 位取り。

スロー なし。

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ランタイム・データ記述クラス』

関連資料:

- 110 ページの『RemotePassthru クラス (Java)』
- 114 ページの『RemoteQuery クラス (Java)』
- 160 ページの『RuntimeData クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

RuntimeDataDescList クラス (Java)

このセクションでは、RuntimeDataDescList クラスについて説明し、このメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

RuntimeDataDescList クラスは、結果セット内の行または入力パラメーターのリストを記述する RuntimeDataDesc オブジェクトのリストをカプセル化します。

RuntimeDataDesc クラスは、Java API のデータ・クラスの 1 つです。

パッケージ

com.ibm.db2.wrapper

メソッド

次の表は、RuntimeDataDescList クラスのメソッドについて説明しています。表の後に、メソッドの詳細が記載されています。

表 210. RuntimeDataDescList クラスのメソッド

メソッド	説明
getNumberOfValues	リスト内のデータ記述子の数を戻します。
getValue	リスト内の指定された位置のデータ記述子を戻します。
setNumberOfValues	リスト内のデータ記述子の数を設定します。
setValue	リスト内の指定された位置のデータ記述子を設定します。

getNumberOfValues メソッド

目的 リスト内のデータ記述子の数を戻します。

構文

```
public int getNumberOfValues()
```

パラメーター

なし。

戻り値 データ記述子の数。

スロー なし。

getValue メソッド

目的 リスト内の指定された位置のデータ記述子を戻します。

構文

```
public RuntimeDataDesc getValue(int position)
```

パラメーター

表 211. getValue メソッドのパラメーター

名前	説明
position	戻されるデータ記述子の位置。最初のデータ記述子の位置は、リスト内でゼロです。

戻り値 データ記述子。

スロー なし。

setNumberOfValues メソッド

目的 リスト内のデータ記述子の数を設定します。

構文

```
public void setNumberOfValues(int newNumber)
                        throws WrapperException
```

パラメーター

表 212. setNumberOfValues メソッドのパラメーター

名前	説明
newNumber	データ記述子の数。

戻り値 なし。

スロー メソッドが失敗した場合には、WrapperException オブジェクト。

setValue メソッド

目的 リスト内の指定された位置のデータ記述子を設定します。

構文

```
public void setValue(RuntimeDataDesc runtimeDataDesc,
                     int position)
              throws WrapperException
```

パラメーター

表 213. setValue メソッドのパラメーター

名前	説明
runtimeDataDesc	リストに保管されるデータ記述子。
position	データ記述子がある位置。最初のデータ記述子の位置は、リスト内でゼロです。

戻り値 なし。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

関連資料:

- 177 ページの『RuntimeDataDesc クラス (Java)』
- 186 ページの『WrapperException クラス (Java)』

WrapperException クラス (Java)

このセクションでは、WrapperException クラスについて説明し、そのコンストラクターおよびメソッドの詳細について解説します。

概要

WrapperException クラスは、例外をレポートするために Java API により使用される例外クラスです。

public WrapperException クラスは java.lang.Exception クラスを拡張します。

使用法 このクラスはラッパーによりインスタンス化され、ストリング・メッセージ、あるいは SQL エラー・コード、呼び出し元関数名、および DB2 Information Integrator エラー・メッセージにマップされるトークンが含まれる例外をレポートします。

パッケージ

com.ibm.db2.wrapper

コンストラクターおよびメソッド

以下の表で、WrapperException クラスのコンストラクターおよびメソッドについて説明します。表の後に、コンストラクターとメソッドの詳細が記載されています。

表 214. WrapperException クラスのコンストラクター

コンストラクター	説明
WrapperException	DB2 Information Integrator エラーを、コード、呼び出し元関数名、およびトークン別にレポートする例外オブジェクトを構成します。
WrapperException	指定されたメッセージを使用して例外オブジェクトを構成します。

表 215. WrapperException クラスのメソッド

メソッド	説明
getAndResetErrorCode	SQL エラー・コードを戻してリセットし、DB2 Information Integrator エラーをレポートします。
getAndResetFunctionName	DB2 Information Integrator エラーをレポートする呼び出し元関数名を戻してリセットします。
getAndResetTokens	使用する置換トークンを戻してリセットし、DB2 Information Integrator エラーをレポートします。
getErrorCode	SQL エラー・コードを戻して、DB2 Information Integrator エラーをレポートします。
getFunctionName	DB2 Information Integrator エラーをレポートする呼び出し元関数名を戻します。
getMessage	この例外オブジェクトのエラー・メッセージ・ストリングを戻します。
getStackTrace	例外のスタック・トレースをストリングに保管します。

表 215. *WrapperException* クラスのメソッド (続き)

メソッド	説明
getTokens	DB2 Information Integrator エラーをレポートする置換トークンを戻します。
setErrorCode	DB2 Information Integrator エラーをレポートする SQL エラー・コードを設定します。
setFunctionName	DB2 Information Integrator エラーをレポートする呼び出し元関数名を設定します。
setTokens	DB2 Information Integrator エラーをレポートする置換トークンを設定します。

WrapperException コンストラクター

目的 DB2 Information Integrator エラーを、SQL コード、呼び出し元関数名、およびトークン別にレポートする例外オブジェクトを構成します。それぞれの有効な SQL コードはエラー・メッセージを識別します。エラー・メッセージには、メッセージがユーザーに報告される前に指定されたトークンに置き換えられる、プレースホルダーが含まれる可能性があります。

構文

```
public WrapperException(int errorCode,
                        java.lang.String functionName,
                        java.lang.String[] tokens)
```

パラメーター

表 216. *WrapperException* コンストラクターのパラメーター

名前	説明
errorCode	レポートされるエラーの事前定義された SQL コード。
functionName	エラーをレポートする関数の名前。ストリング値の最大長は 5 文字です。クライアント・プログラムは、SQLCA の SQLERRP フィールドを介してこのストリング値にアクセスできます。ストリング値は、SQL の接頭部付きの大文字で表示されます。
tokens	メッセージの置換トークン。

WrapperException コンストラクター

目的 指定されたメッセージを使用して例外オブジェクトを構成します。エラー・コードが、スローされた例外で指定されていない場合、SQL0901 エラーがレポートされ、指定された例外メッセージが SQL0901 エラー・メッセージ・プレースホルダーに置き換えられます。

構文

```
public WrapperException(java.lang.String message)
```

パラメーター

表 217. WrapperException コンストラクターのパラメーター

名前	説明
message	例外について説明するメッセージ。

getAndResetErrorCode メソッド

目的 SQL エラー・コードを戻してリセットし、DB2 Information Integrator エラーをレポートします。 DB2 Information Integrator エラーはそれぞれ、SQL エラー・コードで識別されます。

構文

```
public final int getAndResetErrorCode()
```

パラメーター
なし。

戻り値 レポートされるエラーの事前定義された SQL コード。

スロー なし。

getAndResetFunctionName メソッド

目的 DB2 Information Integrator エラーをレポートする呼び出し元関数名を戻してリセットします。呼び出し元関数名は、SQLCA 構造の SQLERRP フィールド内の DB2 エラーでレポートされます。

構文

```
public final java.lang.String getAndResetFunctionName()
```

パラメーター
なし。

戻り値 エラーをレポートする関数の名前。ストリング値の最大長は 5 文字です。クライアント・プログラムは、SQLCA の SQLERRP フィールドを介してこのストリング値にアクセスできます。ストリング値は、SQL の接頭部付きの大文字で表示されます。

スロー なし。

getAndResetTokens メソッド

目的 使用する置換トークンを戻してリセットし、DB2 Information Integrator エラーをレポートします。置換トークンは、DB2 Information Integrator エラー・メッセージ内のプレースホルダーを置き換えます。

構文

```
public final java.lang.String[] getAndResetTokens()
```

パラメーター
なし。

戻り値 DB2 Information Integrator エラーのレポート時に、エラー・メッセージで使用される置換トークン。

スロー なし。

getErrorCode メソッド

目的 SQL エラー・コードを戻して、DB2 Information Integrator エラーをレポートします。 DB2 Information Integrator エラーはそれぞれ、SQL エラー・コードで識別されます。

構文

```
public final int getErrorCode()
```

パラメーター
なし。

戻り値 レポートされるエラーの事前定義された SQL コード。

スロー なし。

getFunctionName メソッド

目的 DB2 Information Integrator エラーをレポートする呼び出し元関数名を戻します。呼び出し元関数名は、SQLCA 構造の SQLERRP フィールド内の DB2 Information Integrator エラーと共にレポートされます。

構文

```
public final java.lang.String getFunctionName()
```

パラメーター
なし。

戻り値 エラーをレポートする関数の名前。ストリング値の最大長は 5 文字です。クライアント・プログラムは、SQLCA の SQLERRP フィールドを介してこのストリング値にアクセスできます。ストリング値は、SQL の接頭部付きの大文字で表示されます。

スロー なし。

getMessage メソッド

目的 この例外オブジェクトのエラー・メッセージ・ストリングを戻します。

構文

```
public java.lang.String getMessage()
```

パラメーター
なし。

戻り値 WraperException オブジェクトがエラー・メッセージ・メッセージで作成された場合には、この WrapperException オブジェクトのエラー・メッセー

ジ・ストリング。WraperException オブジェクトが SQL エラー・コードで作成された場合には、SQL エラー・コード、呼び出し元関数名、およびトークンのセットを示すストリング。

スロー なし。

getStackTrace メソッド

目的 例外のスタック・トレースをストリングに保管します。

構文

```
public static java.lang.String getStackTrace(java.lang.Throwable throwable)
```

パラメーター

表 218. getStackTrace メソッドのパラメーター

名前	説明
throwable	スタック・トレースから取り出したスロー可能なオブジェクト。

戻り値 スタック・トレース情報が含まれるストリング。

スロー なし。

getTokens メソッド

目的 DB2 Information Integrator エラーのレポートに使用する置換トークンを戻します。置換トークンは、DB2 Information Integrator エラー・メッセージ内のプレースホルダーを置き換えます。

構文

```
public final java.lang.String[] getTokens()
```

パラメーター
なし。

戻り値 DB2 Information Integrator エラーのレポート時に、エラー・メッセージで使用される置換トークン。

スロー なし。

setErrorCode メソッド

目的 DB2 Information Integrator エラーをレポートする SQL エラー・コードを設定します。 DB2 Information Integrator エラーはそれぞれ、SQL エラー・コードで識別されます。

構文

```
public final void setErrorCode(int errorCode)
```

パラメーター

表 219. setErrorCode メソッドのパラメーター

名前	説明
errorCode	レポートされるエラーの事前定義された SQL コード。

戻り値 なし。

スロー なし。

setFunctionName メソッド

目的 DB2 Information Integrator エラーをレポートする呼び出し元関数名を設定します。呼び出し元関数名は、SQLCA 構造の SQLERRP フィールド内の DB2 Information Integrator エラーと共にレポートされます。

構文

```
public final void setFunctionName(java.lang.String functionName)
```

パラメーター

表 220. setFunctionName メソッドのパラメーター

名前	説明
functionName	エラーをレポートする関数の名前。ストリング値の最大長は 5 文字です。クライアント・プログラムは、SQLCA の SQLERRP フィールドを介してこのストリング値にアクセスできます。ストリング値は、SQL の接頭部付きの大文字で表示されます。

戻り値 なし。

スロー なし。

setTokens メソッド

目的 DB2 Information Integrator エラーのレポートに使用する置換トークンを設定します。置換トークンは、DB2 Information Integrator エラー・メッセージ内のプレースホルダーを置き換えます。

構文

```
public final void setTokens(java.lang.String[] tokens)
```

パラメーター

表 221. setTokens メソッドのパラメーター

名前	説明
tokens	DB2 Information Integrator エラーのレポート時に、エラー・メッセージで使用される置換トークン。

戻り値 なし。

スロー なし。

関連資料:

• 193 ページの『WrapperUtilities クラス (Java)』

WrapperUtilities クラス (Java)

このセクションでは、WrapperUtilities クラスについて説明し、そのメソッドの詳細について解説します。

このクラスには、コンストラクターは含まれません。

概要

WrapperUtilities クラスは、いくつかの静的ユーティリティー関数のコンテナーです。

WrapperUtilities クラスは Java API のユーティリティー・クラスです。

使用法 WrapperUtilities クラスはインスタンス化またはサブクラス化しないでください。

パッケージ

com.ibm.db2.wrapper

定数

以下の表では、WrapperUtilities クラスで使用できるオプションの有効な定数について説明します。

表 222. WrapperUtilities クラスの定数

定数	例	説明
EXT_WRAPPERS_ TRACE_ COMPONENT	public static final int EXT_WRAPPERS_TRACE_ COMPONENT	IBM により提供されていないラッパーのトレース・コンポーネント ID。

メソッド

次の表に、WrapperUtilities クラスのメソッドを示します。表の後に、メソッドの詳細が記載されています。

表 223. WrapperUtilities クラスのメソッド

メソッド	説明
getAppCBStatus	アプリケーション制御ブロックの状況を戻します。
getAuthid	現行データベースの許可 ID を戻します。
getCodepage	現行データベースのコード・ページを戻します。

表 223. *WrapperUtilities* クラスのメソッド (続き)

メソッド	説明
getDB2InstallPath	DB2 Universal Database のインストール・パスを戻します。
getDB2InstancePath	DB2 Universal Database のインスタンス・パスを戻します。
getDB2Release	このラッパーが現在動作している DB2 Universal Database リリースおよびフィックスパックを戻します。
getDoubleByteDBCodepage	データベースの 2 バイト・コード・ページを戻します。
getIsolationLevel	現行データベースの分離レベルを戻します。
getSingleByteDBCodepage	データベースの 1 バイト・コード・ページを戻します。
reportWarning	DB2 Information Integrator ユーザーに対する警告をレポートします。
traceEnabled	DB2 Universal Database トレースが使用可能かどうかを検証します。
traceError	エラー・メッセージをトレースします。
traceException	例外とその呼び出しスタックをトレースします。
traceFunctionData	1 つのデータ・トレース・パラメーターを使用して関数データをトレースします。
traceFunctionData	2 つのデータ・トレース・パラメーターを使用して関数データをトレースします。
traceFunctionData	3 つのデータ・トレース・パラメーターを使用して関数データをトレースします。
traceFunctionEntry	関数項目をトレースします。
traceFunctionReturnCode	関数戻りコードをトレースします。

getAppCBStatus メソッド

目的 アプリケーション制御ブロックの状況を戻します。

構文

```
public static final int getAppCBStatus()
```

パラメーター

なし。

戻り値 アプリケーション制御ブロック状況。

スロー なし。

getAuthid メソッド

目的 現行データベースの許可 ID を戻します。

構文

```
public static final java.lang.String getAuthid()
                                          throws WrapperException
```

パラメーター
なし。

戻り値 データベース許可 ID。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

getCodepage メソッド

目的 現行データベースのコード・ページを戻します。

構文

```
public static final int getCodepage()
```

パラメーター
なし。

戻り値 コード・ページ。

スロー なし。

getDB2InstallPath メソッド

目的 DB2 Universal Database のインストール・パスを戻します。

構文

```
public static final java.lang.String getDB2InstallPath()
                                                  throws WrapperException
```

パラメーター
なし。

戻り値 DB2 information Integrator のインストール・パス。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

getDB2InstancePath メソッド

目的 DB2 Universal Database のインスタンス・パスを戻します。

構文

```
public static final java.lang.String getDB2InstancePath()
                                                   throws WrapperException
```

パラメーター
なし。

戻り値 DB2 Information Integrator のインスタンス・パス。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

getDB2Release メソッド

目的 このラッパーが現在動作している DB2 Universal Database リリースおよび

フィックスパックを戻します。この値は、それぞれの DB2 Universal Database フィックスパックで更新されます。

構文

```
public static final int getDB2Release()
```

パラメーター

なし。

戻り値 DB2 Information Integrator のリリース。

スロー なし。

getDoubleByteDBCodepage メソッド

目的 データベースの 2 バイト・コード・ページを戻します。

構文

```
public static final int getDoubleByteDBCodepage()
```

パラメーター

なし。

戻り値 2 バイト・コード・ページ。

スロー なし。

getIsolationLevel メソッド

目的 現行データベースの分離レベルを戻します。

構文

```
public static final int getIsolationLevel()
```

パラメーター

なし。

戻り値 分離レベル。

スロー なし。

getSingleByteDBCodepage メソッド

目的 データベースの 1 バイト・コード・ページを戻します。

構文

```
public static final int getSingleByteDBCodepage()
```

パラメーター

なし。

戻り値 1 バイト・コード・ページ。

スロー なし。

reportWarning メソッド

目的 DB2 Information Integrator ユーザーに対する警告をレポートします。 DB2 Information Integrator 警告は、SQL コード、呼び出し元関数名、およびトークンのセットで構成されます。それぞれの有効な SQL コードは警告メッセージを識別します。警告メッセージには、プレースホルダーが含まれる可能性があり、メッセージがユーザーに報告される前に指定されたトークンに置き換えられます。

構文

```
public static final void reportWarning(int sqlCode,
                                       java.lang.String funcName,
                                       java.lang.String[] tokens)
                                throws WrapperException
```

パラメーター

表 224. reportWarning メソッドのパラメーター

名前	説明
sqlcode	レポートされるエラーの事前定義された SQL コード。
funcName	エラーをレポートする関数の名前。ストリング値の最大長は 5 文字です。クライアント・プログラムは、SQLCA の SQLERRP フィールドを介してこのストリング値にアクセスできます。ストリング値は、SQL の接頭部付きの大文字で表示されます。
tokens	メッセージの置換トークン。

戻り値 なし。

スロー 操作が失敗した場合には、WrapperException オブジェクト。

traceEnabled メソッド

目的 DB2 Universal Database トレースが使用可能かどうかを検証します。

構文

```
public static final boolean traceEnabled()
```

パラメーター
なし。

戻り値 トレースが使用可能の場合は、真の値。それ以外の場合は、値は偽です。

スロー なし。

traceError メソッド

目的 エラー・メッセージをトレースします。

構文

```
public static final void traceError(int funcID,
                                    java.lang.String funcName,
                                    int probe,
                                    java.lang.String errorData)
```

パラメーター

表 225. traceError メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。
probe	プローブ ID。
errorData	トレースするエラー・データ。

戻り値 なし。

スロー なし。

traceException メソッド

目的 例外とその呼び出しスタックをトレースします。

構文

```
public static final void traceException(int funcID,
                                        java.lang.String funcName,
                                        int probe,
                                        java.lang.Throwable exception)
```

パラメーター

表 226. traceException メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。
probe	プローブ ID。
exception	トレースする例外。

戻り値 なし。

スロー なし。

traceFunctionData メソッド

目的 1 つのデータ・トレース・パラメーターを使用して関数データをトレースします。

構文

```
public static final void traceFunctionData(int funcID,
                                           java.lang.String funcName,
                                           int probe,
                                           java.lang.String data)
```

パラメーター

表 227. traceFunctionData メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。
probe	プローブ ID。
data	トレースするデータ。

戻り値 なし。

スロー なし。

traceFunctionData メソッド

目的 2 つのデータ・トレース・パラメーターを使用して関数データをトレースします。

構文

```
public static final void traceFunctionData(int funcID,
                                           java.lang.String funcName,
                                           int probe,
                                           java.lang.String data1,
                                           java.lang.String data2)
```

パラメーター

表 228. traceFunctionData メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。
probe	プローブ ID。
data1	トレースする最初のデータ集合。
data2	トレースする 2 番目のデータ集合。

戻り値 なし。

スロー なし。

traceFunctionData メソッド

目的 3 つのデータ・トレース・パラメーターを使用して関数データをトレースします。

構文

```
public static final void traceFunctionData(int funcID,
                                           java.lang.String funcName,
                                           int probe,
                                           java.lang.String data1,
                                           java.lang.String data2,
                                           java.lang.String data3)
```

パラメーター

表 229. traceFunctionData メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。
probe	プローブ ID。
data1	トレースする最初のデータ集合。
data2	トレースする 2 番目のデータ集合。
data3	トレースする 3番目のデータ集合。

戻り値 なし。

スロー なし。

traceFunctionEntry メソッド

目的 関数項目をトレースします。

構文

```
public static final void traceFunctionEntry(int funcID,
                                             java.lang.String funcName)
```

パラメーター

表 230. traceFunctionEntry メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。

戻り値 なし。

スロー なし。

traceFunctionReturnCode メソッド

目的 関数戻りコードをトレースします。

構文

```
public static final void traceFunctionReturnCode(int funcID,
                                                  java.lang.String funcName,
                                                  int returnCode)
```

パラメーター

表 231. traceFunctionReturnCode メソッドのパラメーター

名前	説明
funcID	呼び出し元関数のトレース ID。
funcName	呼び出し元関数の名前。
returnCode	戻りコード。

戻り値 なし。

スロー なし。

関連概念:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ラッパー・トレース機能』

関連資料:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『ラッパー・ユーティリティー・クラス』
- 186 ページの『WrapperException クラス (Java)』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 204 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 204 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 204 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは『キーボード・ショートカットおよびアクセラレーター: Common GUI help』を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 「*インフラストラクチャー・トピック (DB2 共通ファイル)*」の『ドット 10 進シンタックス・ダイアグラム』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』
- 『メニューおよびテキストのフォントを変更する: Common GUI help』

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム(本プログラムを含む)との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケ

ーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

商標

以下は、IBM Corporation の商標です。

IBM
DB2

以下は、それぞれ各社の商標または登録商標です。

Java は、Sun Inc. の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

C

F

J

N

P

Q

R

S

U

W

IBM と連絡を取る

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Information Integrator についての情報は、電話または Web から入手することができます。

米国にお住まいの場合は、以下のいずれかの番号にお問い合わせください。

- 製品の注文または一般情報の入手: 1-800-IBM-CALL (1-800-426-2255)
- 資料の注文: 1-800-879-2755

Web 上で http://www.ibm.com/software/data/integration/db2ii/support.html にアクセスします。このサイトには、最新のテクニカル・ライブラリーに関する情報、資料の注文、クライアントのダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースのリンクが含まれています。

お住まいの国や地域の IBM 事務所の所在地を調べる場合は、Web 上で IBM Directory of Worldwide Contacts (www.ibm.com/planetwide) を参照してください。

資料についてのコメント

お客様のフィードバックは IBM が良質な情報を提供する助けになります。この資料や他の DB2 Information Integrator の資料についてのコメントをお送りください。コメントの送付には、以下のいずれかの方法を利用することができます。

- www.ibm.com/software/data/rcf で、オンラインの読者コメント・フォームを使用して送信する。
- 電子メール (E メール) で comments@us.ibm.com に送信する。お送りいただく情報には、製品の名前、製品のバージョン番号、および資料の名前と部品番号 (該当する場合) を必ず含めてください。特定の本文についてコメントする場合は、本文の位置 (たとえば、タイトル、表の番号、またはページ番号) を含めてください。

Printed in Japan

SC88-9922-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12