PUBLIC (公開)

SAP BusinessObjects Business Intelligence プラットフォーム

ドキュメントバージョン: 4.3 Support Package 4 – 2023-12-07

データアクセスドライバ開発者ガイド

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

目次

1 ドキュメント履歴 **3**

2 このガイドについて **4**

3 データアクセス **Driver Development** システム **5**

3.1 システムアーキテクチャ 5

3.2 データアクセスドライバの使用 6

オープンドライバ 7

ドライバ設定ファイルの設定 7

ドライバの設定 9

接続プールの使用 12

新規接続ウィザードの設定 14

4 **DDK** 環境の設定 **16**

5 接続の作成 **17**

6 データ要求の準備 **19**

7 データソースのデータの要求 **22**

8 接続の終了 **23**

1 ドキュメント履歴

以下の表は、重要なドキュメント変更の概要です。

バージョン	日付	説明
データアクセスドライバ開発者ガイド	2016 年 8 月	このドキュメントの初版です。

2 このガイドについて

このガイドでは、データアクセス Driver Development Kit (DDK) を使用してカスタムドライバを開発する方法、および Open Connectivity のドライバをカスタマイズする方法、SAP BusinessObjects BI システムの一部としてドライバを使用する方法について説明します。

また、DDK では、BI プラットフォームから接続オブジェクトにアクセスし、カスタムドライバで接続オブジェクトを使用するための Java API が提供されます。

このガイドの対象読者

データアクセスドライバ開発者ガイドは、カスタムドライバを作成し、カスタマイズされたドライバを SAP BusinessObjects BI プラットフォームの一部として使用する開発者を対象としています。

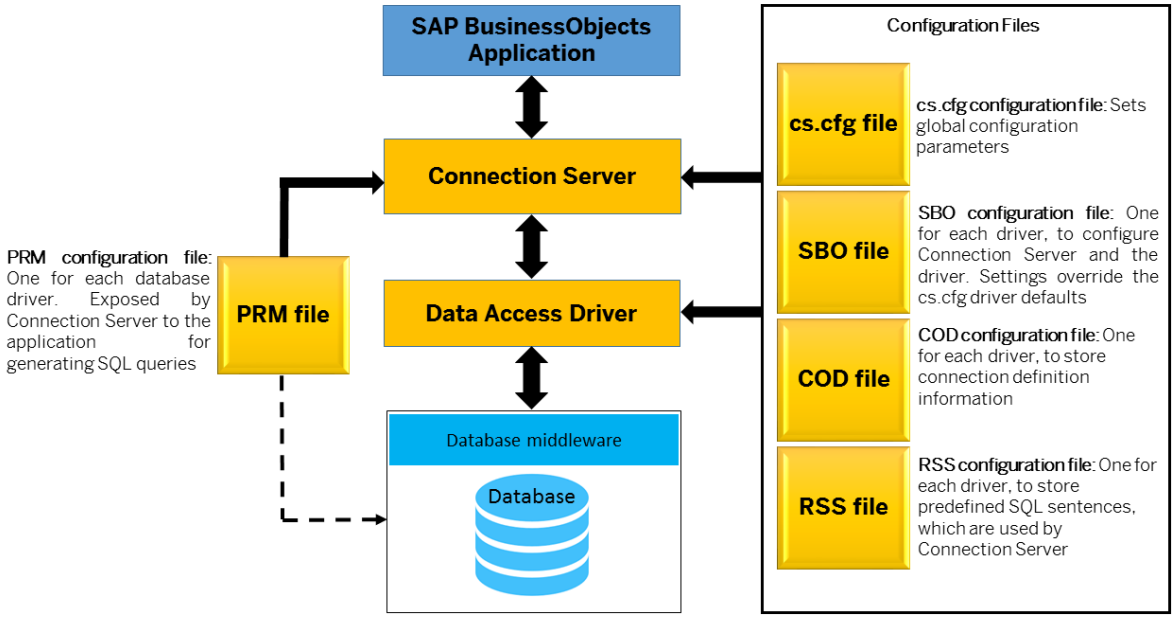
このガイドをよりよく理解するには、以下のことが必要です。

- Java テクノロジーについての知識
- Java アプリケーションの開発およびテストについての経験
- Connection Server を使用する SAP BusinessObjects アプリケーションの理解
- Connection Server 設定の経験
- カスタムドライバ向けのデータベース機能についての知識

3 データアクセス Driver Development システム

3.1 システムアーキテクチャ

以下の図は、データアクセスドライバの SAP BusinessObjects BI プラットフォーム設定での位置付けを説明しています。



データアクセスドライバ

データアクセスドライバは、Connection Server とデータベースのミドルウェアとの間のインタフェースとして機能するソフトウェアコンポーネントです。

データアクセスドライバでは、次のタスクを実行できる必要があります。

- データベースとの接続を確立し、接続を管理する。
- データベースのメタデータを取得する。
- SQL クエリなど、Connection Server からのデータ要求を受信する。
- データベースへのクエリを処理のために転送する。
- 要求されたデータのデータベースからの受信、および Connection Server への正しい形式でのデータを返送する。

Connection Server

Connection Server は、SAP BusinessObjects アプリケーションとデータソースとの間の接続を管理するデータアクセスソフトウェアです。Connection Server は、ユニバースデザインツール、情報デザインツール、および SAP BusinessObjects Web Intelligence などのアプリケーションがデータソースに接続し、クエリを実行できるようにします。

設定ファイル

設定ファイルでは、以下のシステム間の接続を設定するためのパラメータを定義します。

- アプリケーションと Connection Server
- アプリケーションとデータアクセスドライバ
- Connection Server とデータアクセスドライバ

3.2 データアクセスドライバの使用

Driver Development Kit (DDK) API を使用すると、データアクセスドライバを開発して、Connection Server で使用することができます。

SAP BusinessObjects アプリケーションが、利用可能なドライバがない場合でもデータソースにアクセスできるようにすることができます。たとえば、XML データファイルや Web サービスなどの非リレーショナルデータソースへの接続を作成することができます。

また、DDK API を使用して、SAP BusinessObjects アプリケーションと以下のようなさまざまなデータベースとの間にデータベースに依存しない接続を提供することができます。

- SQL 指向データベース
- XML-SQL データベース
- スプレッドシートやカンマ区切り値 (CSV) フラットファイルなど、表形式のデータソース

SAP BusinessObjects アプリケーションは、データソースに適用される SQL クエリを生成します。データベースミドルウェアに応じて、ドライバは生成されたクエリを解析し、データベースで処理できるフォームに変換する必要があります。DDK API パッケージには、以下のようなデータアクセスドライバの SQL パーサを開発するためのクラスおよびメソッドを提供するサンプルが含まれています。

- オープンドライバ
- 設定ファイル
- 接続プール
- 新規接続ウィザードの設定

3.2.1 オープンドライバ

オープンドライバは、SAP BusinessObjects Business Intelligence プラットフォームで配布されている標準のデータアクセスドライバと同じように動作します。

オープンドライバは以下を実行します。

- SQL クエリの構文およびデータソースの形式が正しいことを確認する。
- データソースに SQL クエリを適用する。
- クエリによって生成されたデータを Connection Server に返し、アプリケーションで使用できるようにする。

3.2.2 ドライバ設定ファイルの設定

ドライバをカスタマイズするには、ドライバの以下の設定ファイルを作成する必要があります。

- PRM 設定ファイル
- COD 設定ファイル
- SBO 設定ファイル
- RSS 設定ファイル

3.2.2.1 PRM 設定ファイル

各データアクセスドライバは PRM 設定ファイルに関連付けられています。これらのファイルは、データベースソフトウェアの機能に応じてアプリケーションの SQL クエリ生成方法を制御します。インフォメーションデザインツールなどのアプリケーションで、PRM ファイルが使用されます。

SAP BusinessObjects アプリケーションでは、データのソースとしてデータベースが使用されます。PRM ファイルは、これらのデータベースの機能を説明するパラメータが提供されます。

PRM ファイルでは、接続とデータベースに基づき、データベースに依存する要素によって、ユニバースで使用する SQL クエリが制御されます。ユニバース内で一部のデータベースパラメータを設定することもできます。したがって、これらの設定は PRM ファイルの設定を上書きします。

PRM ファイルは次のフォルダに保存されています。

- `<connectionserver-install-dir>¥connectionServer¥<RDBMS>` ディレクトリ。<RDBMS> はネットワークレイヤまたはミドルウェア名です。
- `<connectionserver-install-dir>¥connectionServer¥<RDBMS>¥extensions¥qt` ディレクトリ。これらの PRM ファイルは、拡張ファイルと呼ばれます。

拡張ファイルのパラメータの詳細については、インフォメーションデザインツールユーザガイドの章「SQL および MDX のリファレンス」を参照してください。

3.2.2.2 COD 設定ファイル

COD ファイルでは、接続を設定するドライバを選択すると新規接続ウィザードに表示される画面およびフィールドが定義されます。COD ファイルのセクションおよびエントリを設定して、新規接続ウィザードを設定する必要があります。

各オープンドライバで、COD ファイルは、対応する SBO ファイルと同じ場所にある必要があります。

3.2.2.3 SBO 設定ファイル

SBO ファイルでは、ConnectionServer とともに使用できるドライバが定義され、各ドライバのデフォルトパラメータが指定されます。このファイルには、開発に DDK を使用する利用可能な各ドライバのセクションが含まれている必要があります。また、SBO ファイルを編集して、接続に使用できるようドライバサンプルを登録する必要があります。

SBO ファイルの詳細については、データアクセスガイドを参照してください。

3.2.2.4 RSS 設定ファイル

各データアクセスドライバには RSS ファイルがあり、このファイルには、ConnectionServer で使用される定義済みの SQL 文が格納されます。

各オープンドライバで、COD ファイルは、対応する SBO ファイルと同じ場所にある必要があります。

3.2.2.5 設定ファイルの作成

オープンドライバが接続を確立する必要がある設定ファイルを作成する必要があります。

設定ファイルを作成するには、次の手順を実行します。

1. `<connectionserver-install-dir>¥connectionServer¥DDK¥examples¥open` ディレクトリに移動します。
2. `open_sample.prm` ファイルと `open_sampleen.cod` ファイルをコピーし、`<connectionserver-install-dir>¥connectionServer¥DDK¥examples¥open` ディレクトリに貼り付けます。
3. ファイルの名前を `<mydriver>.prm` および `<mydriver>en.cod` に変更します。

   > ⓘ 注記
   >
   > ドライバが使用されるマシンに、各言語に対応するローカライズ済みの PRM ファイルおよび COD ファイルがあることを確認する必要があります。たとえば、日本語環境では接続ウィザードで `<mydriver>ja.cod` ファイルが使用されます。

4. `open_sample.sbo` ファイルを以下のように編集します。

1. 以下の行を DataBases セクションにコピーします。

```
<DataBase Active="Yes" Name="DDK Sample Text Files">
 <Class
JARFile="dbd_open_sample">com.businessobjects.connectionserver.datasources.
opensample.CSVDriver</Class>
 <Parameter Name="Extensions">csv,open</Parameter>
 <Parameter Name="Description File">open_sample</Parameter>
 <Parameter Name="SQL Parameter File">open_sample</Parameter>
 <Parameter Name="Max Rows Available">Yes</Parameter>
</DataBase>
```

   DataBases セクションには、利用可能な各ドライバの DataBase エントリが含まれています。DataBase 要素によって、テキストファイルデータアクセスドライバがアクティブであること、およびデータソース名が定義されます。

2. DataBase 要素を以下のように編集します。

```
<DataBase Active="Yes" Name="My Database">
 <Class
JARFile="dbd_mydriver">com.businessobjects.connectionserver.datasources.myc
lass.CSVDriver</Class>
 <Parameter Name="Extensions">mydriver,open</Parameter>
 <Parameter Name="Description File">mydriver</Parameter>
 <Parameter Name="SQL Parameter File">mydriver</Parameter>
 <Parameter Name="Max Rows Available">Yes</Parameter>
</DataBase>
```

   Class 要素によって、Connection Server がドライバを利用可能として初期化するためにロードするクラスが設定されます。

3. ドライバのサポート内容に従って、Max Rows Available パラメータを設定します。

5. open_sample.sbo ファイルを保存します。

オープンドライバが接続を確立する必要がある設定ファイルを作成しました。

3.2.3 ドライバの設定

3.2.3.1 Driver Development Kit の設定インタフェース

DDK API には、データアクセスドライバの設定を処理するためのインタフェースが用意されており、以下のセクションで説明しています。

Context インタフェース

Context インタフェースは、データアクセスドライバの設定情報の管理に役立ちます。設定ファイル (PRM、SBO) からの情報およびユーザ認証情報や動的設定プロパティなどの情報を取得する一連のメソッドが提供されます。

DriverConfiguration インタフェース

`DriverConfiguration` インタフェースでは、ドライバの SBO ファイルに定義されているデフォルト設定が提供されます。これらの設定は、データアクセスドライバセッションの間は変更することができません。

追加の設定が必要なドライバを開発する場合は、SBO ファイルを更新して、`DriverConfiguration` 設定を追加する必要があります。これにより、ドライバ機能を拡張することができます。

PropertySet インタフェース

`PropertySet` インタフェースでは、動的なドライバ設定プロパティが提供されます。

プロパティは読み取り専用にするか、またはドライバによって変更することができます。つまり、コードで処理時の内容に応じて `PropertySet` 設定値を変更することができます。

DbParameterSet インタフェース

`DbParameterSet` インタフェースでは、ドライバの PRM ファイルに定義されている設定が提供されます。これらの設定は、データアクセスドライバセッションの間は変更することができません。

追加の設定が必要なドライバを開発する場合は、PRM ファイルを更新して、DbParameterSet 設定を追加するか、または新しいファイルを作成します。これにより、ドライバ機能を拡張することができます。

> ⓘ 注記
>
> PropertySet インタフェースの一部の設定は、DriverConfiguration インタフェースと同じであり、一部の設定は追加されたものです。詳細については、*Driver Development Kit Javadocs* を参照してください。

3.2.3.2 ドライバ設定の取得

`Context` インタフェースで定義されている `getconfiguration` を使用して、`DriverConfiguration` インタフェース設定を取得します。設定を取得したら、以下のメソッドを使用してデータを取得します。

- `boolean getBoolean(String key)`。論理値を返します。
- `int getInteger(String key)`。整数値を返します。
- `Object getObject(String key)`。オブジェクトを返します。
- `String getString(String key)`。文字列を返します。

> ⓘ 注記
>
> `DriverConfiguration` インタフェース設定の新しい値を設定するには、SBO ファイルを編集し、ドライバセッションを再起動します。

また、Context インタフェースに定義されている update(String connectionDefinition) メソッドを使用して、環境で有効なコンテキストが生成されるようにし、接続を作成することができます。このメソッドは、コンテキストの認証情報およびプロパティを指定された接続定義で更新します。

3.2.3.3 接続プロパティの設定と取得

Context インタフェースおよび PropertySet インタフェースに定義されている getProperties メソッドを使用して、ドライバプロパティの値を設定および取得することができます。使用するメソッドは、プロパティのデータ型によって異なります。たとえば、setIntProperty メソッドと getIntProperty メソッドを使用して、整数プロパティを設定および取得します。

例

接続の作成時に、ドライバサンプルでは、PropertySet インタフェースメソッドを使用して、Max Rows 値を取得した値に設定します。

```
setMaxRows (getContext ().getProperties ()
.getIntProperty (PropertySet.MAX_ROWS));
```

以下のコードでは、ID の大文字と小文字の区別のプロパティを区別ありに設定しています。

```
getContext ().getProperties ().setStringProperty
(PropertySet.IDENTIFIER_CASE, "SensitiveCase");
```

3.2.3.4 PropertySet のデフォルト値の設定

SAP BusinessObjects アプリケーションによってデータソースへの接続が確立されると、ConnectionServer によって PropertySet インタフェースの各プロパティがデフォルト値に設定されます。プロパティのデフォルト値は、以下のいずれかの方法で設定することができます。

- ConnectionServer が SBO 設定ファイルから値を取得するか、または SBO ファイルで値が定義されていない場合は cs.cfg ファイルから値を取得する。
- ConnectionServer が設定ファイルから値を取得し、設定ファイルに値が定義されていない場合はデータベースミドルウェアから値を取得する。
- ConnectionServer が、最初に設定ファイルをチェックせずにデータベースミドルウェアから値を取得する。
- ConnectionServer がプロパティを特定の値に設定する。

PropertySet インタフェースの詳細については、DDK Javadocs を参照してください。

3.2.3.5 データベースのパラメータ設定の取得

Context インタフェースに定義されている getDbParameters メソッドを使用して、データアクセスドライバの PRM ファイルから DbParameterSet インタフェース設定を取得することができます。DbParameterSet イン

タフェース設定の取得後に、`DbParameterSet` インタフェースに定義されているメソッドを使用して、ドメイン、関数名、SQL 構文など、特定のデータベース機能情報を取得します。

PRM ファイルの詳細については、『データアクセスガイド』を参照してください。

3.2.4 接続プールの使用

3.2.4.1 接続プールについて

ドライバがデータベースへの接続を確立して、データにアクセスします。データベースに接続するには、以下の方法を使用します。

- ConnectionServer で情報が必要になるたびに、データアクセスドライバがデータベースへの接続を確立し、データを取得して、接続を解除する。
- Connection Server が使用可能な接続を保持し、接続プールで接続の詳細を更新する。
  ConnectionServer でデータソースからの情報が必要になるたびに、データアクセスドライバが接続プールをチェックして、接続プールに未使用の適切な接続が含まれているかどうかを確認します。適切な接続が利用可能である場合、ConnectionServer ではその接続が使用されます。

3.2.4.2 接続プールの実装

プールの接続を管理するため、ドライバは以下の DDK インタフェースを実装します。

- `<Driver_Name> connection` クラスによって、`Connection` インタフェースが実装され、接続プールで接続を管理するメソッドが提供されます。
- `ConnectionPool` インタフェースは、接続プールを管理するためにドライバで使用されるメソッドを提供します。

Connection インタフェース

Connection インタフェースには、次の機能を持つメソッドが含まれます。

- 接続プールでは計算済みの一意のキーを使用して、接続プロファイルを識別する。
- 接続がアクティブのまま保持される期間を設定および取得する。
- 接続を終了する。

ConnectionPool インタフェース

`ConnectionPool` インタフェースには、次の機能を持つメソッドが含まれます。

- プール内の接続の詳細を取得する。
- プール内の接続を追加および解放する。
- 接続に対する排他的アクションを有効にするために、使用中の接続をロックする。
  ConnectionPool インタフェースのメソッドは、接続がプールに戻されるとロックを解放します。ロック機能により、接続に対して 1 つのコミットまたはロールバックのみが実行されるようになります。

3.2.4.3 プールの接続の取得

SAP BusinessObjects アプリケーションが接続プールの接続を要求すると、ドライバサンプルは以下の一連のメソッドを呼び出します。これは、createConnection メソッドが呼び出されると実行されます。

1. computeKey メソッドは、ユーザ認証情報 (ユーザログインおよびパスワード) によって接続プロファイルを識別するためのキーを計算します。
   Object key = CSVConnection.computeKey(filename);
2. getIntProperty メソッドは、POOL_TIME パラメータ値を取得します。
   int poolTime = Context().getProperties().getIntProperty (PropertySet.POOL_TIME);

   > ⓘ 注記
   >
   > POOL_TIME 設定によって、未使用の接続が接続プールに保持される期間が決定されます。これは、新しい接続ウィザードでの接続の作成時にユニバース作成者によって定義されます。

3. getBooleanProperty メソッドは、データソースが接続を共有できるかどうかを決定する場合に役立ちます。

   ```
   boolean sharedConnection = Context().getProperties()
   .getBooleanProperty(PropertySet.SHARED_CONNECTION);
   ```

4. getConnectionPool メソッドは、接続が稼働している場合にプールから接続を取得します。

   ```
   if (poolTime != 0) {
   connection = (CSVConnection)
   getConnectionPool().acquire(key, sharedConnection);
   if (connection != null) {
   connection.setDuration(poolTime);
   }
   }
   ```

   PoolTime0 値は、この種類の接続に接続プールを使用できないことを示します。
5. CSVConnection メソッドは、プールに利用可能な接続がない場合に新しい接続を作成します。

   ```
   if (connection == null) {
   connection = new CSVConnection(key, poolTime, new CSVFile (filename,
   separator, properties));
   }
   ```

3.2.4.4 プールからの接続のリリース

ドライバサンプルは、接続をリリースして、今後の操作に利用可能にすることによって、接続を終了します。これは、disconnect メソッドが呼び出されると実行されます。

releaseConnection メソッドは、poolTime 値が 0 の場合、プールへの接続をリリースします。それ以外の場合は、close メソッドが接続を終了します。

```
private void releaseConnection () throws DDKException
{
try
{
if (connection != null)
{
if (getContext ().getProperties ().getIntProperty (PropertySet.POOL_TIME) != 0)
{
getConnectionPool ().release (connection);
}
else
connection.close ();
Using a connection pool
connection = null;
}
}
...
}
```

> ⓘ 注記
>
> - ドライバは、共用接続が必要ではなくなった場合に再度プールにリリースする必要があります。
> - 接続プールは、共用接続を終了する必要があります。
> - ドライバは共用接続を明示的に終了することができません。これは、データソースへのライブ接続が失われる可能性があるためです。

3.2.5 新規接続ウィザードの設定

3.2.5.1 新規接続ウィザードについて

新規接続ウィザードは、ユニバースデザインツールおよびインフォメーションデザインツールに含まれています。新規接続ウィザードを使用して、データソースへの接続を作成し、データをユニバースに提供します。また、新規接続ウィザードでは、接続の設定に使用されるユーザ入力情報を収集します。

データアクセスドライバの COD 設定ファイルを設定して、新規接続ウィザードに表示される入力フィールドを定義します。このファイルは、対応する SBO ファイルと同じディレクトリに配置されている必要があります。

新規接続ウィザードは情報を収集します。ドライバコードに DDK API メソッドを実装して、収集された情報を取得します。接続ウィザードでは、この情報を使用して接続を設定および作成します。

3.2.5.2 新規接続ウィザードの機能

最初に、新規接続ウィザードは SBO ファイルの `DataBases` セクションを読み取って、データベースミドルウェアの選択ウィンドウに利用可能なドライバを表示します。

`Family` パラメータは、ドライバが表示される新規接続ウィザード内のセクションを設定します。`Description` ファイルパラメータは、ドライバに必要な設定情報を指定する COD ファイルの名前を定義します。

SBO パラメータの詳細については、『データアクセスガイド』を参照してください。

4 DDK 環境の設定

<Driver_Name> クラスは DBEnvironment インタフェースを実装して、特定のドライバおよび設定情報を処理します。ドライバは以下の一連のメソッドを呼び出します。

1. inquiry メソッドは、<Driver_Name> COD 設定ファイルの Inquiry パラメータ値から作成された XML 文字列を取得します。

   ```
   public String inquiry (String id, Map params) throws DDKException
   ```

   XML 文字列の構造は、以下のとおりです。

   ```
   <Inquiry ID = "InquiryID">
   <Entry>...</Entry>
   <Entry>...</Entry>
   <Entry>...</Entry>
   </Inquiry>
   ```

   > ⓘ 注記
   >
   > 上記手順の実行は、COD ファイルに設定されている条件によって異なります。

2. validate
   メソッドは、接続の定義を確認します。

   ```
   public void validate (ConnectionDefinition definition) throws DDKException
   ```

3. newDBEnvironmentInstance メソッドは、<Driver_Name> 環境クラスのインスタンスを作成します。

   ```
   public static DBEnvironment newDBEnvironmentInstance (String networkLayer,
   String dbms, Context context)
   throws DDKException
   {
   return new CSVEnvironment (networkLayer, dbms, context);
   }
   ```

5 接続の作成

<Driver_Name> クラスは OpenDriver インタフェースを実装します。このインタフェースは、DDK API の com.businessobjects.connectionserver.datasources.ddk パッケージの DBDriver インタフェースを拡張します。

Connection Server は以下の一連のメソッドを呼び出して、接続を作成します。

1. initDriver メソッドはドライバを初期化します。

```
public static void initDriver (DriverConfiguration config,
String networkLayer, String dataBase)
```

2. newDBDriverInstance メソッドは、ドライバのインスタンスを作成して返します。

```
public static DBDriver newDBDriverInstance (Context context, ConnectionPool
pool)
throws DDKException
{
return new <name -of-the-driver> (context, pool);
}
```

   <Driver_Name> constructor メソッドは、以下を行います。

   1. SAP BusinessObjects アプリケーションによってクエリが生成され、コンストラクタによってパーサが割り当てられてクエリが解析されます。

```
parser = ParserFactory.newParser ();
```

   2. ドライバをその設定から初期化します。

```
DriverConfiguration config = context.getConfiguration ();
{
...
try
{
setArrayFetchSupported (config.getBoolean ("Array Fetch Available"));
}
...
}
```

3. connect メソッドはドライバをデータソースに接続します。

```
public void connect () throws DDKException
{
createConnection ();
setConnected (true);
...
}
```

   createConnection メソッドは、新規接続ウィザードから認証情報を取得し、接続を設定します。

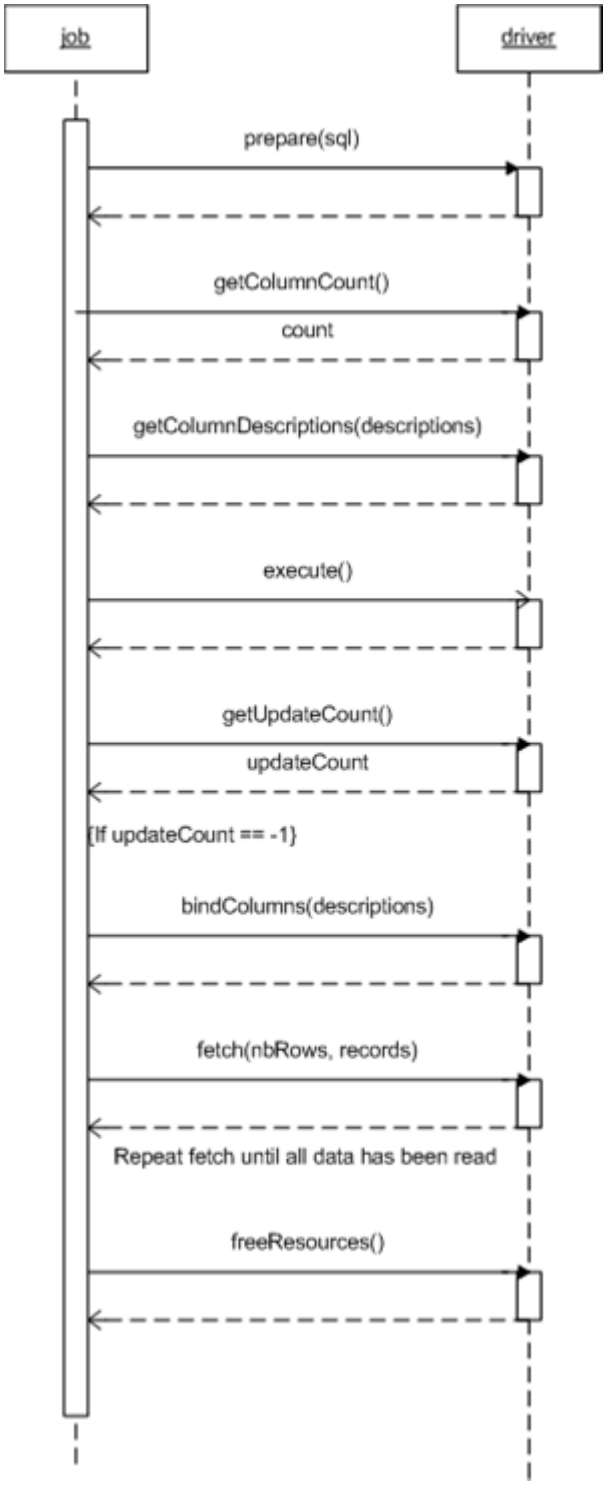
```
private void createConnection () throws DDKException
{
try
{
String filename = (String)getContext ().getCredentials ()
.get ("DATASOURCE");
```

```
String separator = (String)getContext ().getCredentials ()
.get ("SEPARATOR");
...
}
```

> **注記**
>
> `createConnection` メソッドを使用すると、接続プールを実装することができます。

6　データ要求の準備

以下は、ドライバによって処理されるクエリの実行のシーケンス図を示しています。



ドライバの初期化と接続の作成後に、Connection Server はメタデータおよびデータに対する要求を処理することができます。以下の一連のドライバメソッドが呼び出されます。

1. データソースは `prepare` メソッドから要求を実行します。

```
public void prepare (String query) throws DDKException
{
try
```

```
{
statement = parser.parseQuery (query);
validateStatement ();
}
...
}
```

- データソースが SQL 言語をサポートしていない場合は、ドライバがクエリを解析します。
- データソースが SQL をサポートしている場合は、ドライバが検証のためにクエリを直接データソースに送信することができます。

2. `getColumnCount` メソッドは、`ResultSet` オブジェクトの列の数を返します。

```
public int getColumnCount () throws DDKException
{
return statement.getColumnNames ().size ();
}
```

3. `getColumnDescriptions` メソッドは、`ResultSet` オブジェクトの列の説明を返します。`getColumnDescriptions` は descriptionSet パラメータにクエリによって返された列の詳細を入力します。

```
public void getColumnDescriptions (DescriptionSet descriptionSet)
```

4. `BindColumns` メソッドは、データソース列とデータの取得に使用されるドライババッファのバインドを定義します。`BindColumns` は `DataManager` クラスを使用します。

```
public void bindColumns (DescriptionSet descriptionSet) throws DDKException
```

7 データソースのデータの要求

結果のメタデータの取得後に、Connection Server では、クエリを実行して、以下の一連のメソッドを呼び出すことができます。

1. `execute` メソッドはデータソースへの要求に適用されます。

```
public void execute () throws DDKException
```

2. `fetch` メソッドによってレコードのセットにデータソースから取得されたデータが入力されます。

```
void fetch (int requestedSize, RecordSet recordSet) throws DDKException
```

`requestedSize` は、データソースから取得するレコードの最大数です。`recordSet` は、ドライバによって入力されるレコードの出力セットです。

3. `freeResources` メソッドは、データの取得時に使用されたリソースをリリースします。

```
public void freeResources () throws DDKException
{
columnIndexes = null;
rowsFetched = 0;
Creating an Open driver
dataIterator = null;
fieldBinding = null;
metadataRequested
```

例

コードスニペットは、`sizeToFetch` の数のレコードまたはレコードが `sizeToFetch` 未満の場合は行の最大数を入力します。

```
for (int i = 0; i < sizeToFetch && (hasNext
= dataIterator.hasNext ()); i++)
{
String[] rowData = (String[])dataIterator.next ();
Record record = recordSet.nextRecord ();
for (int j = 0; j < columnIndexes.length; j++)
{
String cell = rowData[columnIndexes[j]];
try
{
fieldBinding[j].fetch (record, cell);
}
catch (NumberFormatException exception)
{
throw getContext ().getErrors ().getError
(ERROR_BAD_BINDING + exception.getMessage ());
}
}
}
```

8 接続の終了

ドライバで接続が使用されなくなると、その接続は終了します。Connection Server は、接続を終了する以下の一連のメソッドを呼び出すことができます。

1. `disconnect` メソッドは接続をリリースします。

```
public void disconnect () throws DDKException
{
releaseConnection ();
setConnected (false);
}
```

> ⓘ 注記
>
> 接続プールによって `releaseConnection` メソッドが実行されます。接続プールなしで、`<Driver_Name> Connection` オブジェクトの `close()` メソッドが呼び出されます。

2. `<Driver_Name> Connection` クラスの `close` メソッドによって、`ResultSet` オブジェクトがリリースされます。ドライバは、今後の操作に引き続き利用可能です。

```
public void close ()
{
}
```

3. `The finiDriver` メソッドによって、ドライバプロセス `public static void finiDriver ()` が終了します。

重要免責事項および法的情報

ハイパーリンク

リンクの一部は、アイコンやマウスオーバーテキストで分類されています。これらのリンクから、追加の情報を得ることができます。
アイコンについて。

- このアイコンが付いたリンク: SAP がホストしているものではない Web サイトに移動します。これらのリンクを使用することで、お客様は (お客様と SAP との契約書に別段の明示的な記載がない限り) 以下のことに同意することになります。
  - リンク先のサイトのコンテンツが SAP のドキュメンテーションではないこと。お客様は、この情報に基づいて SAP に対する製品クレームを推断することはできません。
  - SAP が、リンク先のサイトのコンテンツについて同意することも反対することもなく、また SAP がその利用可能性や正確性について保証しないこと。SAP は、かかるコンテンツの使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。
- このアイコンが付いたリンク: 当該の特定の SAP 製品又はサービスのドキュメンテーションから離れ、SAP がホストしている Web サイトに移動します。これらのリンクを使用することで、お客様は (お客様と SAP との契約書に別段の明示的な記載がない限り)、この情報に基づいて SAP に対する製品クレームを推断することはできないことに同意します。

外部プラットフォームでホストされているビデオ

一部のビデオは、サードパーティのビデオホスティングプラットフォームに置かれている場合があります。SAP では、これらのプラットフォームに保存されているビデオが将来にわたって利用できると保証することはできません。また、これらのプラットフォームにホストされている、いかなる広告またはその他のコンテンツ (関連ビデオまたは同じサイトでホストされている別のビデオに移動する場合など) については、SAP の管理外であり責任を負いません。

ベータおよびその他の試験的機能

試験的機能は、SAP が将来のリリースを保証する正式に提供される機能の範囲外です。これは、試験的機能は、SAP により通知なく理由の如何を問わず随時変更される場合があることを意味します。試験的機能は、本稼働使用のためのものではありません。お客様は、試験的機能を実際の運用環境で、又は十分なバックアップがとられていないデータとともに、デモンストレーション、テスト、試験、評価その他の方法で使用してはなりません。
試験的機能の目的は、早期にフィードバックを得ることで、それに応じて顧客の皆様やパートナーが将来の製品に影響を与えることを可能にすることです。SAP コミュニティなどにおいてフィードバックを提供することで、お客様は、投稿物や二次的著作物の知的財産権が SAP の独占的所有物であり続けることを承認することになります。

コード例

ソフトウェアのコーディングやコードスニペットはすべて、例です。それらは、本稼働使用のためのものではありません。コード例は、構文や表現規則を分かりやすく説明し視覚化することのみを目的としています。SAP は、コード例の正確性や完全性について保証しません。SAP は、コード例の使用により発生した過誤や損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、損害に対して一切責任を負いません。

偏見のない表現

SAP は、ダイバーシティ & インクルージョンの文化を支持しています。SAP の文書では、可能な限り、文化、民族性、ジェンダー、および障がいの有無を問わず、すべての人々に対する偏見を伴わない表現を採用します。

www.sap.com/contactsap

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

本書のいかなる部分も、SAP SE 又は SAP の関連会社の明示的な許可なくして、いかなる形式でも、いかなる目的にも複製又は伝送することはできません。本書に記載された情報は、予告なしに変更されることがあります。

SAP SE 及びその頒布業者によって販売される一部のソフトウェア製品には、他のソフトウェアベンダーの専有ソフトウェアコンポーネントが含まれています。製品仕様は、国ごとに変わる場合があります。

これらの文書は、いかなる種類の表明又は保証もなしで、情報提供のみを目的として、SAP SE 又はその関連会社によって提供され、SAP 又はその関連会社は、これら文書に関する誤記脱落等の過失に対する責任を負うものではありません。SAP 又はその関連会社の製品及びサービスに対する唯一の保証は、当該製品及びサービスに伴う明示的保証がある場合に、これに規定されたものに限られます。本書のいかなる記述も、追加の保証となるものではありません。

本書に記載される SAP 及びその他の SAP の製品やサービス、並びにそれらの個々のロゴは、ドイツ及びその他の国における SAP SE （又は SAP の関連会社）の商標若しくは登録商標です。本書に記載されたその他すべての製品およびサービス名は、それぞれの企業の商標です。

商標に関する詳細の情報や通知については、https://www.sap.com/japan/about/legal/trademark.html をご覧ください。

THE BEST RUN SAP