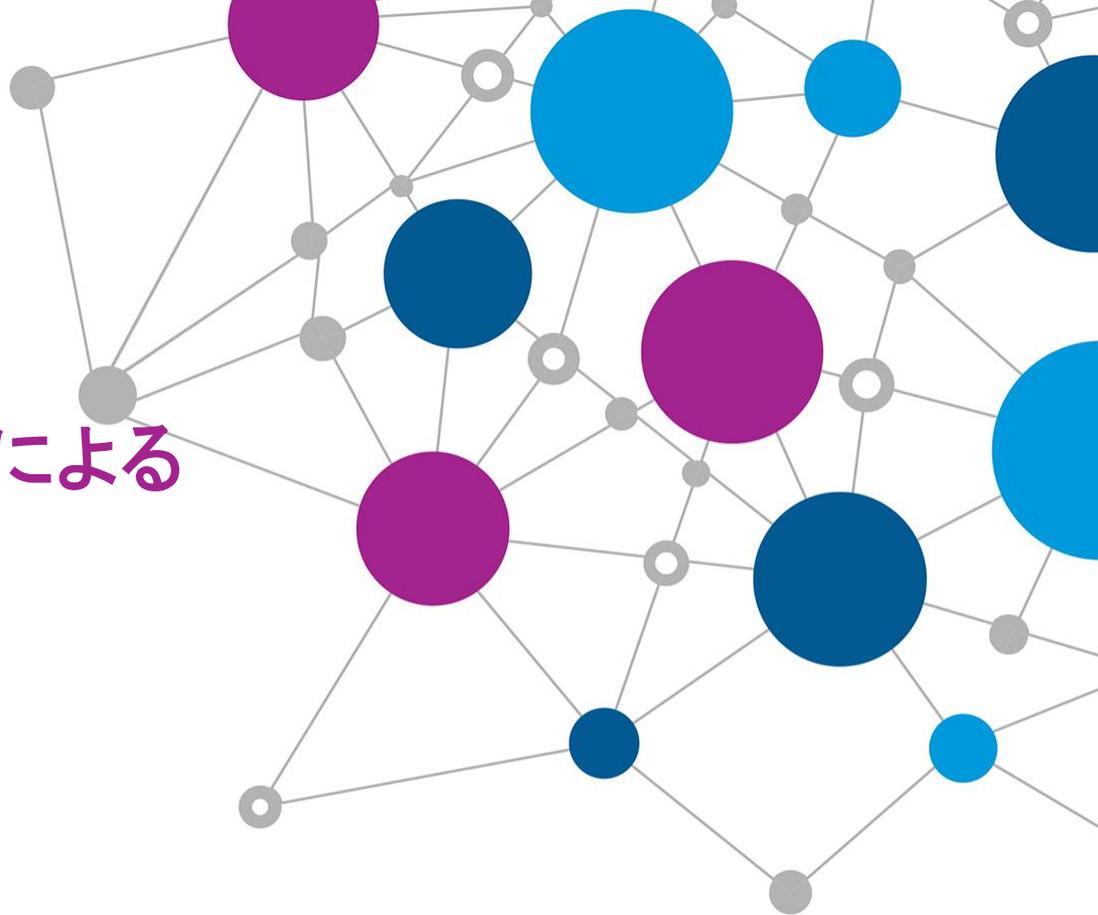


Magic xpi コネクタビルダによる コネクタ作成方法

マジックソフトウェア・ジャパン株式会社



OUTPERFORM THE FUTURE™



目次: Magic xpi コネクタビルダによるコネクタ作成方法

- **第1章 Methodインタフェースコネクタの作成方法**
 - 1.1 Methodインタフェースについて
 - 1.2 クラスの作成
 - 1.3 コネクタビルダによるコネクタの作成
 - 1.4 Magic xpiスタジオにおけるコネクタの使用
- **第2章 Data Mapperインタフェースコネクタの作成方法**
 - 2.1 Data Mapperインタフェースについて
 - 2.2 コネクタビルダによるコネクタの作成
 - 2.3 UIクラス作成
 - 2.4 Runtimeクラス作成
 - 2.5 Magic xpiスタジオにおけるコネクタの使用



第1章

Methodインタフェース コネクタの作成方法



OUTPERFORM THE FUTURE™

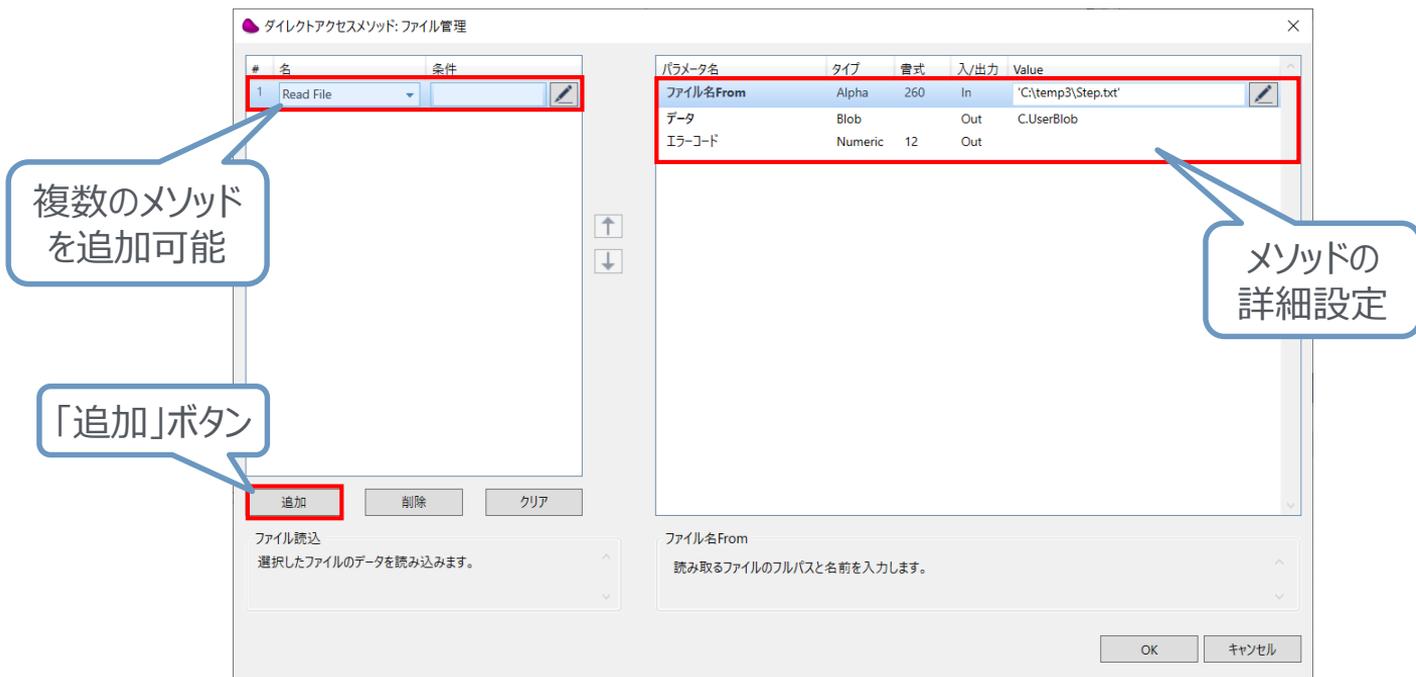
1.1 Method インタフェースについて



OUTPERFORM THE FUTURE™

1.1 Methodインタフェースについて

- ・ xpi標準のファイル管理コンポーネントなどに見られるインタフェース。
- ・ 左ペインの「追加」ボタンでメソッドを追加し、右ペインでメソッドの詳細を設定します。

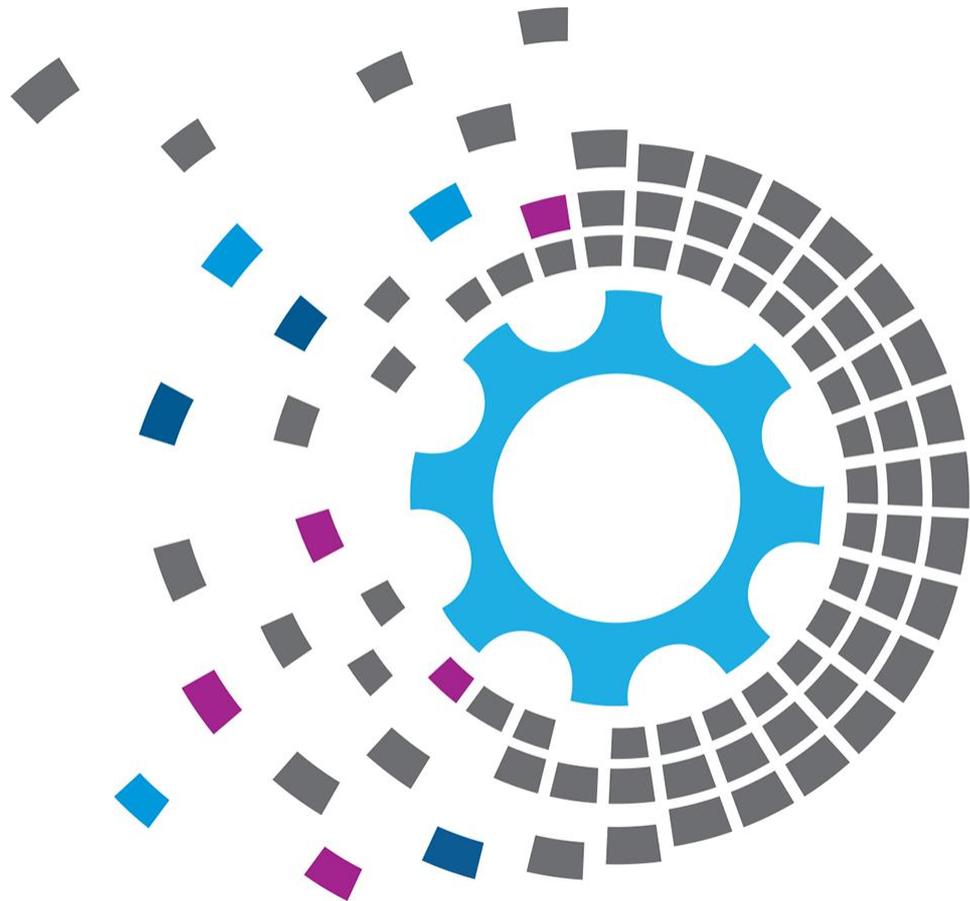


1.1 Methodインタフェースについて

- Methodインタフェースコネクタを作成する場合、UIは前頁の画面固定となり、開発不要です。
- コネクタで使用するメソッドを持つクラスを.NETまたはJavaで開発します。



1.2 クラスの作成



OUTPERFORM THE FUTURE™

1.2 クラスの作成

- Methodインタフェースコネクタを作成する場合、予めメソッドを持つクラスを作成します。
- 開発言語は .NET もしくは Javaのどちらかを選択することができます。

Visual Studio(.NET)によるクラスの開発例

The image shows a screenshot of Visual Studio Express 2017 for Windows Desktop. The main window displays the source code for a class named `Class1` in the `ClassLibrary1` namespace. The code includes using statements for `System`, `System.Collections.Generic`, `System.Linq`, and `System.Text`. The class `Class1` has a private integer field `age`, a `SetAge(int age)` method that sets the field, and a `GetAge()` method that returns the value of the field.

A blue callout bubble points to the code with the text: "staticなクラスメソッド、インスタンスメソッド、どちらも定義可能" (Both static class methods and instance methods can be defined).

In the bottom right, a Windows File Explorer window shows the output directory. A red box highlights the file `ClassLibrary1.dll`, with a blue callout bubble pointing to it containing the text: "ビルドしてdllを作成" (Build to create DLL).

| 名前 | 日時 | 種類 | サイズ |
|-------------------|-----------------|--------------------|-------|
| ClassLibrary1.dll | 2019/12/06 9:49 | アプリケーション拡張 | 4 KB |
| ClassLibrary1.pdb | 2019/12/06 9:49 | Program Debug D... | 12 KB |



OUTPERFORM THE FUTURE™

1.3

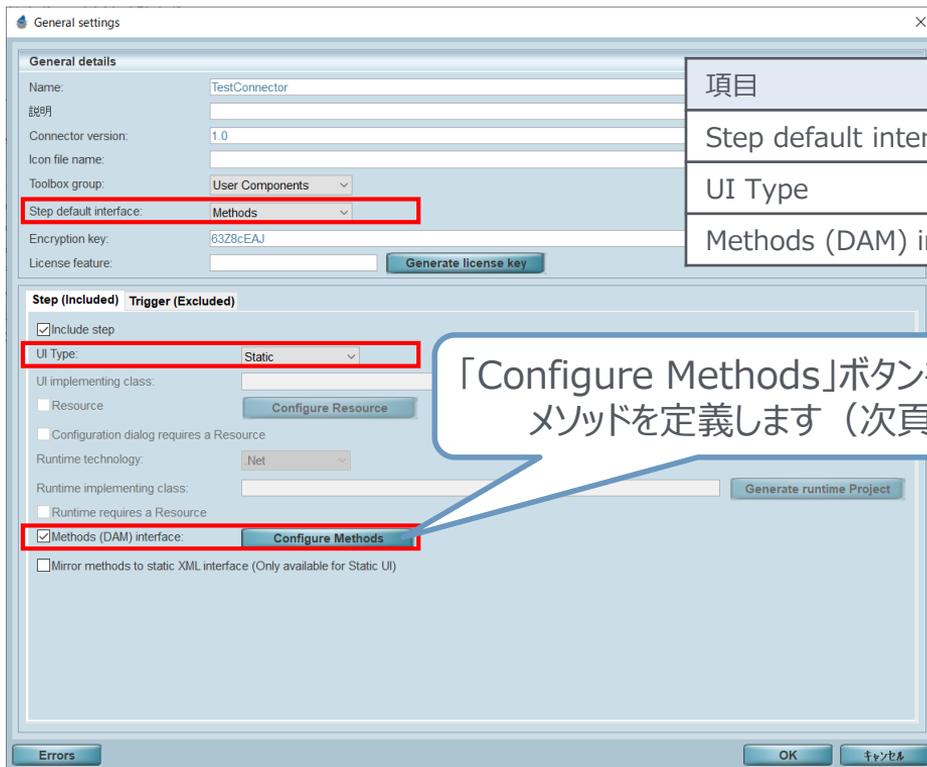
コネクタビルダによる コネクタの作成



OUTPERFORM THE FUTURE™

1.3 コネクタビルダによるコネクタの作成

- コネクタビルダでコネクタを作成します。



| 項目 | 値 |
|-------------------------|----------|
| Step default interface | Methods |
| UI Type | Static |
| Methods (DAM) interface | チェックをつける |

「Configure Methods」ボタンを押し、メソッドを定義します（次頁）



1.3 コネクタビルダによるコネクタの作成

- 1.2で作成したクラスをロードします。

| 項目 | 値 |
|------------|---------------|
| Technology | .Net または Java |
| Assembly | dll または jar |
| Class name | ネームスペース.クラス名 |

| メソッド名 | Num of Args | Method Type |
|---|-------------|-------------|
| <input checked="" type="checkbox"/> SetAge | 1 | Instance |
| <input checked="" type="checkbox"/> GetAge | 0 | Instance |
| <input checked="" type="checkbox"/> Constructor 1 | 0 | Constructor |

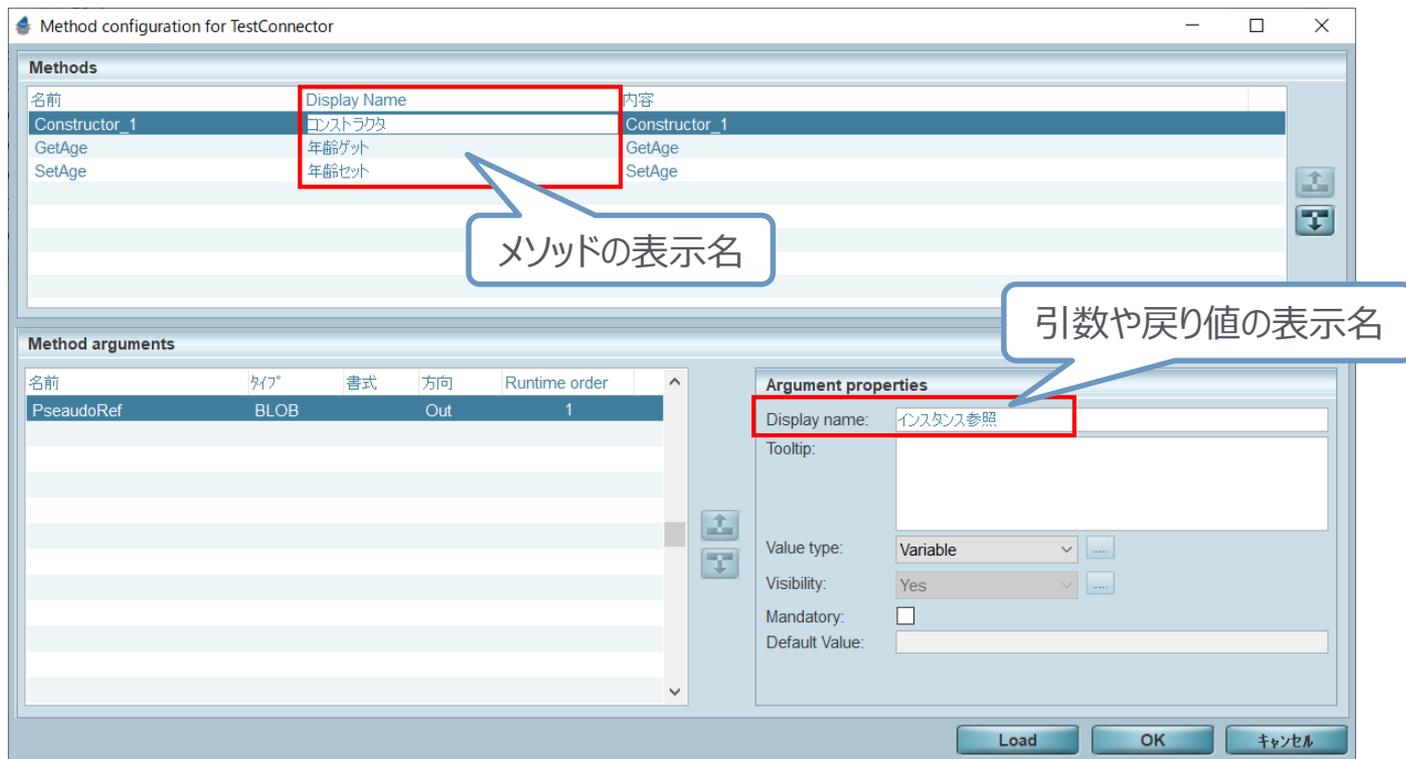
クラスで定義されたメソッドが表示される
使用したいメソッドにチェックをつける

「Load」ボタン押下 (次頁)



1.3 コネクタビルダによるコネクタの作成

- 各メソッドごとに、メソッド、引数、戻り値の表示名を設定します。



1.4

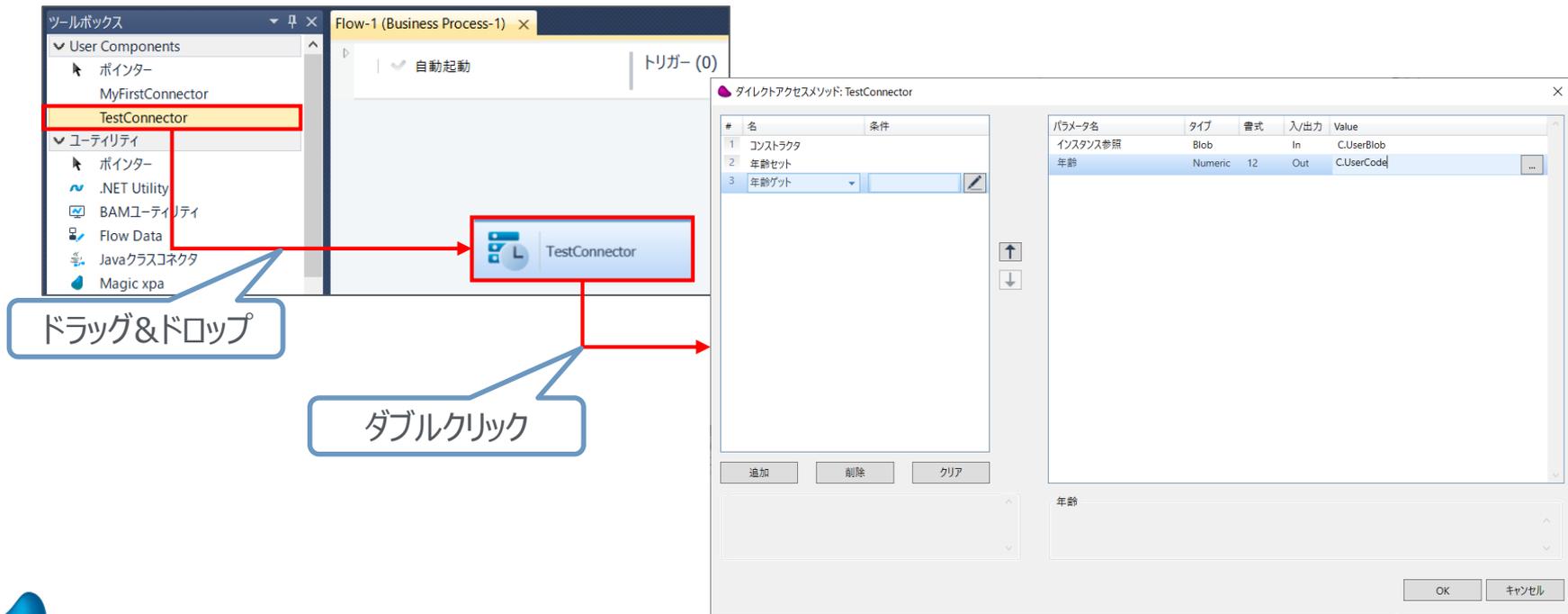
Magic xpiスタジオ におけるコネクタの使用



OUTPERFORM THE FUTURE™

1.4 Magic xpiスタジオにおけるコネクタの使用

- xpiスタジオを開き、作成したコネクタをフローエリアにドラッグ&ドロップします。
- コネクタをダブルクリックし、設定画面を開くと、コネクタビルダで定義したメソッドが使用できます。



第2章

Data Mapperインタフェース コネクタの作成方法



OUTPERFORM THE FUTURE™

2.1 Data Mapper インタフェースについて



OUTPERFORM THE FUTURE™

2.1 Data Mapperインタフェースについて

- xpi標準のSalesforceコンポーネントなどに見られるインタフェース。
- 設定画面で設定後、データマッパーを使用して設定を行います。

設定画面

The screenshot shows a configuration window titled "Salesforce設定". It contains several sections for setting up a Salesforce connection and data mapping. A blue arrow points from this window towards the Data Mapper window.

| | |
|-------------|--|
| 接続 | リソース名: SalesforceResource |
| オペレーション | オブジェクト: Product2 |
| オペレーション: | クエリ |
| 戻フィールド: | <input checked="" type="checkbox"/> 全て |
| 戻子オブジェクト: | |
| 結果オプション | 結果保存: 変数 (C.UserBlob) |
| オペレーション成功: | Variable (C.オペレーション成功) |
| 分割ファイルオプション | ディレクトリ: |
| | 接頭辞: |
| | ファイル毎レコード数: |
| | 分割数: |

Buttons: XSDリフレッシュ, OK, 取消

データマッパー

The screenshot shows the "Product2を検索 (クエリ)" window. It is divided into two main panes. The left pane is titled "送り元" and is currently empty. The right pane is titled "送り先" and displays a tree view of the Salesforce schema. The "Product2" object is expanded, showing its fields. The "ProductCode" field is selected, and a blue arrow points from it to the "送り元" pane, indicating the mapping process.

| | |
|-----|------------------|
| 送り先 | IFO_Salesforce |
| | Product2 |
| | ObjectID |
| | Id |
| | SOQL |
| | WHERE |
| | Fields |
| | type |
| | Id |
| | Name |
| | ProductCode |
| | Description |
| | IsActive |
| | CreatedDate |
| | CreatedById |
| | LastModifiedDate |
| | LastModifiedById |
| | SystemModstamp |
| | Family |

クエリ (通常API)



OUTPERFORM THE FUTURE™

2.1 Data Mapperインタフェースについて

- Data Mapperインタフェースコネクタを作成する場合、独自の設定画面の開発が必要です。設定画面(UIクラス)の開発には.NETを使用します。
- データマッパーの送り先には、XML、JSON、FlatFileのいずれかを選択することができ、XMLやJSONの場合はスキーマファイルを準備、FlatFileの場合は区切文字や項目の名前、型などをプログラムの中で定義する必要があります。
- 実行時の動作を実装したRuntimeクラスを作成する必要があります。Runtimeクラスの開発には.NET、Java、Magic xpaのいずれかを使用することができます。



2.2

コネクタビルダによる コネクタの作成



OUTPERFORM THE FUTURE™

2.2 コネクタビルダによるコネクタの作成

- コネクタビルダでコネクタを作成します。

General settings

General details

Name: SQLCommander

説明

Connector version: 1.0

Icon file name: StoredProcedure.gif

Toolbox group: Connectors

Step default interface: Data Mapper

Encryption key: F9YOLC7b

License feature:

Step (Included) Trigger (Excluded)

Include step

UI Type: Dynamic

UI implementing class: SQLCommanderUI.SQLCommanderUI

Resource

Configuration dialog requires a Resource

Runtime technology: .Net

Runtime implementing class: SQLCommanderRuntime.SQLCommanderRuntime

Runtime requires a Resource

Methods (DAM) interface

Mirror methods to static XML interface (Only available for Static UI)

| 項目 | 値 |
|----------------------------|-----------------------------|
| Step default interface | Data Mapper |
| UI Type | Dynamic |
| UI implementing class | ネームスペース名.UIクラス名 |
| Runtime technology | .Net または Java または Magic xpa |
| Runtime implementing class | ネームスペース名.Runtimeクラス名 |
| Methods (DAM) interface | チェックをはずす |

この時点では、UIクラス、Runtimeクラスはまだ実装できていなくて大丈夫です。
ネームスペース名とクラス名だけ決めます。



2.3

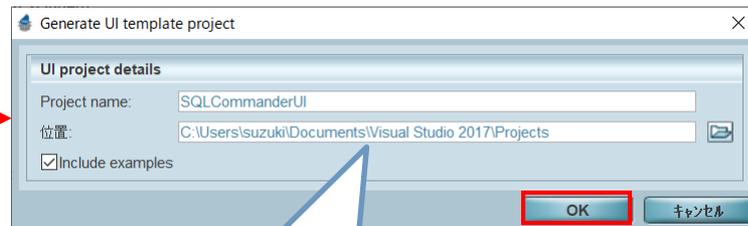
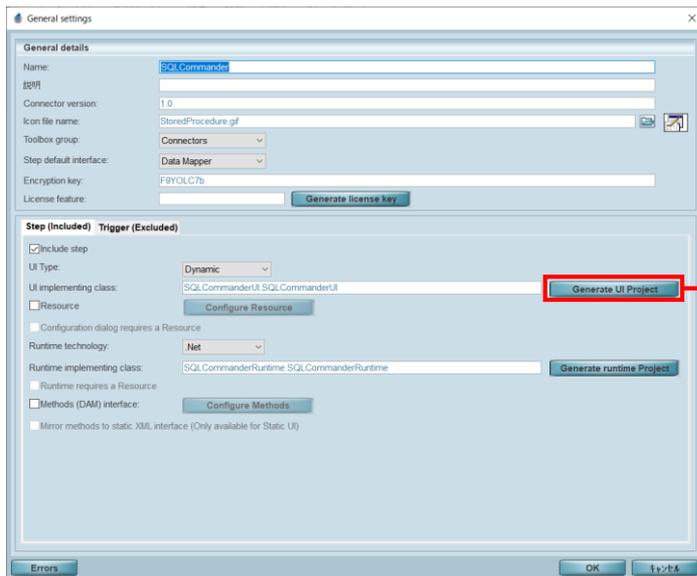
UIクラスの作成



OUTPERFORM THE FUTURE™

2.3 UIクラスの作成

- 「Generate UI Project」ボタンを押し、UIクラスのひな型を作成します。



プロジェクト名と
プロジェクト格納フォルダを
指定し「OK」ボタン押下



2.3 UIクラスの作成

- ・ サンプルコードつきの下記のクラスやメソッドが作成されますので、コードを修正します。
- ・ 設定画面は作成されないなので、Windowsフォームで別途作成する必要があります(下記赤字部分)。

| クラス | メソッド | 役割 | MVCモデル |
|-------------------|----------------------|--|------------|
| MyData | コンストラクタ | 設定画面で設定した内容を保持するためのクラス | Model |
| Windowsフォーム | コンストラクタ | ひな型が作成されないなので、Windowsフォームで設定画面を作成する | View |
| | ConfigurationChanged | 設定画面で設定が変更されたか否かを保持するプロパティ | |
| | ConfigurationSuccess | 設定画面で設定した内容が妥当であるか否かを保持するプロパティ | |
| コネクタビルダで入力したUIクラス | コンストラクタ | 設定画面やマップパー画面を開くときの挙動を定義するクラス | Controller |
| | Configure | 設定画面を開くときの挙動を実装する 別途作成したWindowsフォームをインスタンス化し、ShowDialogメソッドで設定画面を開く | |
| | GetSchema | マップパー画面を開くときの挙動を実装する スキーマファイルを読み込むなどの方法で、送り先の項目を一覧表示する | |



2.3 UIクラスの作成

- UIクラスをビルドし、dllを所定のフォルダに配置します。

Visual Studio(.NET)によるクラスの開発例

The screenshot shows the Visual Studio environment with the 'SQLCommanderUI' project. The 'SQLCommanderUI.dll' is being built. A callout box highlights the deployment path: **【UIクラスdll配置フォルダ】**
<xpiインストールフォルダ>
└ Runtime
└─┬─ addon_connectors
└─┬─ <コネクタ名>
└─┬─ ui
└─┬─ lib

The screenshot also shows the 'lib' folder in the file explorer, where the 'SQLCommanderUI.dll' file is listed with a size of 23 KB and a date of 2019/10/17 15:30.

| 名前 | 更新日時 | 種類 | サイズ |
|--------------------|------------------|------------|-------|
| SQLCommanderUI.dll | 2019/10/17 15:30 | アプリケーション拡張 | 23 KB |



OUTPERFORM THE FUTURE™

2.4

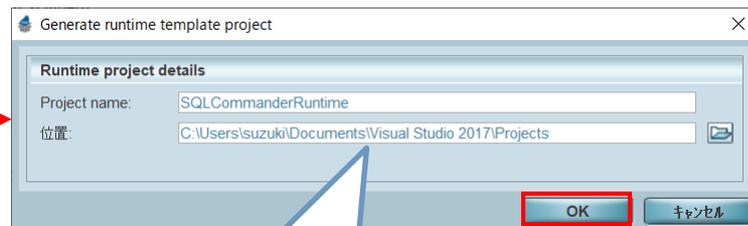
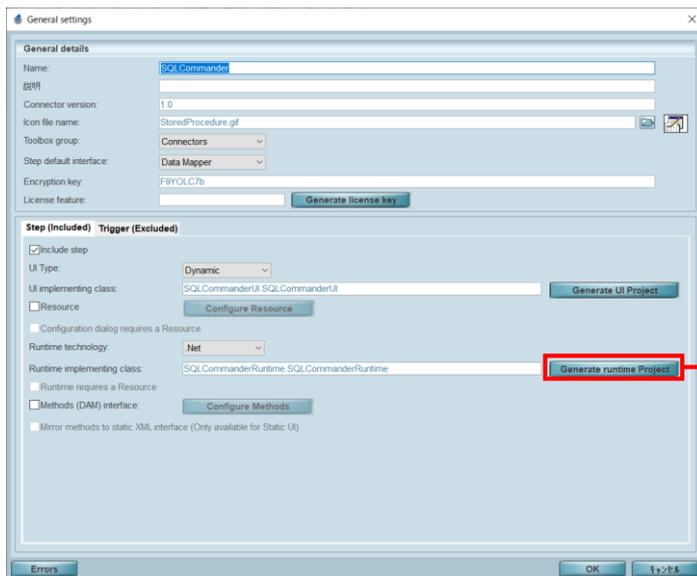
Runtimeクラスの作成



OUTPERFORM THE FUTURE™

2.4 Runtimeクラスの作成

- 「Generate runtime Project」ボタンを押し、Runtimeクラスのひな型を作成します。



プロジェクト名と
プロジェクト格納フォルダを
指定し「OK」ボタン押下



2.4 Runtimeクラスの作成

- ・ サンプルコードつきの下記のクラスやメソッドが作成されますので、コードを修正します。
- ・ 必要に応じて他のクラスを作成し、オブジェクト指向でコーディングすることができます。

| クラス | メソッド | 役割 |
|------------------------|---------|-------------------------|
| コネクタビルダで入力したRuntimeクラス | コンストラクタ | コネクタが実行されたときの動作を定義するクラス |
| | invoke | コネクタが実行されたときの動作を実装する |



2.4 Runtimeクラスの作成

- Runtimeクラスをビルドし、dllを所定のフォルダに配置します。

Visual Studio(.NET)によるクラスの開発例

The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays the source code for `SQLCommanderRuntime.cs`. The code includes namespace declarations for `System`, `MagicSoftware`, and `System.Xml`, and defines a `SQLCommanderRuntime` class implementing the `IStep` interface. The `invoke` method contains logic to create a stored procedure engine, load XML, and process it.
- Output Window:** Shows the build output, indicating a successful build of `SQLCommanderRuntime.dll`.
- File Explorer:** Shows the deployment path: `lib > Runtime > addon_connectors > SQLCommander > runtime > dotnet > lib`. The `SQLCommanderRuntime.dll` file is visible in the `lib` folder.

【Runtimeクラスdll配置フォルダ】
 <xpiインストールフォルダ>
└─ Runtime
└─ addon_connectors
└─ <コネクタ名>
└─ runtime
└─ dotnet
└─ lib

2.4 Runtimeクラスの作成

- Runtimeクラスの開発には.NET、Java、Magic xpaのいずれかを使用することができます。
- 使用する言語に応じて、ビルドで作成されるファイルや配置フォルダが異なります。

| 開発言語 | ファイル | 配置フォルダ |
|-----------|------|---|
| .NET | dll | <xpiインストールフォルダ>¥Runtime¥addon_connectors¥<コネクタ名>¥runtime¥dotnet¥lib |
| Java | jar | <xpiインストールフォルダ>¥Runtime¥addon_connectors¥<コネクタ名>¥runtime¥java¥lib |
| Magic xpa | ecf | <xpiインストールフォルダ>¥Runtime¥addon_connectors¥<コネクタ名>¥runtime¥magic¥lib |



2.5 Magic xpiスタジオにおけるコネクタの使用

- xpiスタジオを開き、作成したコネクタをフローエリアにドラッグ&ドロップします。
- コネクタをダブルクリックすると、Windowsフォームで作成した設定画面が開きます。



2.5 Magic xpiスタジオにおけるコネクタの使用

- 設定画面で設定後、OKボタンを押すとマッパー画面が開きます。
- マッパー画面では任意の送り元から、または計算値を使用して、送り先の値を設定します。

