

ユーティリティ起動コンポーネントサンプルアプリケーション

1. 概要

ユーティリティ起動コンポーネントは、オブジェクトのクラス名・パッケージ名の取得や型変換、ガベージコレクタの起動など、少し高度な操作を行うときに必要なメソッドを提供するコンポーネントです。ユーティリティ起動コンポーネントは、アプリケーションビルダーのメニューから以下のように選びます。

[コンポーネント追加]-[処理部品]-[ユーティリティ]-[ユーティリティ起動]

2. 用途

- アプリケーション開発中、動作確認のために中間データの情報を取得したいとき。
- アプリケーション実行中、強制的にガベージコレクタを起動したいとき。
- アプリケーション実行中のメモリ消費量を確認したいとき。
- オブジェクトのファイル保存・読込を行いたいとき。
- 各種メッセージ、ダイアログを表示したいとき。

3. ここで使用されるイベントとメソッド

ここで使用するユーティリティ起動コンポーネントのメソッドを表 1 に示します。なお、このコンポーネントから発生するイベントはありません。

表 1 ここで使用するユーティリティ起動コンポーネントのメソッド

使用されるメソッド	処理内容
Save 版ファイル選択ダイアログの表示(Component)	ファイル保存用のダイアログを表示します。このダイアログは、指定したコンポーネントの上に表示されます。
Open 版ファイル選択ダイアログの表示(Component)	ファイルを開くためのダイアログを表示します。このダイアログは、指定したコンポーネントの上に表示されます。
saveData(Object, File)	第 1 引数で指定したオブジェクトを、第 2 引数で指定したファイルに保存します。
loadData(File)	指定したファイルからオブジェクトをロードします。

4. コンポーネント使用例

付属のサンプルアプリケーションを使って、ユーティリティ起動コンポーネントの使い方を説明します。アプリケーションビルダーを起動し、インストールフォルダ以下にある“AP_DATA¥Sample¥ユーティリティ起動.mzax”をロードしてください。

4.1. 動作確認

最初に、サンプルアプリケーションの動作確認を行います。アプリケーションビルダーの[実行 (設定可)]ボタンをクリックして、サンプルアプリケーションを起動してください。ウィンドウが表示されたら、まず、テーブルデータを作成します。テーブル上で右クリックしたときに表示されるポップアップメニューから、列と行を追加します。図 1 で示しているのは、1行3列の文字列型データのテーブルを作成した例です。

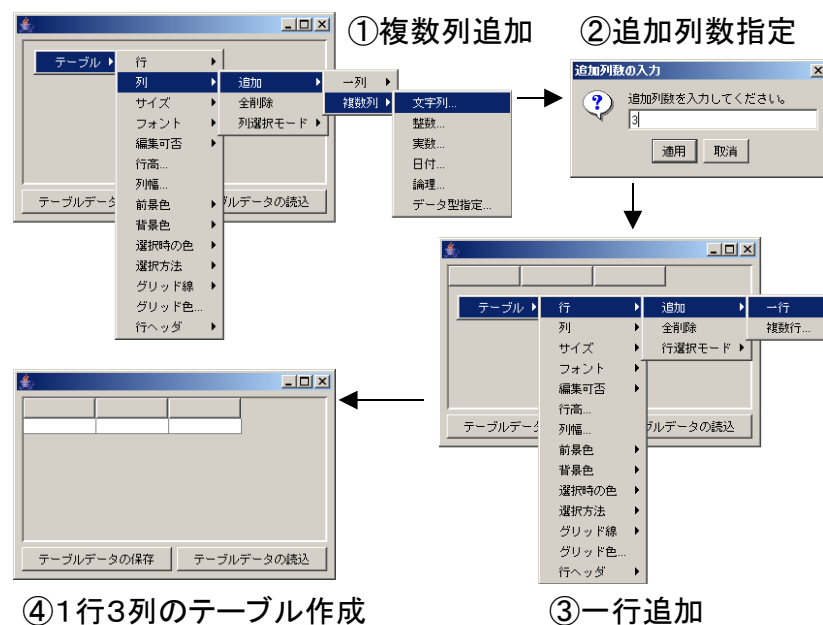


図 1 テーブルの作成 (1行3列文字列型)

このテーブルにデータを記入し、[テーブルデータの保存]ボタンをクリックします。保存用のファイル選択ダイアログが表示されますので、保存先のファイル名を指定して[保存]ボタンをクリックします。下図は、テーブルのデータとして「a」、「b」、「c」を記入し、tab.dat というファイルに保存した例を示しています。

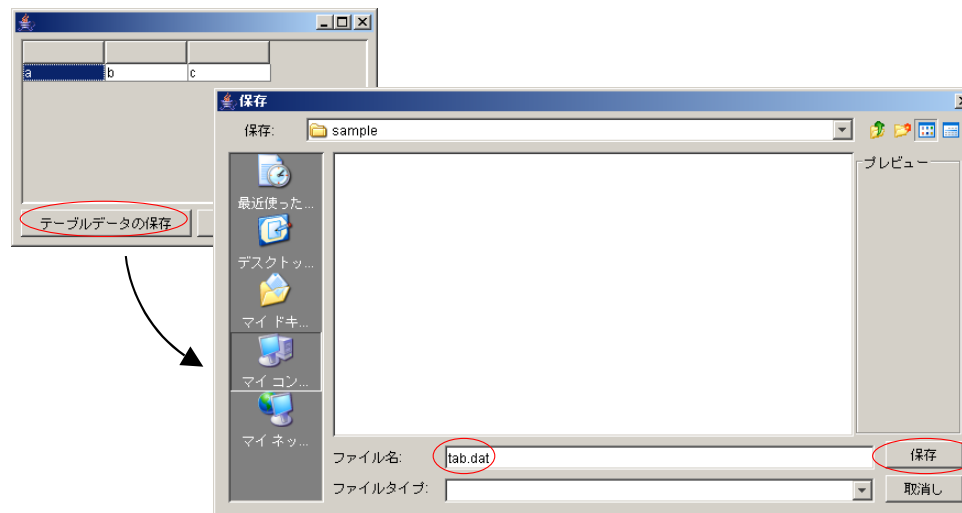


図 2 テーブルデータの保存

今度は、保存したファイルを読み込んでみて、テーブルデータがファイルに保存されていることを確かめます。テーブル上で右クリックし、表示されたポップアップメニューから[テーブル]-[行]-[全削除]と選択して、一旦、テーブルのデータを削除します（図 3）。

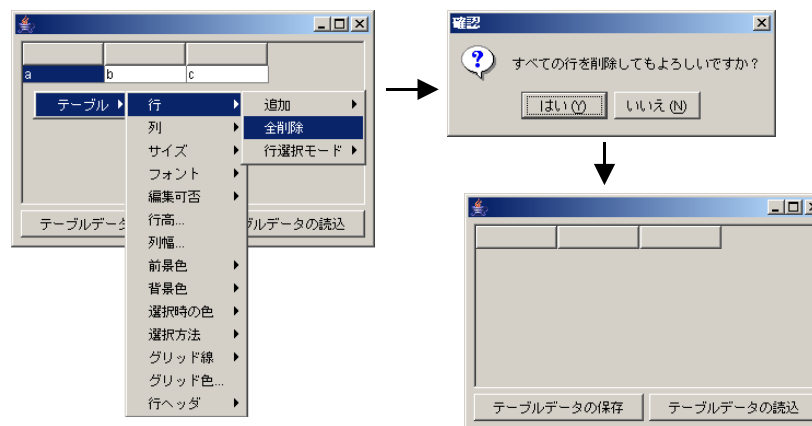


図 3 テーブルデータの削除

[テーブルデータの読込]ボタンをクリックし、表示されたファイル選択ダイアログから先程保存したファイルを選んで[開く]ボタンをクリックすると、テ

ーブルのデータが設定されます (図 4)。

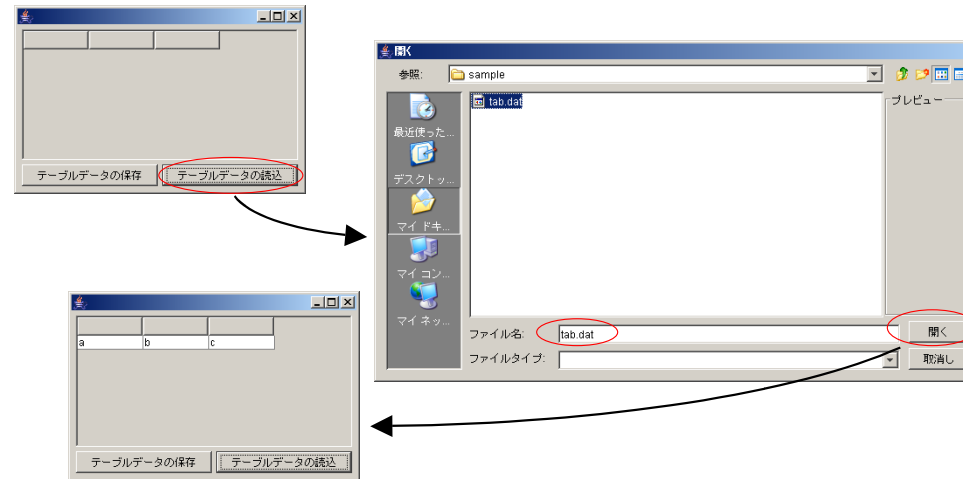


図 4 テーブルデータの読み込み

4.2. コンポーネント接続の確認

では、このような動作がどのように実現されているのか、アプリケーションビルダーに表示されているコンポーネントの接続図をたどってみましょう。

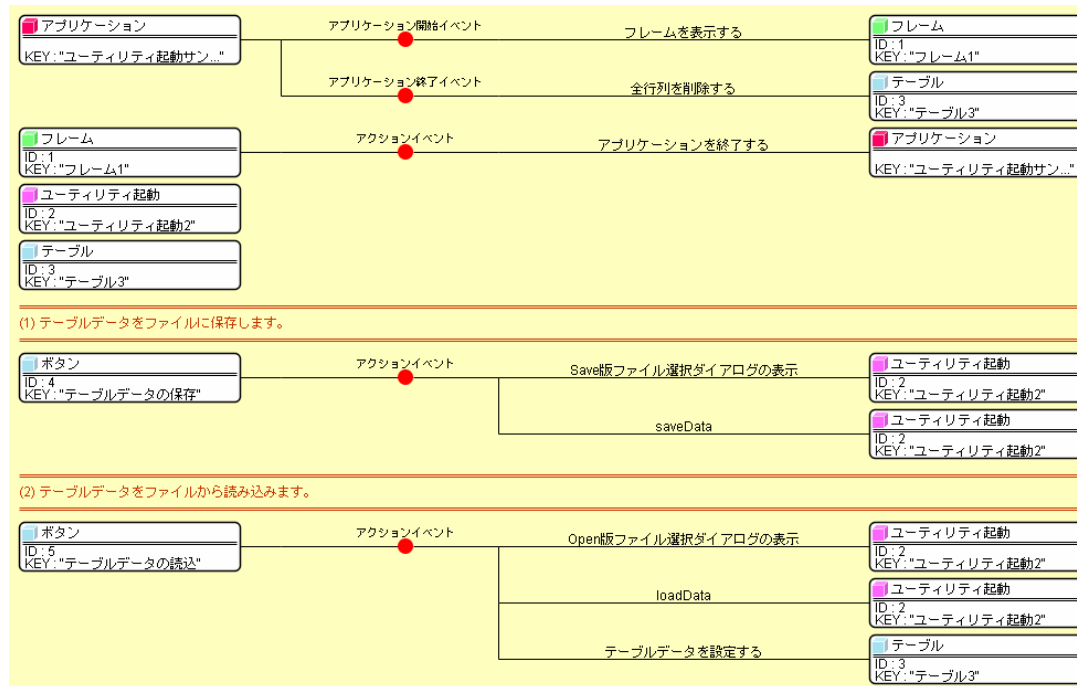


図 5 サンプルアプリケーションコンポーネント接続図

ボタン（テーブルデータの保存）をクリックするとアクションイベントが発生し、ユーティリティ起動（ユーティリティ起動2）の「Save 版ファイル選択ダイアログの表示()」メソッドが呼び出されます。このメソッドの処理結果（戻り値）は、ダイアログで選択された File オブジェクトです。

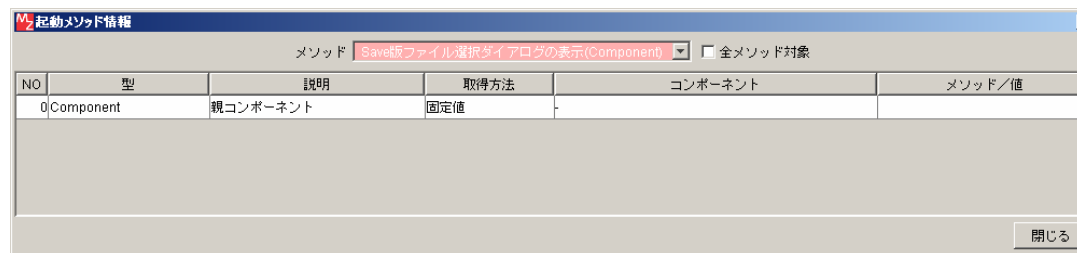
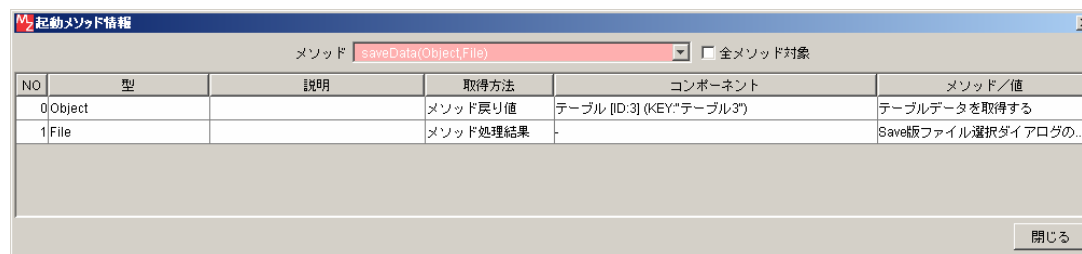


図 6 ユーティリティ起動の「Save 版ファイル選択ダイアログの表示()」メソッド呼び出し

そして、ユーティリティ起動（ユーティリティ起動2）の saveData()メソッドが呼び出されます。このメソッドの引数は、保存するデータ（オブジェクト

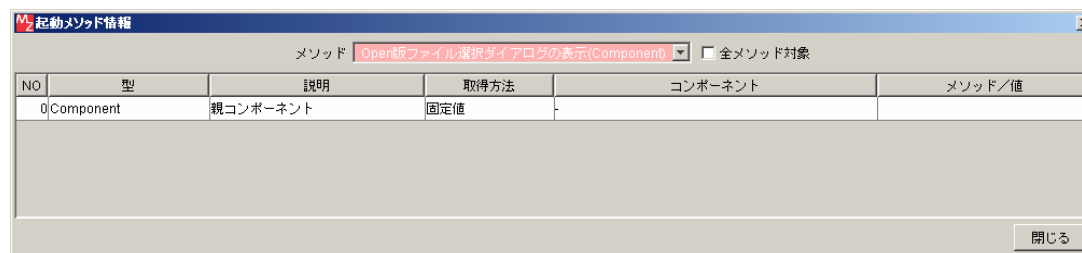
ト) と保存先のファイル (File オブジェクト) です。ここでは、保存するデータとしてテーブル (テーブル 3) から取得したテーブルデータ、保存先のファイルとして、先に呼び出した「Save 版ファイル選択ダイアログの表示0」メソッドの処理結果 (戻り値) を指定しています。



NO	型	説明	取得方法	コンポーネント	メソッド/値
0	Object		メソッド戻り値	テーブル [ID:3] (KEY:"テーブル3")	テーブルデータを取得する
1	File		メソッド処理結果	-	Save版ファイル選択ダイアログの...

図 7 ユーティリティ起動の saveData()メソッド呼び出し

次に、ボタン (テーブルデータの読込) をクリックしたときの動作を見てみましょう。このボタンをクリックするとアクションイベントが発生し、ユーティリティ起動 (ユーティリティ起動 2) の「Open 版ファイル選択ダイアログの表示0」メソッドが呼び出されます。このメソッドの処理結果 (戻り値) は、ダイアログで選択された File オブジェクトです。



NO	型	説明	取得方法	コンポーネント	メソッド/値
0	Component	親コンポーネント	固定値	-	

図 8 ユーティリティ起動の「Open 版ファイル選択ダイアログの表示0」メソッド呼び出し

そして、ユーティリティ起動 (ユーティリティ起動 2) の loadData()メソッドが呼び出されます。このメソッドの引数は読込先のファイル (File オブジェクト)、処理結果 (戻り値) は読み込んだデータ (オブジェクト) です。ここでは、読込先のファイルとして、先に呼び出した「Open 版ファイル選択ダイアログの表示0」メソッドの処理結果 (戻り値) を指定しています。

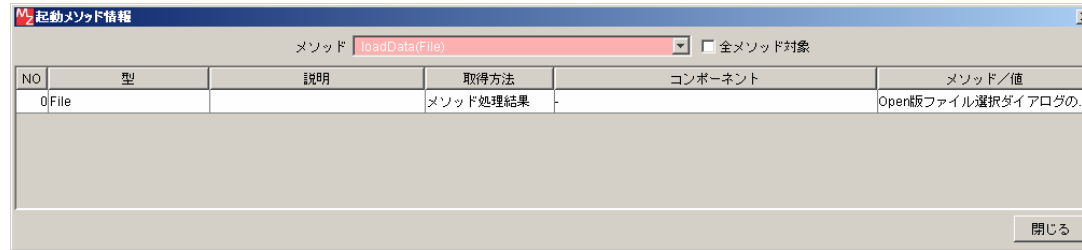


図 9 ユーティリティ起動の loadData()メソッド呼び出し

最後に、テーブル（テーブル 3）の「テーブルデータを設定する0」メソッドを呼び出して、読み込んだデータをテーブルに設定します。このメソッドの引数には、先に呼び出した loadData()の処理結果（戻り値）を指定しています。

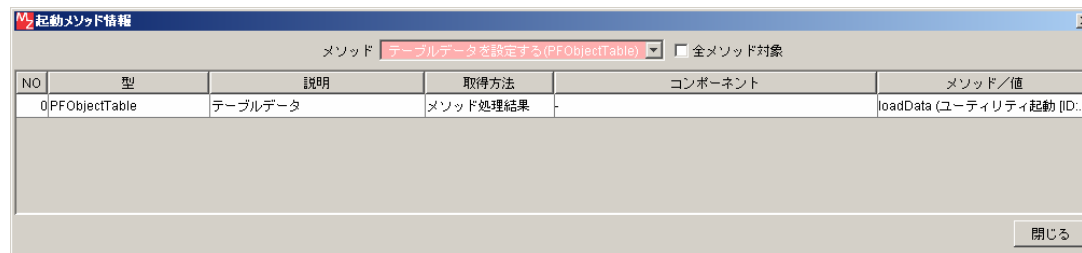


図 10 テーブルの「テーブルデータを設定する0」メソッド呼び出し

4.3. ダイアログコンポーネントとの使い分け

このサンプルアプリケーションでは、ファイル選択ダイアログを表示するためにユーティリティ起動コンポーネントを使用しました。一方、ファイル選択ダイアログは単独のコンポーネントとしても提供されており、メニューから

[コンポーネント追加]-[画面構成部品]-[ダイアログ]-[ファイル選択]

と選ぶことによって利用することができます。同様に、他のメッセージダイアログも単独のコンポーネントとして用意されています。ダイアログコンポーネントとユーティリティ起動コンポーネントのダイアログ表示メソッドとは、どのように使い分ければよいのでしょうか。

これらの間の大きな違いは、発生するイベントにあります。ユーティリティ起動コンポーネントはイベントを発生しませんが、ダイアログコンポーネントの場合、クリックしたボタンによって番号の異なるイベントが発生します。例えば、ファイル選択ダイアログの場合、キャンセルボタンをクリックするとイベント番号 0、showOpenSingleFile()と showSaveSingleFile()で OK ボタンをクリックしたときにはイベント番号 1、showOpenFile(), showSaveFile()で OK ボタンをクリックしたときにはイベント番号 2 のデータ選択イベントが発生します。これによって、OK ボタンをクリックしたときには処理を行うがキャンセルボタンをクリックしたときには何もしないなど、細かい設定を行うことができます。ユーティリティ起動コンポーネントでは、そのような設定

を行うことはできません。

実際、このサンプルアプリケーションでテーブルデータを読み込むときに[取消しボタン]をクリックすると下図のようなエラーメッセージが表示されます。

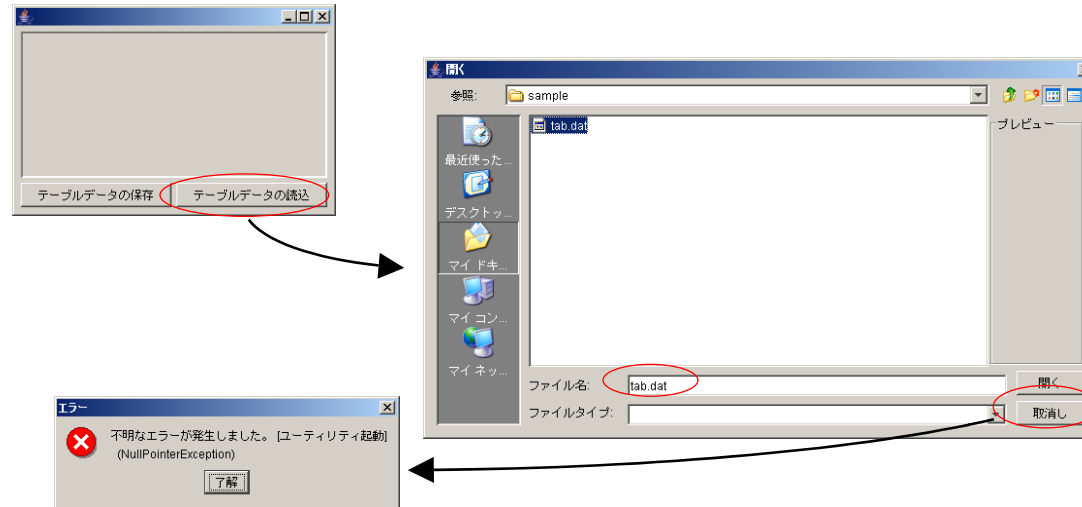


図 11 [取消し]ボタンクリックによって表示されるエラーメッセージ

これは、[取消し]ボタンのクリックによってファイルの読み込みが行われなかったにも関わらず、後の処理を実行してしまったために生じたエラーです。

したがって、クリックしたボタンによって処理を変えたいときにはダイアログコンポーネント、単にメッセージを表示するだけなど、特に処理を変える必要がない場合にはユーティリティ起動コンポーネントを使うというのが、1つの指針だと言えるでしょう。