変数コンポーネントとオブジェクトバッファコンポーネント

1. 概要

アプリケーションを実行すると、コンポーネントの間で様々なデータ（オブジェクト）がやりとりされます。それらのデータは、イベント内包データやメソッド戻り値といった形で取得され、各コンポーネントのメソッド引数として使われることが普通です。しかしながら、時には、やりとりされるデータそのものに対して編集などの操作を加えることが必要になります。

例えば、「ようこそ XX さん」というメッセージを表示することを考えましょう。XX の部分には人の名前が入るとします。このようなときは、あらかじめ ”ようこそ XX さん” という文字列（String オブジェクト）を用意しておき、必要に応じて XX の部分を人の名前を示す文字列で置き換えるのが便利です。この場合、String クラスには replaceFirst()という文字列を置き換えるためのメソッドが提供されていますので、

(1) “ようこそ XX さん” というデータを持つ String オブジェクトの設定
(2) (1)で設定したオブジェクトの replaceFirst()メソッド呼び出し

という手順で上述の機能が実現されます。このような、「オブジェクトの設定とそのメソッド呼び出し」を行うためのコンポーネントが、変数コンポーネントとオブジェクトバッファコンポーネントです。

変数コンポーネントは、ある特定のクラスのオブジェクトを設定して、そのメソッドを実行するためのコンポーネントです。アプリケーションビルダーのメニューからは、以下のようにして選びます。

[コンポーネント追加]-[処理部品]-[変数]-[（各種変数コンポーネント）]

今のところ、以下のクラスに対応した変数コンポーネントが用意されています。

- 文字列(String)
- 任意精度実数(BigDecimal)
- 任意精度整数(BigInteger)
- 浮動小数点数(Double)
- 浮動小数点数(Float)
- 整数(Long)
- 整数(Integer)

- 整数(Short)
- バイト値(Byte)
- 論理値(Boolean)
- 日付格納変数(Date)
- 画像データ(Image)
- リスト(PFObjectList)
- テーブル(PFObjectTable)
- ツリー(PFObjectTree)
- ツリーノード(PFObjectTreeNode)
- ラベル付きリスト(PFLabeledObjectList)
- 算術演算子コンポーネント(PFArithmeticOperator)

これらの変数コンポーネントは、各クラスの公開メソッドを基本として、アプリケーションで必要となる様々なメソッドを提供しています。詳細は、各コンポーネントの Javadoc（HTML マニュアル）をご覧ください。

一方、オブジェクトバッファコンポーネントは、任意のオブジェクトを格納し、そのメソッド実行やフィールド値の取得を行うためのコンポーネントです。用途は、各種変数コンポーネントとほぼ同じですが、変数コンポーネントがそれぞれ特定のクラスのオブジェクトのみを扱うのに対して、オブジェクトバッファコンポーネントは任意のクラスオブジェクトを扱うことができます。一方、変数コンポーネントにはそれぞれのクラスオブジェクトを操作するための各種メソッドが用意されていますが、オブジェクトバッファコンポーネントにはそのようなメソッドはなく、ユーザは格納したオブジェクトを操作するためのメソッドを文字列で指定しなくてはなりません。すなわち、ユーザは格納したオブジェクトのメソッドやフィールド名をあらかじめ知っておく必要があります。その意味で、オブジェクトバッファコンポーネントは上級者向けのコンポーネントだと言えます。オブジェクトバッファコンポーネントは、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]-[オブジェクト]-[オブジェクトバッファ]

2. 用途

変数コンポーネントとオブジェクトバッファコンポーネントは、それぞれ以下のような場合に使います。

変数コンポーネント

- 特定のクラスのオブジェクトに対して、編集などの操作を行いたいとき。

オブジェクトバッファコンポーネント

- 変数コンポーネントが用意されていないクラスのオブジェクトを操作したいとき。
- オブジェクトのフィールド値を取得したいとき。

3. ここで使用されるイベントとメソッド

ここで使用するイベントとメソッドを表 1～表 3 に示します。なお、ここでは使用していませんが、変数コンポーネントからはデータ設定イベントが発生します。

表 1 オブジェクトバッファコンポーネントから発生するイベント（データ設定イベント）

トリガ	対象データ	イベント番号
setObject()メソッドの呼び出しによるオブジェクトの設定	設定されたオブジェクト	デフォルトは 0。設定可能

表 2 ここで使用する文字列格納変数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
文字列を設定する(String)	文字列格納変数コンポーネントに文字列を設定します。
replaceFirst(String,String)	設定されている文字列の中から第１引数で指定されている文字列を見つけ出し、それを第２引数の文字列で置き換えた文字列を取得します。設定されている文字列そのものは変更されません。

表 3 ここで使用するオブジェクトバッファ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
オブジェクトの設定（イベント番号指定）(Object)	オブジェクトバッファコンポーネントにオブジェクトを設定します。発生するデータ設定イベントのイベント番号を指定します。
フィールドの選択(String)	設定されているオブジェクトのフィールドを名前指定で選択します。
選択したフィールド値の取得()	設定されているオブジェクトの選択したフィールド値を取得します。
オブジェクトのクラス名取得()	設定されているオブジェクトのクラス名をフルパスで取得します。
メソッド起動（１引数オブジェクト指定）(String,Object)	設定されているオブジェクトの１引数のメソッドを起動します。第１引数がメソッド名、第２引数がそのメソッドの引数になります。

4. コンポーネント使用例

4.1. 変数コンポーネント

それでは、文字列格納変数コンポーネントを例として、変数コンポーネントの使い方を見ていくことにしましょう。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¥変数とオブジェクトバッファ_1.mzax”をロードしてください。

4.1.1. 動作確認

[実行]もしくは[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。このアプリケーションは、「ようこそ XX さん」というメッセージを表示するものです。[メッセージの表示]ボタンをクリックすると、XX の部分をテキストフィールドに記入した文字列で置き換えたメッセージが表示されます。

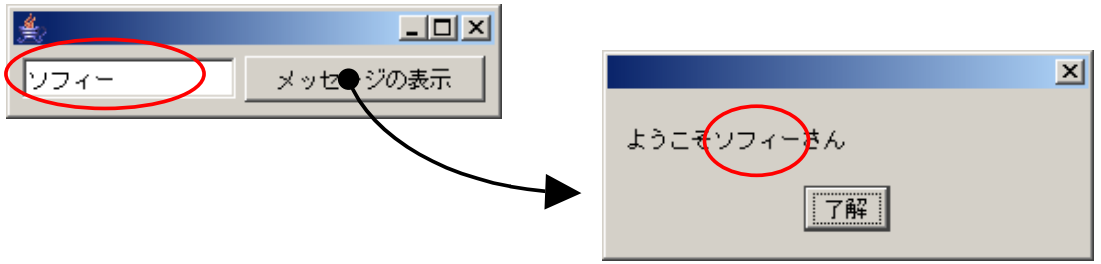


図 1 「ようこそ」メッセージの表示

図に示した例では「ソフィー」という名前を入れていますが、「ジョナサン」と入れれば「ようこそジョナサンさん」というメッセージが、「地球」と入れれば「ようこそ地球さん」というメッセージが表示されます。

4.1.2. コンポーネント接続の確認

では、この動作がどのように実現されているのか、ビルダー上のコンポーネント接続図をたどってみることにしましょう。

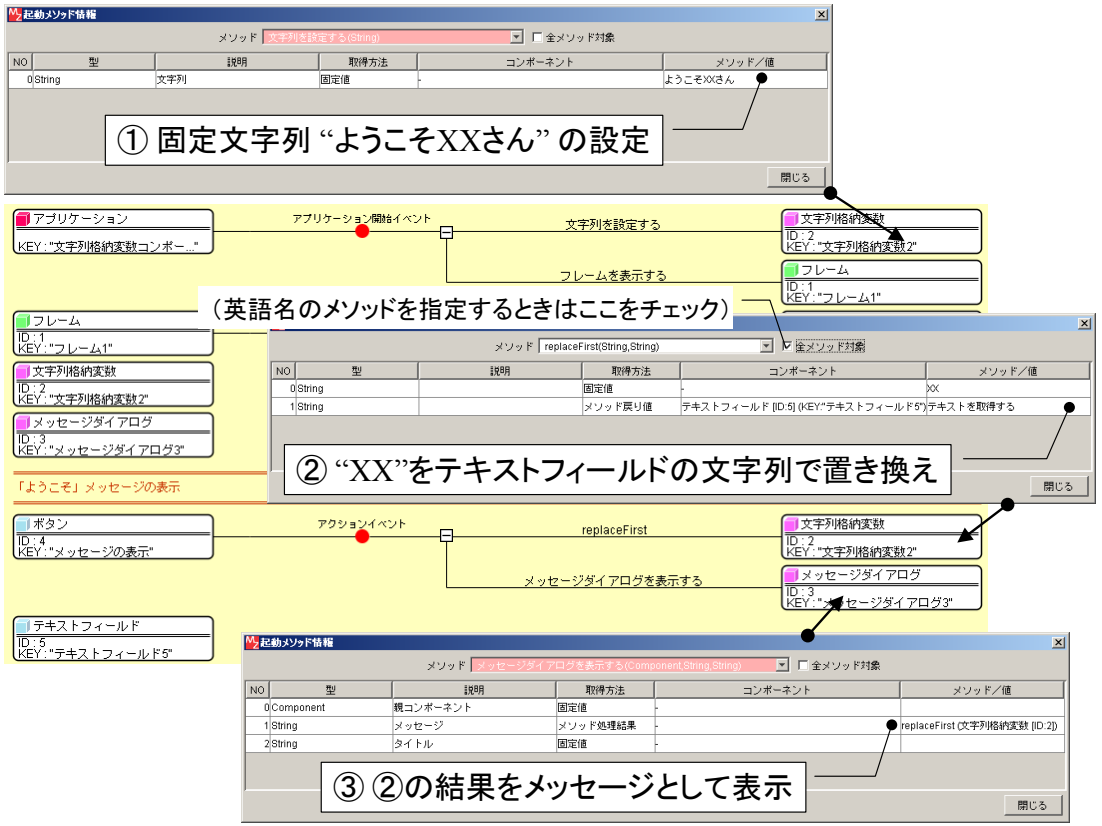


図 2 変数とオブジェクトバッファ_1.mzax のコンポーネント接続図

アプリケーションを起動したとき、最初に文字列格納変数（文字列格納変数２）に対して、固定文字列 “ようこそ XX さん” を設定します（図中①）。ボタン（メッセージの表示）をクリックすると、ボタンからアクションイベントが発生し、文字列格納変数（文字列格納変数２）の replaceFirst()メソッドが呼び出されます。このメソッドの第 1 引数は置き換えの対象になる文字列で、第 2 引数はその部分へ代入される文字列です。また、戻り値は置き換えられた結果を示す文字列です。この場合、第 1 引数は “XX”、第 2 引数はテキストフィールドの文字列になります（図中②）。なお、replaceFirst()メソッド呼出し後も、文字列格納変数（文字列格納変数２）に設定されている文字列自身は変化しません。最後に、このメソッドの戻り値をメッセージ表示します（図中③）。

変数コンポーネントを使用すると、このようなオブジェクトの格納やオブジェクトに対する操作を行うことができます。ここでは例として文字列格納変数コンポーネントを取り上げましたが、このほかにも、リスト格納変数コンポーネントを用いるとリストに対する要素の追加、削除、検索などが行えます。

また、テーブル格納変数コンポーネントを使うと、テーブルに対する行や列の追加および削除、セルデータの設定や取得などを行うことができます。

4.1.3. 変数コンポーネントの作成（上級者向け）

自分自身で独自のクラスを作成した場合など、それらのオブジェクトを操作するために新しく変数コンポーネントを作成する必要が生じた場合には、付属のサンプルアプリケーション“VariableImplementerSample.mzax”を利用します。これを使うと、設定したオブジェクトの公開メソッドを呼び出すためのプログラム（ソースコード）が自動生成されます。上位の抽象クラスである PFVariable を継承して必要な編集を施せば、そのクラスを対象とした変数コンポーネントとなります。なお、コンポーネント作成方法の詳細につきましては、付属のコンポーネント開発ガイドをご覧ください。コンポーネント開発ガイドは、developer¥manual フォルダに収録されています。

では、VariableImplementerSample.mzax の動作を見てみましょう。このアプリケーションは、AP_DATA¥Sample に収録されています。下図は、文字列格納変数のソースコード生成を例として、その操作手順を示したものです。

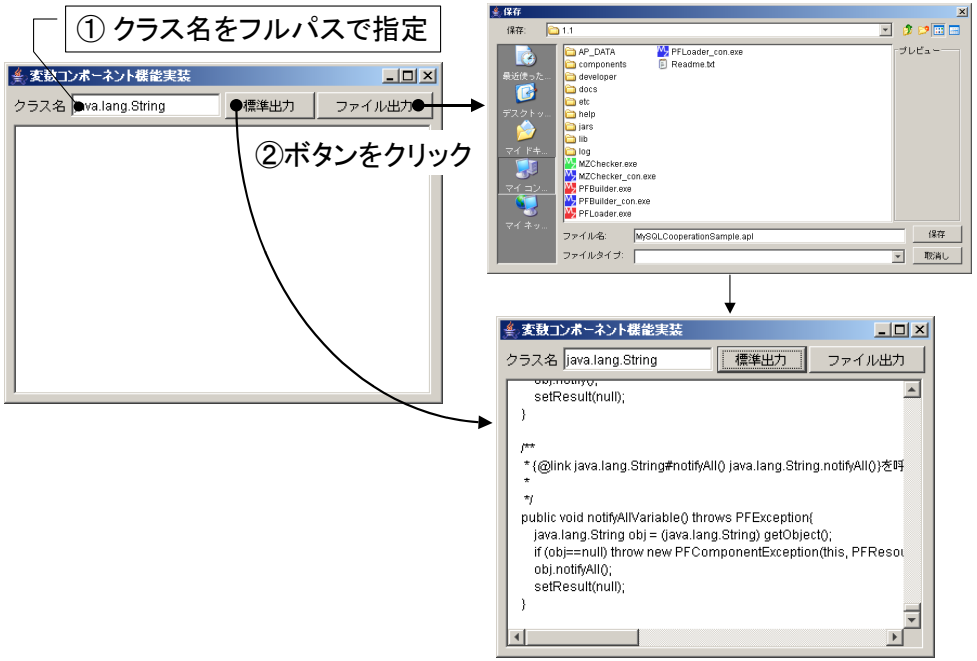


図 3 変数コンポーネントの機能実装

アプリケーションのテキストフィールドに、対象とするクラスのフルパス名を記入します。そして、[標準出力]ボタンもしくは[ファイル出力]ボタンをクリックするとソースコードが生成されます。[ファイル出力]ボタンをクリックした場合にはファイル選択ダイアログが表示されますので、出力先ファイルを

指定します。

4.2. オブジェクトバッファコンポーネント（上級者向け）

オブジェクトバッファコンポーネントの使用例を示します。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¥変数とオブジェクトバッファ_2.mzax”をロードしてください。

4.2.1. 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]もしくは[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。このアプリケーションは、テーブルの水平スクロールバーの表示と非表示の切り替えを行います。画面のチェックボックスに対して、チェックを外す、あるいはチェックを入れるといった操作を行ってください。下図のように、メッセージダイアログが表示され、水平スクロールバーの表示／非表示が切り替わります。

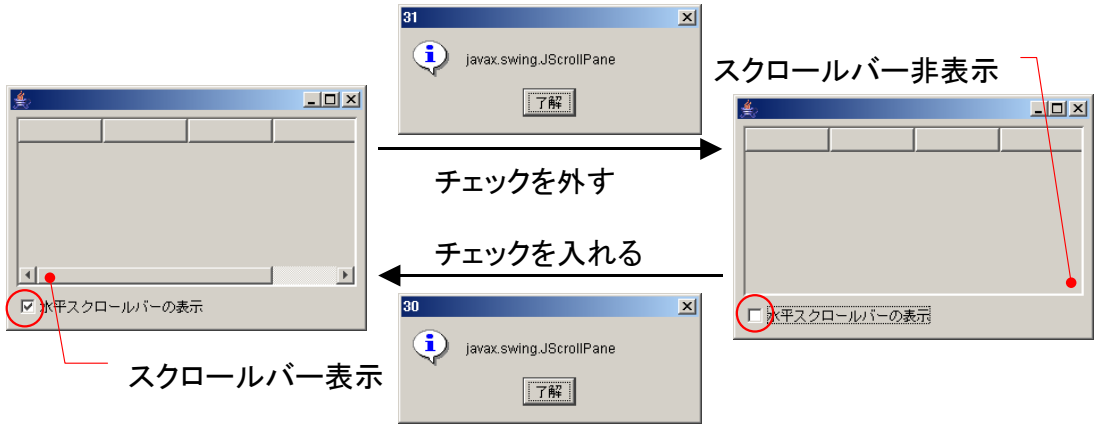


図 4 水平スクロールバーの表示／非表示切り替え

4.2.2. 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。

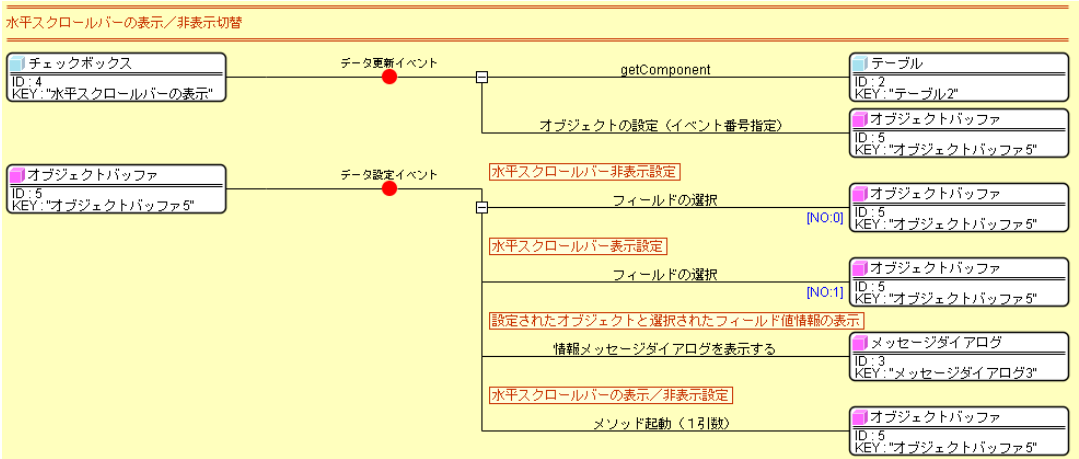


図 5 変数とオブジェクトバッファ_2.mzax のコンポーネント接続図

MZ Platform のテーブルコンポーネントは、水平・垂直スクロールバーを、必要なときは表示し、不要なときは表示しないようになっています。その性質を変更するには少し高度な Java の知識が必要で、以下の手順を取ります。

(1) テーブルコンポーネントを構成している Java Bean（オブジェクト）の中から、スクロールバーの表示属性を設定しているものを取得する。
(2) (1)で取得したオブジェクトを操作して、スクロールバーの表示／非表示を切り替える。

このサンプルアプリケーションでは、上述の操作を、チェックボックスのチェック切替が行われたときに実行しています。
チェックボックスのチェックが切り替わるとデータ更新イベントが発生し、まず、テーブル（テーブル２）の getComponent()メソッドが呼び出されます。

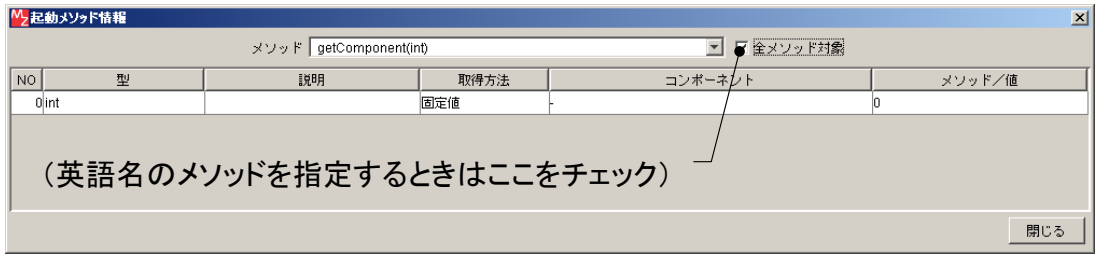


図 6 テーブルの getComponent()メソッド呼び出し

このメソッドは java.awt.Container クラスから継承したメソッドで、テーブルコンポーネントを構成するオブジェクトをインデックス指定で取得します。ちなみに、同じく java.awt.Container から継承したメソッドに getComponentCount()というものがあり、これを使うとコンポーネントを構成するオブジェクトの数がわかりますので、0 から始まるすべてのインデックス番号について getComponent()を呼び出せば、テーブルコンポーネントを構成するすべてのオブジェクトを取得できます。

ここで、インデックス番号 0 で取得したのは、javax.swing.JScrollPane オブジェクトです。このオブジェクトの setHorizontalScrollBarPolicy(int)を実行すると、水平スクロールバーの表示／非表示の設定を行うことができます。引数として与える整数は、以下に示す javax.swing.JScrollPane クラスのフィールド値です。

スクロールバーを必要なときのみ表示: HORIZONTAL_SCROLLBAR_AS_NEEDED
スクロールバーを非表示: HORIZONTAL_SCROLLBAR_NEVER

したがって、水平スクロールバーの表示／非表示を設定する手順は以下のようになります。

(1) オブジェクトバッファに対して、取得した javax.JScrollPane オブジェクトを設定する。
(2) スクロールバーを表示する場合は HORIZONTAL_SCROLLBAR_AS_NEEDED のフィールド値を、非表示とする場合には HORIZONTAL_SCROLLBAR_NEVER のフィールド値を取得する。
(3) (2)で取得したフィールド値を引数として、setHorizontalScrollBarPolicy()メソッドを実行する。

それでは、コンポーネント接続図で、以上の処理の流れを追ってみましょう。テーブル（テーブル２）の getComponent()メソッドで取得した javax.JScrollPane オブジェクトをオブジェクトバッファ（オブジェクトバッファ５）にイベント番号指定で設定します（図 7)。

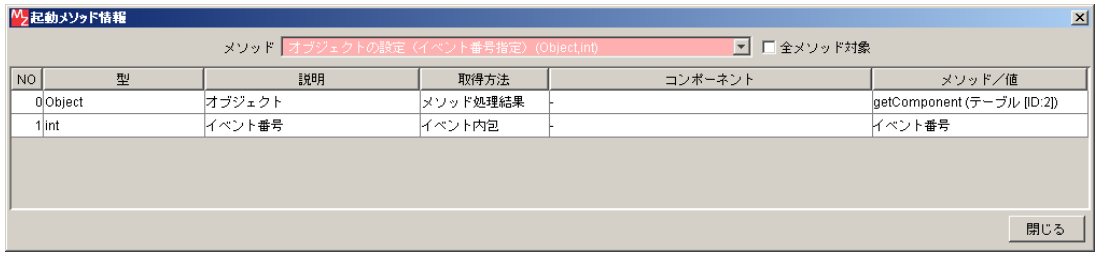


図 7 オブジェクトバッファへのオブジェクト設定（イベント番号指定）

指定するイベント番号は、チェックボックス（チェックボックス４）から発生したデータ更新イベントの番号で、非選択状態になったときには 0、選択

状態になったときには 1 となります（下図）。

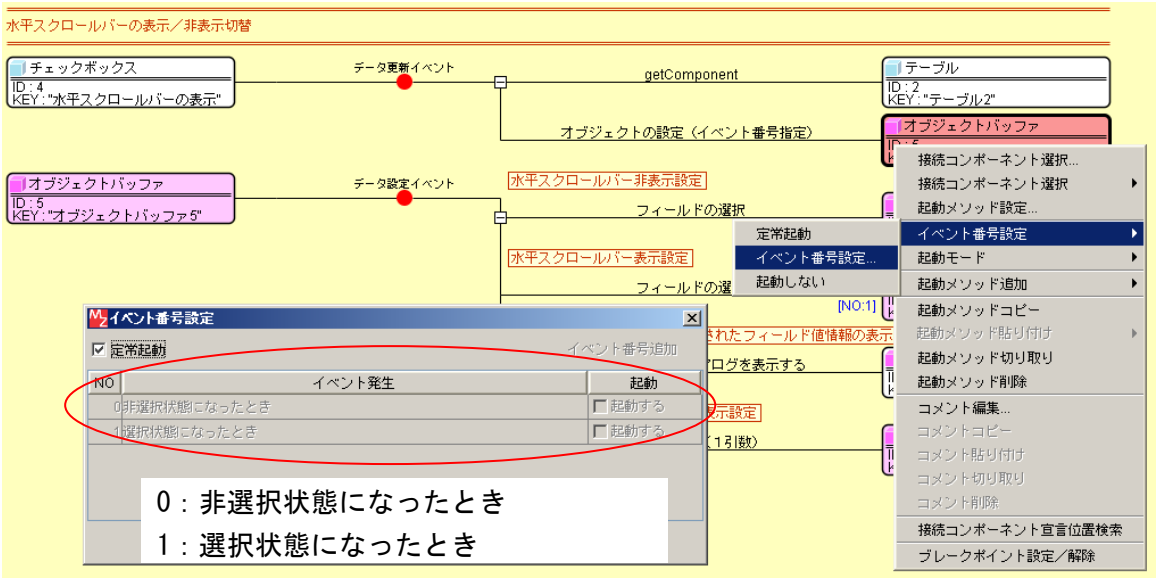
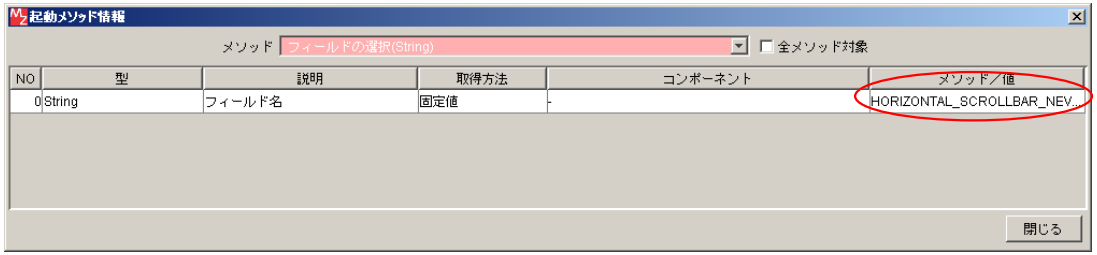


図 8 チェックボックスのデータ更新イベント番号

オブジェクトバッファ（オブジェクトバッファ５）にオブジェクトを設定するとデータ設定イベントが発生します。このとき、イベント番号によって処理が異なります。イベント番号が 0 の場合はスクロールバー非表示としますので、HORIZONTAL_SCROLLBAR_NEVER フィールドを選択します。一方、1 の場合にはスクロールバーを表示しますので、HORIZONTAL_SCROLLBAR_AS_NEEDED フィールドを選択します（図 9）。



(a) イベント番号0の場合

(b) イベント番号1の場合

図 9 イベント番号による選択フィールドの違い

この先の処理はイベント番号によらず同じです。ここで情報メッセージダイアログを表示していますが、これはスクロールバーの表示／非表示切替という処理には関係ありません。このメッセージダイアログは、オブジェクトバッファに設定されているオブジェクトのクラス名と、選択されたフィールド値を表示します。

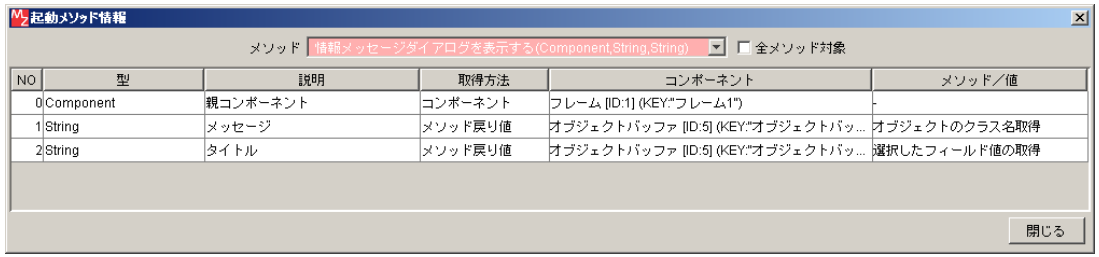


図 10 情報メッセージダイアログの表示

最後に、javax.swing.JScrollPane オブジェクトの setHorizontalScrollBarPolicy()を実行すれば、水平スクロールバーの表示／非表示の設定が完了しま

す。

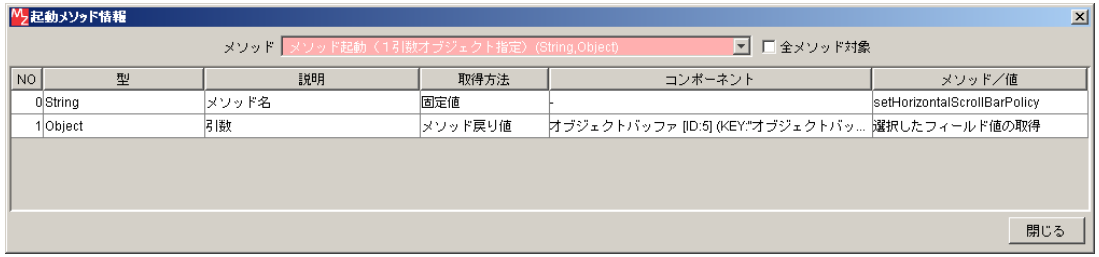


図 11 水平スクロールバー表示／非表示設定