

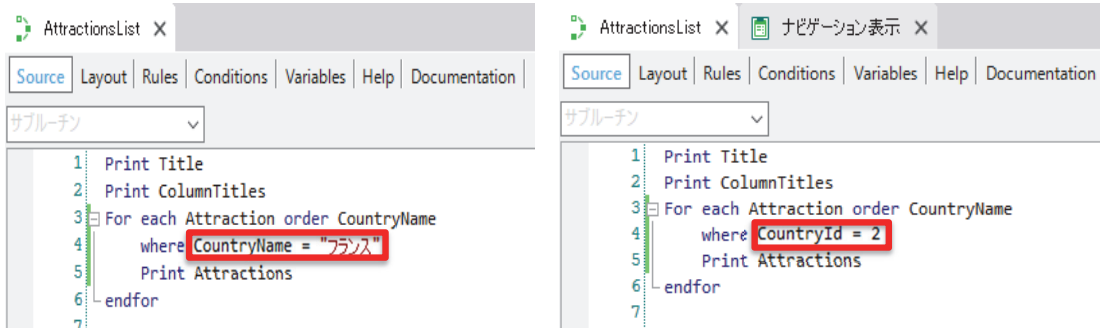
# オブジェクト間の通信

オブジェクトから別のオブジェクトを  
呼び出す必要がある場合

*GeneXus*<sup>™</sup>

## これまでの研修では

### フィルタで定数を使用（この例では「国」）



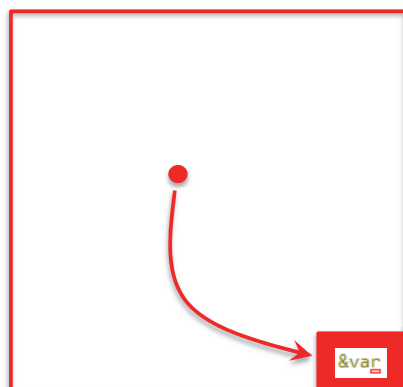
一般的なリストを作成してフィルタに使用する国を  
「受け取る」にはどうすればよいか？

これまでの学習では、あるオブジェクトから別のオブジェクトを呼び出さなければならぬ課題も含まれていました。

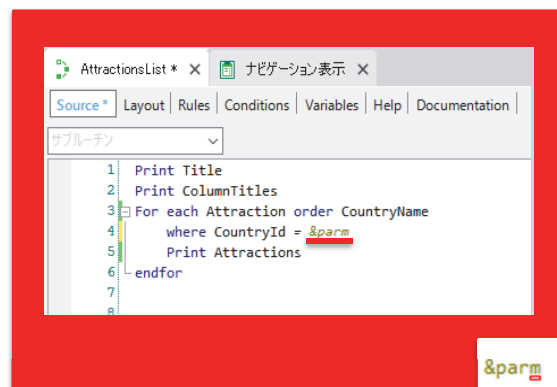
たとえば、AttractionsList プロシージャオブジェクトを実装した際、国名が「フランス」、または国番号が「2」（「フランス」に対応）である観光名所をフィルタする必要がありました。これを実現するために、コード内で定数を使用しました。

しかし、これでは「フランス」以外の国の観光名所をフィルタするには、毎回プロシージャのコードを変更しなければなりません。

呼び出し元



呼び出し先: AttractionsList



フィルタに使用する値をこのオブジェクト内で「受け取る」ことができるのが理想的です。つまり、別の GeneXus オブジェクトでユーザーが値を選択すると、このプロセージャーオブジェクトにその値が送信され、受け取った国に応じて観光名所のリストが表示されるようにします。

それでは、この例を使用して、GeneXus オブジェクト間の通信を実装する方法を見ていきましょう。

## オブジェクトの呼び出し元の実装

### 1) 対象の国を指定するための Web パネルを作成する。

ダイナミック コンボ ボックス タイプのコントロールを使用した変数で、データベースにある国をユーザーに表示する。

2) AttractionsList プロシーチャーを読み出すボタンを追加する。

ListAttractionsByCountry イベントが関連付けられたボタン

Web パネルフォームで指定された国を保持する変数

Event '国の観光名所リスト'  
AttractionsList(&CountryId)  
Endevent

名前	タイプ	Is
&Variables		
Standard Variables		
CountryId	Attribute:CountryId	

最初に、ユーザーが値を指定すると、その値に応じて処理を行う画面のオブジェクトを作成する必要があります。これを実現するオブジェクトは、**Web パネル**です。この Web パネルはこのコースの後の章で説明します。ここでは、さまざまな用途に柔軟に対応できる画面用のオブジェクトと考えてください。ユーザーにデータの入力を要求したり、データベースやその他のソースなどから取得した情報を表示したりすることができます。一例として、観光名所の Work With 一覧画面は、GeneXus により Web パネルとして自動生成されたものです。

ここでは、このタイプのオブジェクトを作成し、EnterAttractionsFilter という名前にします。Web フォームが作成され、オブジェクトの画面になります。これには単一のテーブルが含まれています。

CountryId 変数を追加します。変数の名前と項目属性の名前が同じであるため、項目属性に基づいて同じデータタイプが使用されます。ここで、項目属性のデータタイプを変更 (例: numeric(10) から numeric(4)) すると、変数は自動的にこの新しい値を使用します。

変数のプロパティを編集します。[Control Type] プロパティの値が **Edit** であることを確認してください。この Web パネルが実行されると、フィールドにユーザーが数値を入力する形になります。データベースに含まれる値を選択肢として表示したり、対応する国を表示したりはしません。コントロールタイプを**ダイナミック コンボ ボックス**に変更します。こうすると、ユーザーはデータベースから取得した一連の値から選択することができます。この [Item Values] プロパティは、CountryId 項目属性です。つまり、Country テーブルが参照され、既存の CountryId がコンボボックスにロードされます。変数には国番号が格納されますが、通常、国番号では詳細が分からないため、ユーザーには変数の [Item Descriptions] プロパティで指定された項目属性の内容を表示させます。ここでは国名が表示されるようにします。コンボボックスであることを示す矢印が表示されます。つまり、実行時には、データベースに格納されている国のリストを含むコンボボックスが表示され、そこから国を選択できます。

ボタンも追加します。ボタンに関連付けるイベントの名前を入力するよう求められます。ここでは、「List Attractions By Country (国別観光名所リスト)」とします。ボタンのテキストは、デフォルトではこの名前になります。このボタンを右クリックして [イベントへ移動] を選択すると、この名前のイベントが作成されており、自動的に [Web Form] エlementから [Events] エlementに変更されます。また、このイベントがトリガーされた場合に実行されるコードを入力する場所にカーソルが表示されます。つまり、ボタンをユーザーが押したときに関連付けられたイベントがトリガーされます。

さらに必要なタスクとして、フィルタに使う国を送信し、観光名所をリストする AttractionsList プロシージャオブジェクトを呼び出す必要があります。

【補足】ボタンを押してこのコードを実行すると、画面に表示されているコンボボックスからユーザーが選択した国番号が &CountryId 変数に設定されます。変数はメモリの一部であり、名前が付けられ、データ項目を一時的に保存するために使用されることを既に確認しました。また、各オブジェクトには変数のセクションがあります。つまり、オブジェクト内で定義された変数は、このオブジェクト内でのみ参照できます。そのため、2 つのオブジェクトが同じ CountryId という名前の変数を持っていますが、2 つの異なる変数として扱われます。

## オブジェクトの呼び出し先の実装

変数とParm ルールの宣言：

変数を宣言し、Parmルールを記述

ソースを変更する：

```

print Title
print ColumnTitles
For each Attraction order CountryName
  where CountryId = 2
  print Attractions
endfor

```

このように変更

```

print Title
print ColumnTitles
For each Attraction order CountryName
  where CountryId = &CountryId
  print Attractions
endfor

```

この例では AttractionsList プロシージャオブジェクトが値を受け取るため、オブジェクトを開いて [Rules] エlementに移動します。ここに示す Parm ルールを入力します。

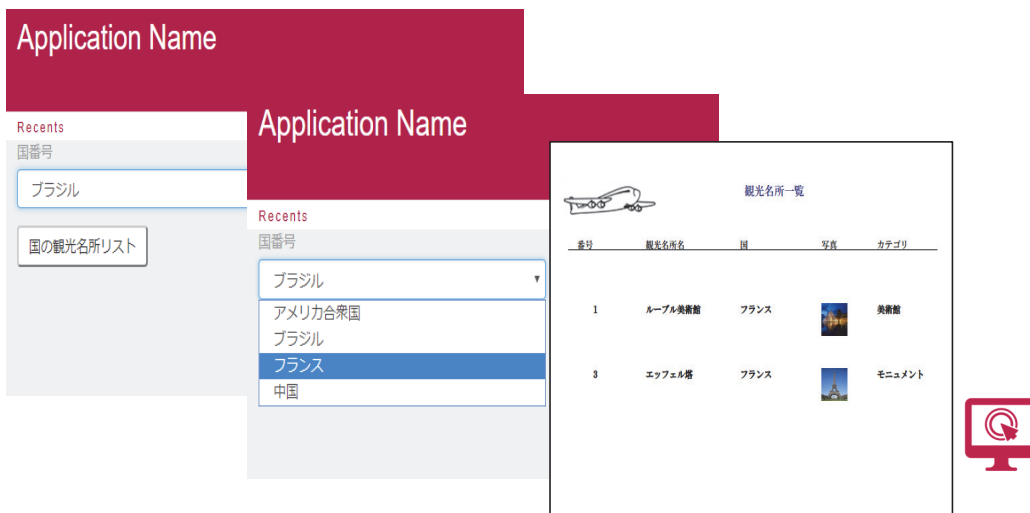
このオブジェクト (AttractionsList) では、ユーザーが国を入力するために使用した Web パネルと同じ名前とデータタイプの変数を作成しました。ただし、既に説明したとおり、2 つの異なる変数があります。1 つは Web パネル内でのみ有効であり、もう 1 つはプロシージャ内でのみ有効です。それぞれのオブジェクトで異なる名前を使用することもできますが、データの通信と共有が適切に行われるようにするには、呼び出し元と呼び出し先のオブジェクトで同じ**データタイプ**にする必要があります。

これで、プロシージャオブジェクトは国番号を受け取る準備ができました。この例では、EnterAttractionsFilter Web パネルから受け取ります。

For each コマンドで指定していた定数のフィルタ (国番号「2」) を、パラメーターとして値を受け取るように変更します。

## デモ

- ここまでの内容を GeneXus で試す: F5 キー  
AttractionsList プロシーチャーは開発者メニューに表示されなくなる



F5 キーを押して、ここまでの内容を実行してみましょう。AttractionsList プロシーチャーは表示されなくなります。Web パネルを通じて呼び出す必要があります。

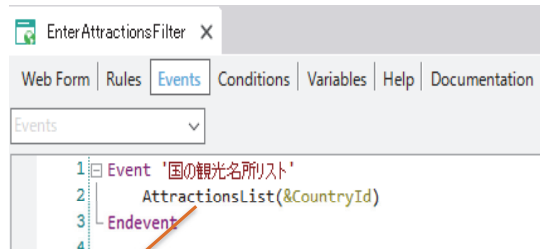
国のコンボボックスで、[フランス] を選択し、ボタンを押します。

ダイナミック コンボ ボックスから [フランス] を選択することで、内部的には「フランス」の国番号が選択されました (この場合、値は「2」)。この値が AttractionsList プロシーチャーに送信されます。

レポートが作成されると、国が「フランス」である観光名所のみが表示されることを確認できます。

## オブジェクトの呼び出し ポイント1 オブジェクト名()で呼び出す

呼び出し元: EnterAttractionsFilter



オブジェクト名()  
で呼び出せる

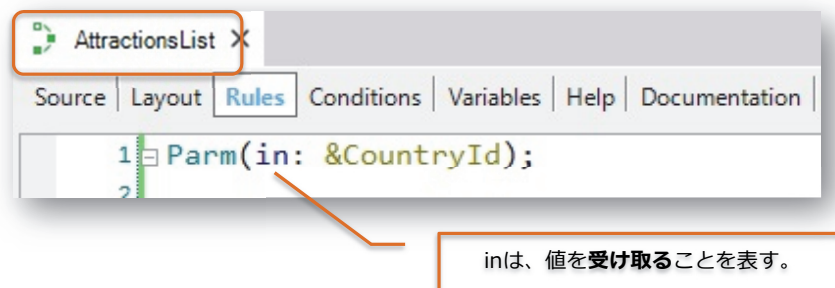
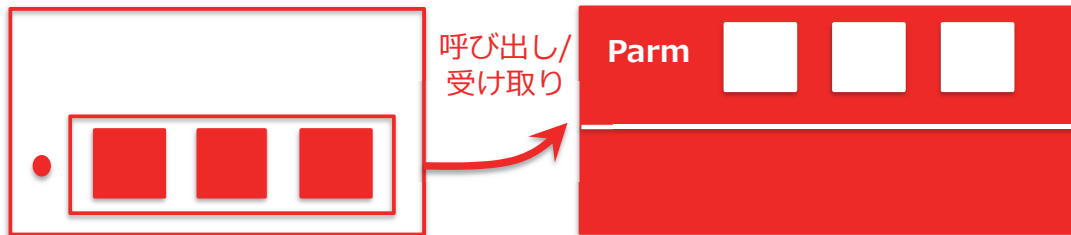


## オブジェクトの呼び出し ポイント2

呼び出し先は、Parm ルールで値 (パラメーター) を受け取る

オブジェクト A

オブジェクト B



オブジェクトが値 (パラメーターと呼びます) を受け取れるようにするには、[Rules] エlementを開いて、**Parm** ルールを記述する必要があります。この **Parm** ルールで、オブジェクトが呼び出し元から値を受け取ったり、呼び出し元に返却したりするためのパラメーターを宣言します。

「in」は、&CountryId 変数が**入力**パラメーターになることを表します。これは、呼び出し元から値を受け取るためだけに使用できることを意味しています。戻り値を返すことはできません。これは省略して、GeneXus に推論させることもできます。

## オブジェクトの呼び出し ポイント3 引数の変数は同じ型でなければならない

呼び出し元: EnterAttractionsFilter

呼び出し先: AttractionsList

The screenshot displays two windows from the GeneXus IDE. The left window, titled 'EnterAttractionsFilter', shows the 'Events' tab with the following code:

```
1 Event '国の観光名所リスト'
2   AttractionsList(&CountryId)
3 Endevent
4
```

The right window, titled 'AttractionsList', shows the 'Rules' tab with the following code:

```
1 Parm(in: &countryId);
2
3 Output_file("観光名所リスト", 'PDF');
4
5
```

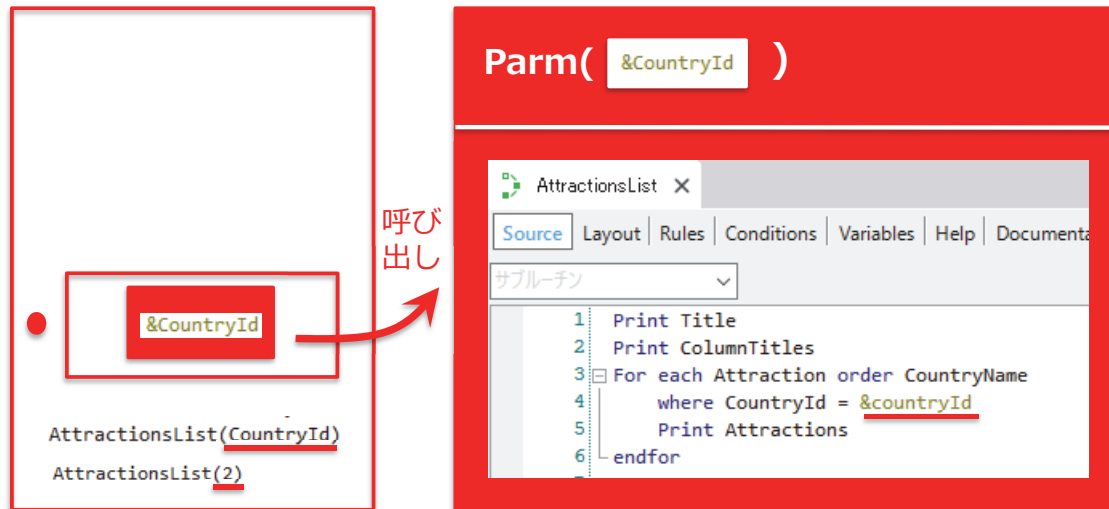
An orange arrow points from the `&CountryId` argument in the `AttractionsList` call in the left window to the `&countryId` parameter in the `Parm` rule in the right window. A box with the text '同じ型であればok' (If the same type is ok) is connected to the arrow.

## オブジェクトの呼び出し ポイント4

変数に限らず、項目属性/定数/エクスプレッションも、値として渡せる

呼び出し元： EnterAttractionsFilter

呼び出し先： AttractionsList



今回の Web パネルのケースでは、この値は (ユーザーが画面上で入力した) **変数**に格納されていましたが、もしも、値が**項目属性**に格納されていたら、呼び出し時の () 内にその項目属性を指定していました。

また、**エクスプレッション**や**定数**を送信することもできます。

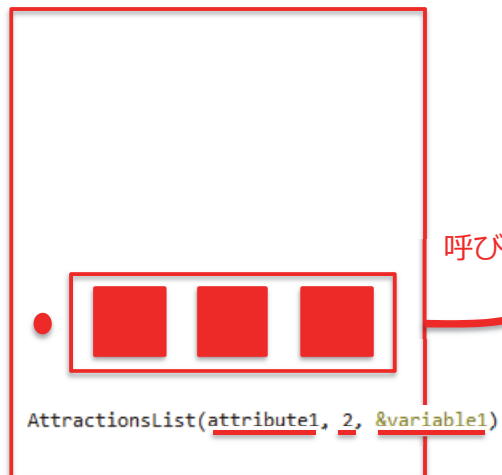
エクスプレッション※

※条件式、計算式、関数などの評価結果のこと

## オブジェクトの呼び出し ポイント5

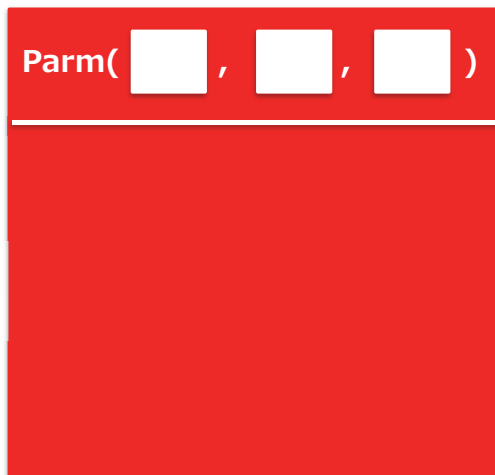
複数のパラメータを扱うにはカンマで区切る

呼び出し元：



呼び出し

呼び出し先：



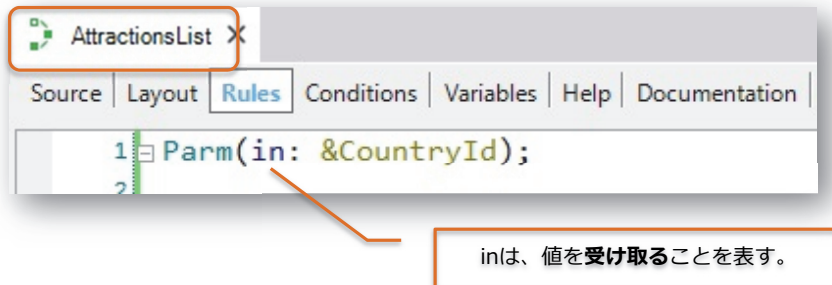
2 つ以上の値を送信する必要がある場合は、項目属性、定数、変数のいずれかを複数使用して、コンマで区切って送信します。

これらのパラメーターは、Parm ルール内でコンマ区切りで順番に宣言する必要があります。

当然、パラメーターを受け取らないオブジェクトで Parm ルールを宣言してはいけません。

**補足：**

Parm ルールを宣言した場合、そのオブジェクトは開発者メニューから表示されなくなる



※開発者メニューには引数の必要のないオブジェクトがリストアップされるため

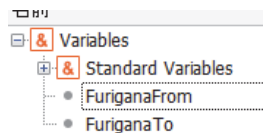
上記、Parm ルールが宣言されていると、今後はプロシーチャーを呼び出すすべてのオブジェクトが国番号の値を送信できるようになります (そうする必要があります)。このタイプの値を送信しないと、プロシーチャーを呼び出せません。そのため、AttractionsList プロシーチャーは開発者メニューに表示されなくなります。

## まとめ オブジェクトの呼び出しの基本

1. オブジェクト名()で呼び出す
2. Parm ルールで値 (パラメーター) を受け取る
3. 引数の変数は同じ型でなければならない
4. 変数/項目属性/定数/エクスプレッションが渡せる
5. 複数のパラメータを扱うにはカンマで区切る

## 例：特定の範囲のふりがなの観光名所をPDF出力する

## 呼び出し元Webパネル：EnterAttractionsFilter

呼び出し先プロシージャー：  
AttractionsByName

```

Parm(&FuriganaFrom,&FuriganaTo);
output_file("AttractionsByName","pdf");

print Title
print columnTitles
For each Attraction order AttractionName
  where AttractionFurigana >= &FuriganaFrom
  where AttractionFurigana <= &FuriganaTo
  print attractions
Endfor

```

ここでは、ユーザーが選択した特定の値の範囲に名前（ふりがな）が含まれるすべての観光名所を表示します。たとえば、「あ」から「と」の範囲とします。

これを行うには、既に作成してある Web パネルに、ユーザーが開始名と終了名を入力できるようにします。そうすることで、ボタンを押したときに、名前がその範囲に含まれるすべての観光名所を表示するリストが呼び出されます。

EnterAttractionsFilter Web パネルを開き、変数を 2 つ持つテーブルを追加します。

- &FromAttractionFurigana は AttractionFurigana 項目属性に基づきます。
- また、&ToAttractionFurigana も AttractionFurigana の定義に基づきます。既に説明したように、これは、変数の定義が項目属性の定義にリンクしており、項目属性のデータタイプを変更すると、それに応じて変数が自動的に変更されることを意味します。

次に、「ふりがなフィルタ付き観光名所リスト」イベントボタンを追加します。追加したボタンを右クリックして [イベントへ移動] を選択します。ここで、選択した範囲に含まれる観光名所を出力するプロシージャーを呼び出す必要があります。

AttractionsList レポートは既にありますが、名前の範囲ではなくパラメーターで国番号を受け取っていました。これを別名 (AttractionsByName) で保存し、Parm ルールを変更して、次の 2 つの入力パラメーターを受け取れるようにします：&FuriganaFrom 変数と &FuriganaTo 変数。

両方の変数を AttractionFurigana 項目属性に基づいて定義します。

変数には、Web パネルで作成した変数とは異なる名前を使用しています。もっとも重要なのは、**送られたデータタイプと受け取ったデータタイプ**が一致していることです。

呼び出し部分で記述した最初のパラメーターは、呼び出し先オブジェクトの Parm ルールで定義された最初のパラメーターにロードされ、呼び出し部分の 2 つ目のパラメーターは、呼び出し先オブジェクトの 2 つ目のパラメーターにロードされます。呼び出し部分の順序と Parm ルールの定義での順序に注意する必要があります。コードを分かりやすくするため、この例のように関連する名前を使用することをお勧めします。

これでプロシーチャーの準備ができました。F5 キーを押して実行しましょう。



## 実行時

## Application Name

Recent観光名所一覧出力画面 — Enter Attractions ...

国番号

観光名所名

観光名所名

観光名所一覧

番号	観光名所名	国	写真	カテゴリ
5	スミソニアン博物館	アメリカ合衆国		美術館
4	コロンバードのキリ	ブラジル		モニュメント
3	エッフェル塔	フランス		モニュメント
1	ルーブル美術館	フランス		美術館
7	凱旋門	フランス		モニュメント
2	万里の長城	中国		モニュメント
6	スカイツリー	日本		有名なランドマー

呼び出されたオブジェクトが  
戻り値を返す場合

## ふりかえり : FlightFinalPrice

The screenshot displays the GeneXus IDE interface. The top menu bar includes 'Structure', 'Web Form', 'Win Form', 'Rules', 'Events', 'Variables', 'Help', 'Documentation', and 'Patterns'. The main workspace shows the 'Flight' object structure with various attributes and their types. A formula editor window is open, showing the formula for 'FlightFinalPrice'.

名前	タイプ	デスクリプション	式	Null 許容
Flight	Flight	フライト		
FlightId	Id	フライト番号		No
FlightDepartureAirportId	Id	出発空港番号		No
FlightDepartureAirportName	Name	出発空港名		
FlightDepartureCountryId	Id	出発空港国番号		
FlightDepartureCountryName	Name	出発空港国名		
FlightDepartureCityId	Id	出発空港都市番号		
FlightDepartureCityName	Name	出発空港都市名		
FlightArrivalAirportId	Id	到着空港番号		No
FlightArrivalAirportName	Name	到着空港名		
FlightArrivalCountryId	Id			
FlightArrivalCountryName	Name			
FlightArrivalCityId	Id			
FlightArrivalCityName	Name			
FlightPrice	Price			
FlightDiscountPercentage	Percentage			
AirlineId	Id			
AirlineName	Name			
AirlineDiscountPercentage	Percentage	航空会社が提示する割引率		
FlightFinalPrice	Price	フライト最終価格	FlightPrice*(1-AirlineDiscountP... count(FlightSeatLocation)	
FlightCapacity	Numeric(4,0)	フライト座席数		
Seat	Seat	座席		
FlightSeatId	Id	座席番号		No
FlightSeatChar	SeatChar	座席文字		No
FlightSeatLocation	Location	座席位置		No

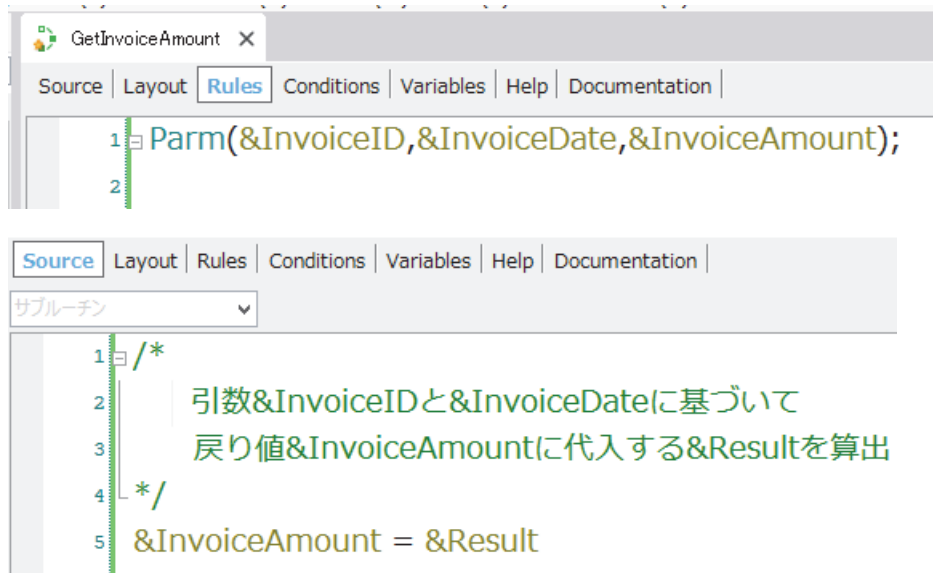
The formula editor shows the following formula for 'FlightFinalPrice':

```
FlightPrice*(1-AirlineDiscountPercentage/100) IF AirlineDiscountPercentage>=FlightDiscountPercentage;  
FlightPrice*(1-FlightDiscountPercentage/100) OTHERWISE;
```

Flight トランザクションでは、航空会社が提示する割引率と、フライト自体に設定された割引率に応じて、航空券の価格を計算する式がありました。もっとも割引率の大きいものを選択して適用しました。

より複雑な要件の例：

⇒プロシージャオブジェクトGetInvoiceAmountによる算出  
(新規Invoiceトランザクションで利用)



今回のケースでは、請求書の割引金額を複雑な条件で決定されるものとして考えます。

このような場合は、これらの計算を実行し、計算結果の値を呼び出し元に返すプロシージャを実装する必要があることがあります。

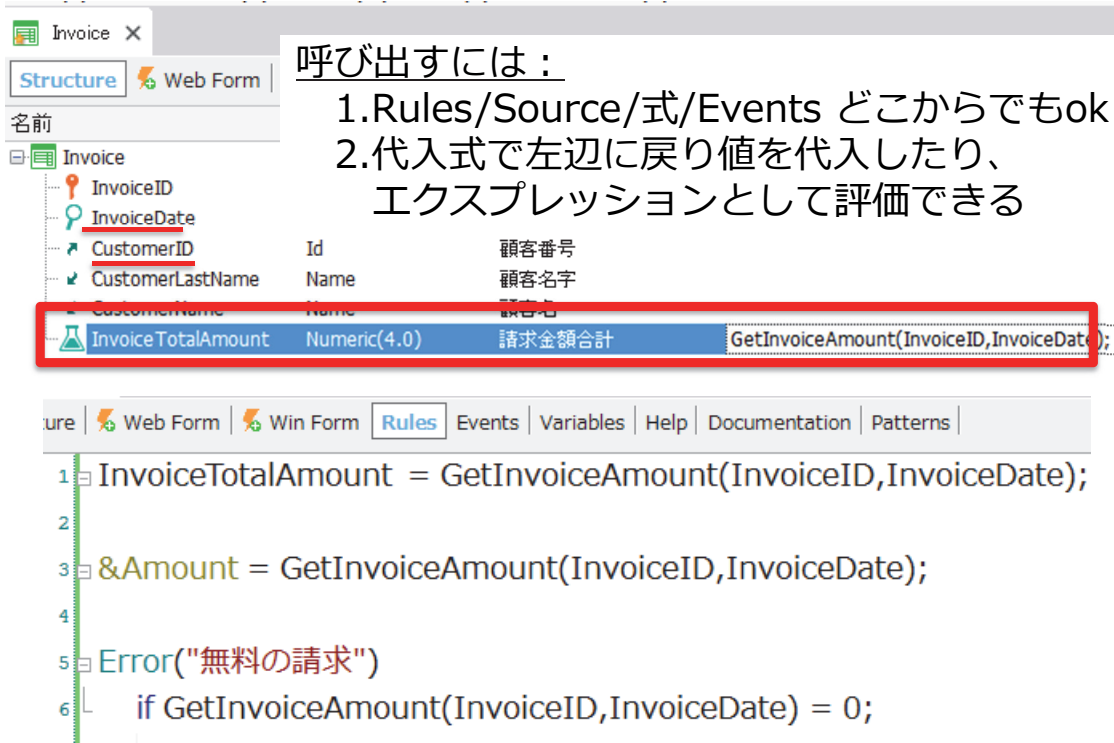
たとえば、この GetInvoiceAmount プロシージャを呼び出す場合です。

プロシージャは、入力パラメーターとして請求番号と請求日付を受け取り、そして計算された金額を返します。

最初の疑問は、プロシージャの結果を必要とするオブジェクトが、この結果をどのようにして受け取るかということです。これは、戻り値を返し、必要なアクションを実行するために呼び出される関数として検討する必要があります。

呼び出すには :

1. Rules/Source/式/Events どこからでもok
2. 代入式で左辺に戻り値を代入したり、  
エクスプレッションとして評価できる



```

1 InvoiceTotalAmount = GetInvoiceAmount(InvoiceID, InvoiceDate);
2
3 &Amount = GetInvoiceAmount(InvoiceID, InvoiceDate);
4
5 Error("無料の請求")
6 if GetInvoiceAmount(InvoiceID, InvoiceDate) = 0;

```

1 つの方法として、結果を項目属性に割り当てることが考えられます。たとえば、Invoice トランザクション構造内で FlightDiscount 項目属性を定義し、呼び出し元の GetDiscount によって計算される**グローバル式**にすることができます。

こうすることで、InvoiceTotalAmount 項目属性が記述されるすべてのオブジェクトで式が評価されるようになります。GetDiscount プロシーチャーが呼び出されて実行されます。実行が終わると、返された結果は式の項目属性の値として表示されます。

この項目属性を式として設定せず、対応するテーブルに格納される項目属性として設定し、トランザクションが実行された場合にのみプロシーチャーの結果と一緒に格納されるようにする場合は、上のスライドの [Rules] の最初にあるとおり、**Assignment ルール**を記述します。

さらに、プロシーチャーの実行結果は、変数に割り当てることもできます。また、割り当てずに、代わりに**エクスプレッション**で使用することもできます。たとえば、ルールをトリガーする条件を指定する場合です。

または、プロシーチャーやイベント内の次のような命令の場合です：

```

If GetInvoiceAmount ( InvoiceID, &Today) > 10
...
Endif

```

現在学習している本題とは関連がないため、ここでは GetDiscount プロシーチャーの実装方法については説明しません。ただし、今確認した例のように、呼び出し構文でオブジェクトが戻り値を返すことを想定している場合は、呼び出し先オブジェクトで **Parm ルール**を記述する方法を把握する必要があります。

呼び出し先の定義:

1. Parm ルールの最後に、戻り値を宣言して呼び出し元の左辺に代入できる。
2. 戻り値は変数で[Source]で代入される必要あり

呼び出し元 :

```
&Amount = GetInvoiceAmount(InvoiceID,InvoiceDate);
```

呼び出し先 :  
GetInvoiceAmount

```
Parm(&InvoiceID,&InvoiceDate,&InvoiceAmount);
```

Source	Layout	Rules	Conditions	Variables	Help	Documentation
サブルーチン						
1	*/					
2		引数&InvoiceIDと&InvoiceDateに基づいて				
3		戻り値&InvoiceAmountに代入する&Resultを算出				
4	*/					
5		&InvoiceAmount = &Result				

GetDiscount プロシーチャーの [Rules] セクションで、呼び出し元に記述されているパラメーターに加えて、最後にもう 1 つパラメーターを追加した Parm ルールを記述する必要があります。

Parm ルールの最後に指定するパラメーターは、次のように定義する必要があります:

- 変数
- オブジェクトコード(この場合はプロシーチャーのソース内)で値を代入

代入された値は、コード実行終了時に、呼び出し元への戻り値となります。

**補足1:**

代入式を使わなくてもok

⇒戻り値を返すことが不明瞭で非推奨

呼び出し元 :

```
GetInvoiceAmount(InvoiceID,InvoiceDate,&Amount);
```

呼び出し先 :

GetInvoiceAmount

```
Parm(&InvoiceID,&InvoiceDate,&InvoiceAmount);
```

**補足2:**

Parmルールでin、out、inoutを指定

⇒複数の戻り値を明示できる

```
Parm(in:&InvoiceID,in:&InvoiceDate,inout:&InvoiceAmount);
```

この同じオブジェクト B に対して、代入式を使わずに呼び出すことも可能です：  
 この場合は、最初の 2 つのパラメーターが入力パラメーターで、最後のパラメーターが出力パラメーターになります。

唯一の問題は、この呼び出しでは、&discount がロードされて返されることが明確に示されていません。つまり、呼び出されたオブジェクトが戻り値を返すことが明確になっていないということです。このような場合は、左辺に戻り値を返すことを明示する呼び出し構文を使用することをお勧めします。

呼び出し元に返す出力パラメーターとして使用できるのは Parm

ルールの変数の最後だけではないということも説明しておく必要があります。すべての変数が、入力変数 (パラメーター)、出力変数 (戻り値)、または入出力変数 (パラメーター兼戻り値) として使用できます。この例では、&var1 は (オブジェクトのコード内で指定することで、parm ルールの最後に位置していないパラメーターにも関わらず) 出力変数にも入出力変数にもなりえます。

ここに示されているように、これは変数名の前に「in」、「inout」、または「out」と入力することで指定できます。

## まとめ 呼び出し先オブジェクトから値を戻す場合

### 戻り値のあるオブジェクトを呼び出すには：

1. Rules/Source/式/Events どこからでも呼び出せる
2. 代入式で左辺に戻り値を代入したり、  
エクスプレッションとして評価できる
3. 代入式を使わなくてもok  
※[Rules]では代入式のほうが望ましい

### 戻り値を戻すオブジェクトの定義：

1. Parm ルールの最後に、戻り値を宣言して  
呼び出し元の左辺に代入できる。
2. 戻り値は変数で[Source]で代入される必要あり
3. Parmルールで、outやinoutを  
使い複数の戻り値を戻すこともできる



## Parmルールによる 自動フィルタ

Parm ルールのパラメーターが変数ではなく項目属性である場合を確認しましょう。

```
Parm(CountryID);
```

と

```
Parm(&CountryID);
```

呼び出されるオブジェクトの **Parm** ルールで変数を使用する場合と項目属性を使用する場合の違いは何でしょうか。

**変数**で値を受け取る場合は、プログラムで自由に使用できます。「等価 (=)」、「より大きい (>)」、「以上 (>=)」などでフィルタする際のフィルタ条件として使用できます。また、計算 (算術演算) や、その他必要な処理にも使用できます。変数は、必要な処理を実行する際に、明示的な命令を通じてオブジェクト内で使用する名前の付いたメモリ領域です。

一方、**項目属性**で値を受け取る場合は定数となり、暗示的に使用されます。一般的に、開発者はデータベース上のテーブルにアクセスするときに (拡張テーブルに含まれる) 項目属性で値を受け取らせます。このように、項目属性のパラメーターを通じて、値を受け取った場合、自動的に等価フィルタが適用され、項目属性にパラメーターとして渡された値を含むレコードが処理対象となります。例を使って説明します。

## 例: 変数で値を受け取る場合と項目属性で値を受け取る場合の比較

AttractionsList	AttractionsReport
<pre> 1 Parm(in: &amp;countryId); 2 3 Output_file("観光名所リスト", 'PDF'); 4 </pre>	<pre> 1 Parm(in: countryId); 2 3 Output_file("観光名所リスト", 'PDF'); 4 </pre>
<pre> 1 print Title 2 print ColumnTitles 3 For each Attraction order CountryId 4   where CountryId = &amp;CountryId 5   print Attractions 6 endfor </pre>	<pre> 1 Print Title 2 Print ColumnTitles 3 For each Attraction order CountryName 4 5   Print Attractions 6 endfor </pre>

ナビゲーションリスト→

AttractionsReport という名前で AttractionsList プロシーチャーのコピーを作成します。

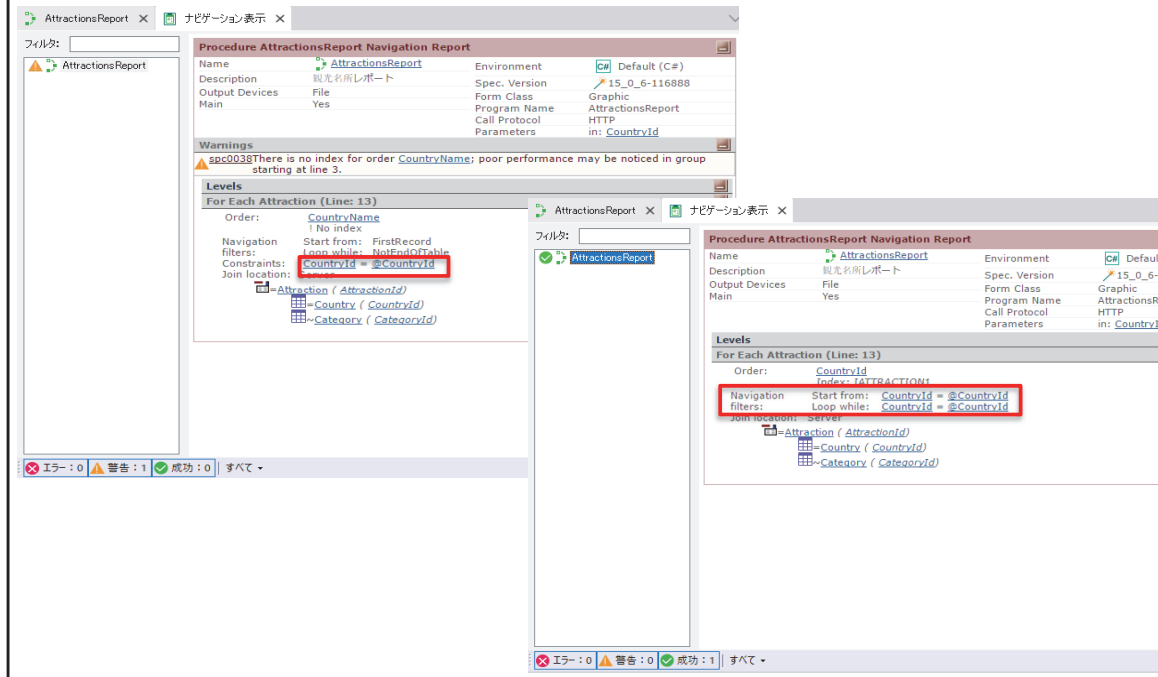
基になるプロシーチャーでパラメーターとして次の変数を使用します: &CountryId。Attraction テーブルの観光名所を国でフィルタするために使用します。

ここに示すように、等価フィルタを実装しています。パラメーターで受け取った &CountryId 変数の値と CountryId が一致する観光名所のみが表示されます。

一方、明示的にフィルタを指定しなくても、まったく同じように実装することも可能です。どのようにすればよいでしょうか。CountryId 項目属性で値を直接受け取ることで実装できます。

Parm ルールで、項目属性で値を受け取る場合は、GeneXus は等価フィルタを使用します。つまり、国番号が一致するレコードのみがアクセスされます。このオブジェクトのナビゲーションリストを確認してみましょう。

## ナビゲーションリスト



Where 節が記述されていなくてもフィルタが適用されることが確認できます。

【補足】 Navigation Filters: For each コマンドが CountryName でソートされて実行されるため、パラメーターに対応する CountryId の値でフィルタするには、テーブル全体を参照する必要があります。

一方、フィルタに使用する項目属性でソートする場合は、テーブル全体が参照されているわけではないことがナビゲーションリストで確認できます。

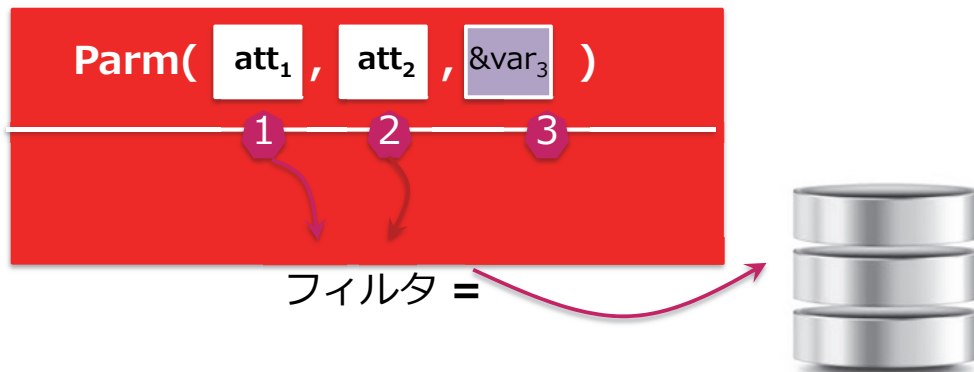
実行します。

[フランス] を選択してリストを表示します。

リストにはフランスの観光名所が表示されます。

**補足1:**

複数の項目属性の引数を受け取った場合はAnd条件

**補足2:**

「=」以外の比較演算子には変数で受け取る

```
parm( inout: &var1, in: &var2, out: &var3 );
```

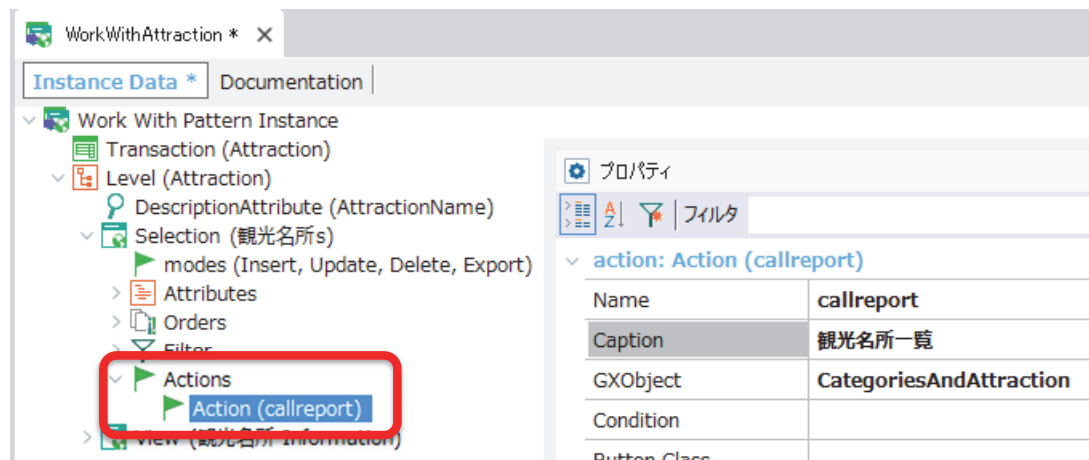
項目属性を使用して複数の値を受け取った場合は、受け取った各項目属性と同じ値のレコードのみがアクセスされます。

これまで同様、変数ではないので、これらの項目属性の値を変更することはできません。

値を受け取って等価フィルタを適用することが目的ではない場合は、項目属性ではなく、変数で値を受け取ることが解決策となります。また、変数はプログラムで自由に使用できます。たとえば、必要に応じて別の値を割り当てることもできます。

Workwithパターンで作成した  
オブジェクトからほかの  
オブジェクトを呼び出す

## 例：観光名所から観光名所リストを出力する



Workwithパターンからほかのオブジェクトを呼び出すには  
ActionノードからGXObjectプロパティで任意のオブジェク  
トを指定します。

## 実行時

**観光名所s**  + 追加 観光名所一覧

観光...	観光...	観光...	国名	都市名	カテ...	カテ...	観光...
5	エッフェル塔	えっふえる	ブラジル		0		更新 削除
2	東京スカイツリー	とうきょうスカイツリー	日本	東京			削除

観光名所一覧

カテゴリ

番号 観光名所名 国

5 エッフェル塔 ブラジル

カテゴリ 世界遺産

番号 観光名所名 国