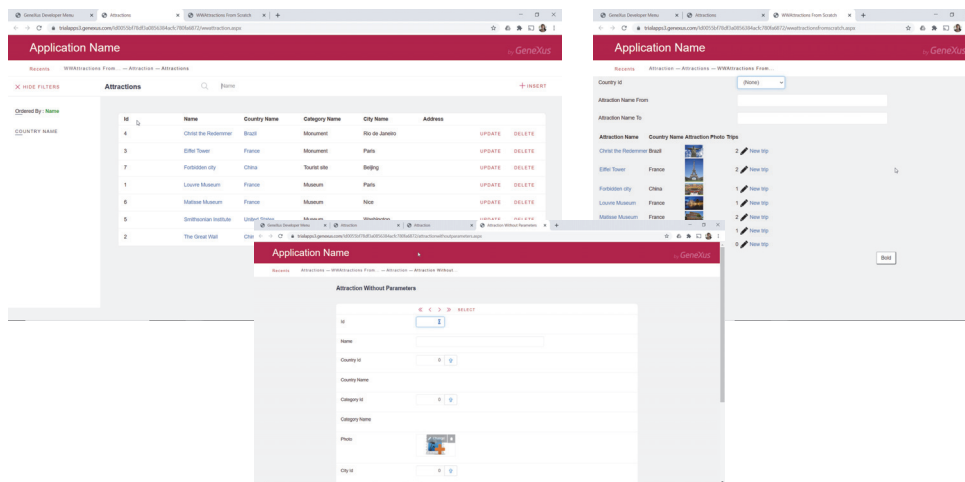


バックオフィスに重点を置いた Web 画面

Web パネルのタイプ

*GeneXus*TM

すべてのページに共通するもの



前の章では、Work With パターンによって自動的に構築されるものと非常によく似たソリューションをゼロから構築しました。

両方のソリューションを実行して比較すると、まずデザインが異なることに気がきます。ここまではロジックに焦点を当て、ユーザーインターフェースは後で説明することにして省略してきました。視覚的な違いを別にするると、どちらの Web パネルにも観光名所の情報を表示するグリッドとフィルタがあります。表示される情報は最新の情報で再表示される可能性があります。

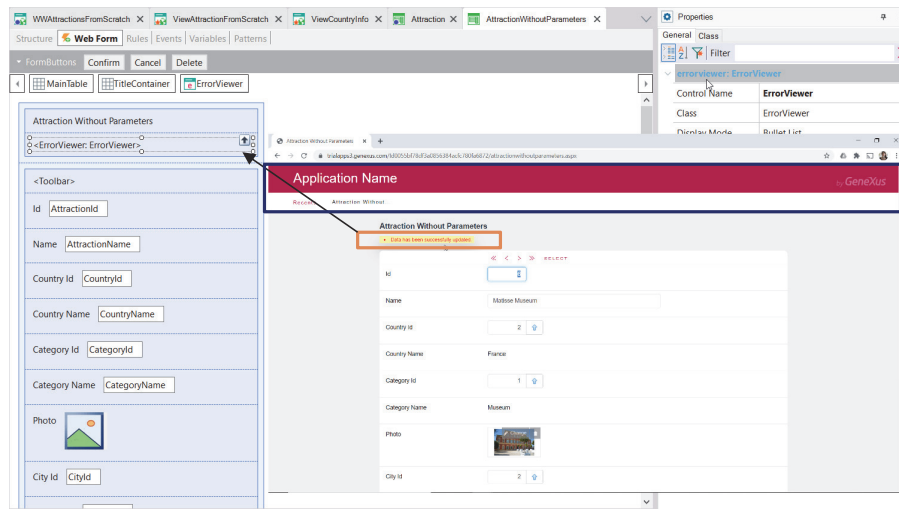
どちらのソリューションも、トランザクションを更新モードで呼び出します。

ユーザーの目を引くものとして、ユーザーが閲覧するすべての画面の上部に、共通のバーが配置されています。このバーには、最近閲覧したリンクが表示されます。いくつかページを開くと、すべてのページにこの部分があることが分かります。

たとえば、並行トランザクション `AttractionWithoutParameters` を直接呼び出した場合、ここに表示されます。

また、観光名所のビュー、パターンのビュー、またはゼロから実装したビューに移動すると、それぞれ相違はありますが、この共通部分は常に表示されます。

トランザクションコントロール



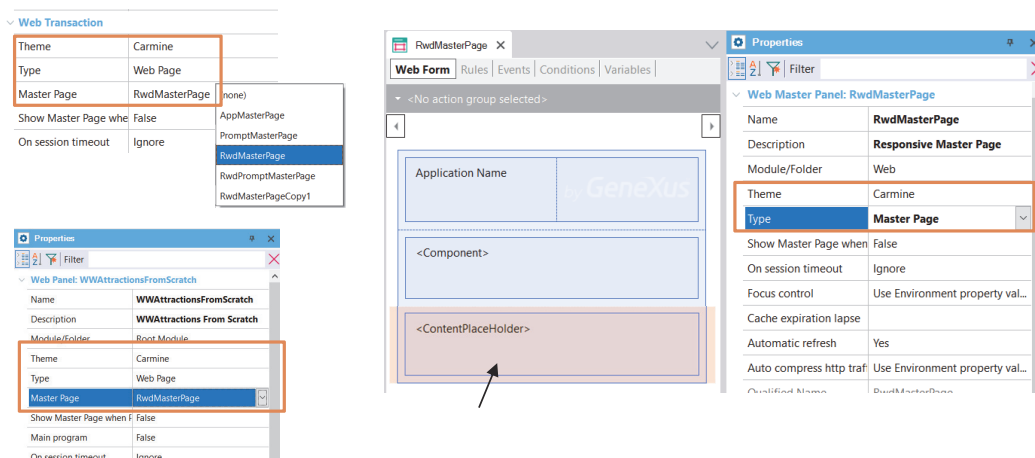
GeneXus にアクセスして、実装したビューのフォームを見てみましょう。この共通機能はどこにあるのでしょうか。

Attraction トランザクションやパラメーターを受け取らない並行トランザクションを開き、フォームを確認しても、この部分は見つかりません。

上部に表示されているのは、トランザクションの名前を持つ TextBlock コントロールです。その下に、成功または失敗などのメッセージが表示される ErrorViewer コントロールがあります。次に、ナビゲーションボタンと [SELECT] ボタンが表示されるアクショングループ コントロールがあります。次に項目属性があり、最後にいくつかのボタンが表示される別のアクショングループがあります。

ページ上部のセクターはどこにあるのでしょうか。

マスターページ



トランザクションのプロパティを見ると、[Web Transaction] グループに 3 つの重要なプロパティが含まれています。1 つ目はコントロールのデザインを制御する [Theme] プロパティです。2 つ目は後で説明するとして、ここでは 3 つ目の [Master Page] プロパティについて説明します。開発者が指定する (このトランザクションオブジェクトのページに対応する) マスターページがあります。ここには、これまでにナレッジベースで定義した、すべてのマスターページがオプションとして選択できます。ナレッジベースが作成されると、これらのページが作成されて既定で提供されますが、ほかのページも作成できるようになります。トランザクションは、作成時に既定でこのマスターページに関連付けられます。

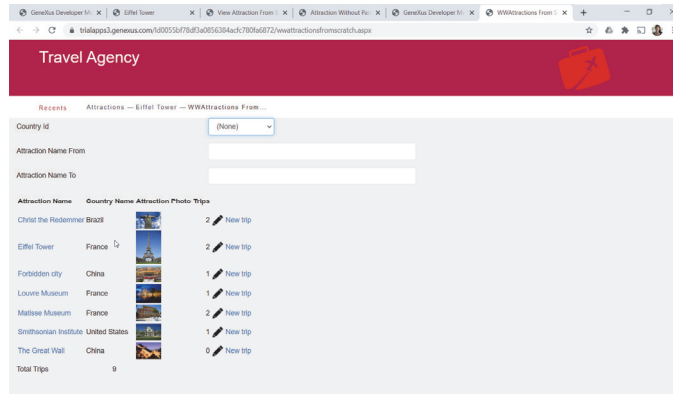
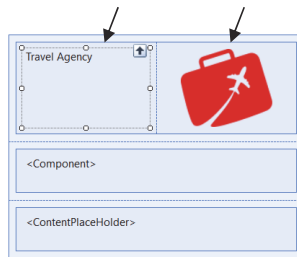
実行時に表示されていた別のオブジェクトを開いた場合、それらのオブジェクトにも、ここに示すようなマスターページがあります。なお、Web パネルにも同じ 3 つのプロパティがあることに留意してください。

それでは、このマスターページを探して、詳しく確認しましょう。場所が分からない場合は、ここから検索できます。また、KB エクスプローラーで検索する場合は、タブを右クリックして検索できます。マスターページは、GeneXus によってパターンなどに関係するオブジェクトが配置される Web フォルダ内にあります。

そのプロパティを見ると、[Theme] というプロパティと [Type] というプロパティはありますが、[Master Page] はありません。なぜでしょうか。これは、[Master Page] という特殊なタイプの Web パネルであるためです。このタイプの Web パネルには、ContentPlaceHolder という特別なコントロールが含まれています。ここには、これをマスターページとして持つページがロードされます。

その結果、トランザクションやビューそしてこれまで見てきた画面は、このスペース (ContentPlaceHolder) の中にロードされます。

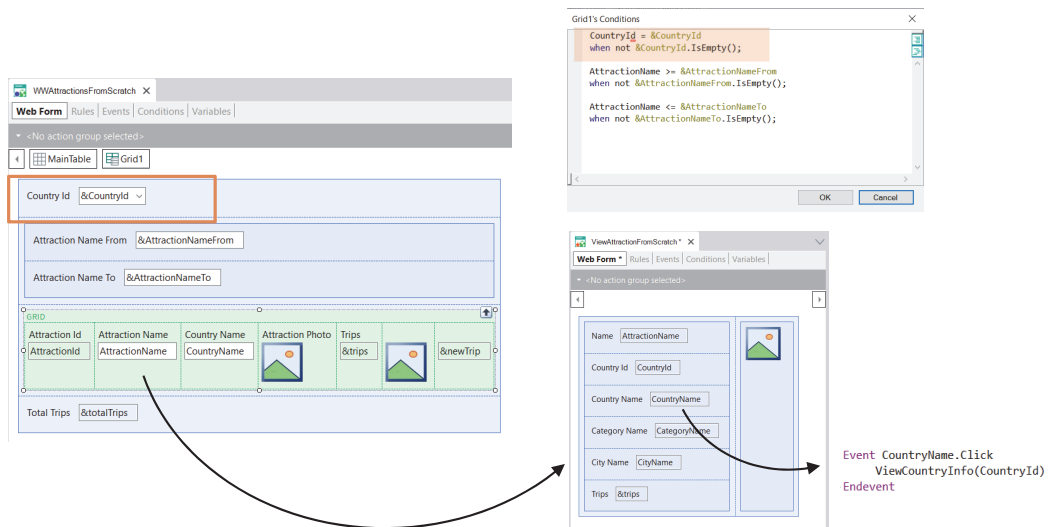
マスターページ



左上は TextBlock コントロールであり、実行時にアプリケーションの名前が表示されます。たとえば、これを「Travel Agency」に変更してみましょう。右側は画像です。事前にナレッジベースに挿入していた画像に変更できます。

これらの変更がアプリケーションに及ぼす影響をテストしてみましょう。結果はスライドに示すとおりです。

Web コンポーネント



これまでに、3 つのタイプの Web 画面のうち、マスターページと共通 Web ページの 2 つを確認しました。後者は、Web パネルやトランザクションを作成するときに毎回使用してきたものです。あと 1 つ、**コンポーネント**タイプの **Web ページ**について学習します。

そのために、手順を少し戻って、前の章で実装したことを思い出してみましょう。観光名所の Work With エレメントには制限がありました。具体的には、ユーザーはこの変数に値を選択することによって国でフィルタリングできますが、この変数をグリッドの条件として使用し、国でフィルタリングできるのは、変数が空でない場合のみです。

それだけでなく、このパネルからユーザーが観光名所の情報を表示するには、対象となる観光名所名をクリックする必要がありました。また、その国の情報をすべて表示するには、国名のリンクをクリックする必要がありました。

Web コンポーネント

The screenshot shows the GeneXus Web Form designer interface. The main workspace displays a form layout with the following components:

- Country Name:** A text input field labeled "CountryName".
- Attraction Name From:** A text input field labeled "&AttractionNameFrom".
- Attraction Name To:** A text input field labeled "&AttractionNameTo".
- GRID (Left):** A table with columns: Attraction Id (AttractionId), Attraction Name (AttractionName), Attraction Photo (image icon), Trips (&trips), and a button (&newTrip). Below the table is a "Total Trips" label with the value "&totalTrips".
- GRID (Right):** A table with columns: City Id (CityId), City Name (CityName), and Attractions (&attractions). Below the table is a "Total Attractions" label with the value "&totalAttractions".

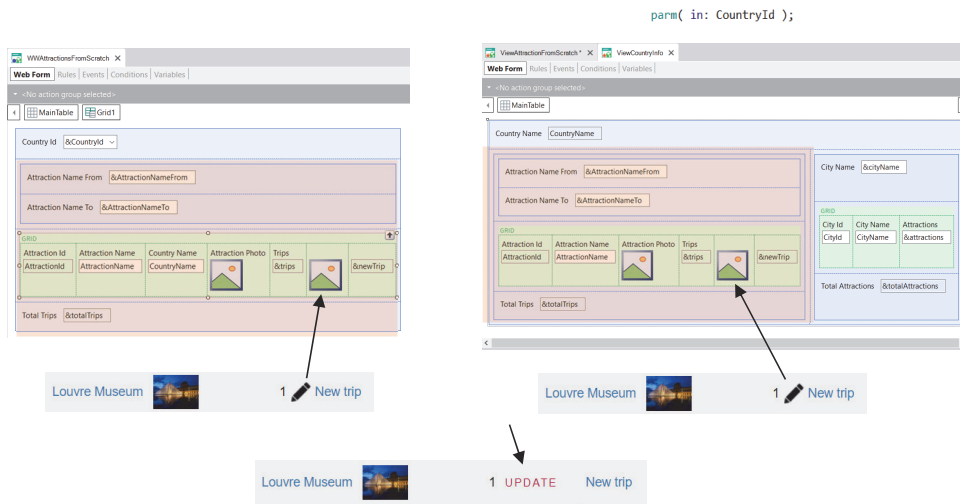
The top of the designer shows tabs for "ViewAttractionFromScratch" and "ViewCountryInfo", and a "Web Form" tab. The "MainTable" is selected in the left sidebar.

```
param( in: CountryId );
```

そのため、観光名所と都市のリストを同時にみるためには、項目属性で国の識別子を受け取り、国名を表示する、スライドのような新しい Web パネルを実装する必要があります。

左側のグリッドはフィルタは、手動で実装した一覧画面と一致します。右側には、都市に関する情報が表示されます。

Web コンポーネント

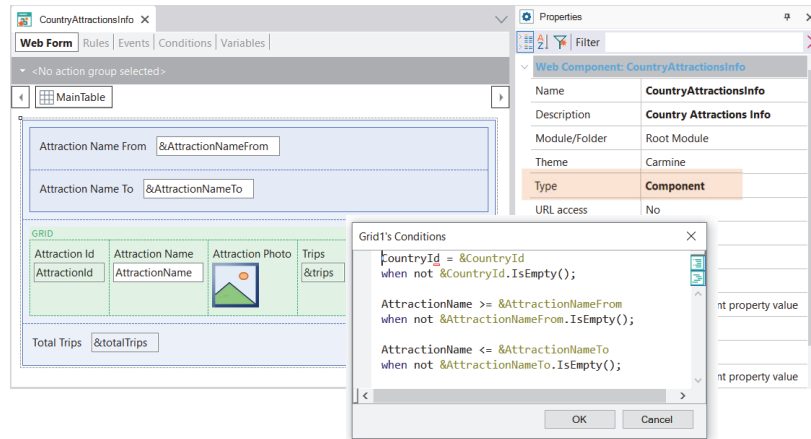


このパネルの一部は、もう一方のパネルとほとんど同じであることが分かります。更新オプションを、鉛筆の画像から「UPDATE」という文字に変更して、パターン内に同じように表示する場合について考えます。

両方のパネルでこの画像をテキストブロックに置き換える必要があります。重複を回避し、パネルの一部の動作とデザインを 1 回だけプログラムするには、コンポーネントタイプの Web パネルを使用します。

Web コンポーネント

```
parm( in: &CountryId );
```

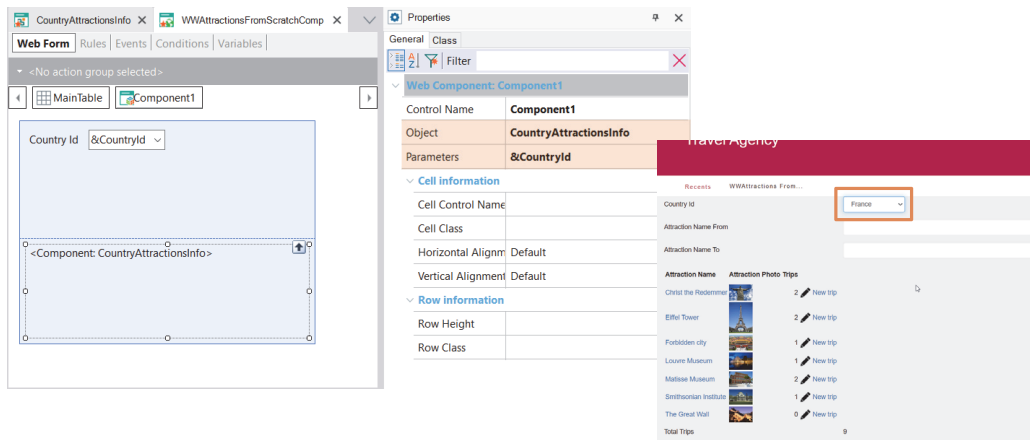


この場合、何を繰り返すのでしょうか。それは画面のこのセクション全体と、その動作です。

この Web パネルを別名で保存し、フォームから CountryId 変数のみを削除し、代わりにパラメーターとして配置します。
値は、この画面のフォームにユーザーが直接入力するのではなく、呼び出し元から受け取ります。

イベントやその他のすべての部分については条件は変わりません。もう 1 つの変更は、[Type] プロパティを [Component] に切り替えたことです。この時点で、この Web パネルは別の Web パネルのコンポーネントとして機能するようになります。

コンポーネント



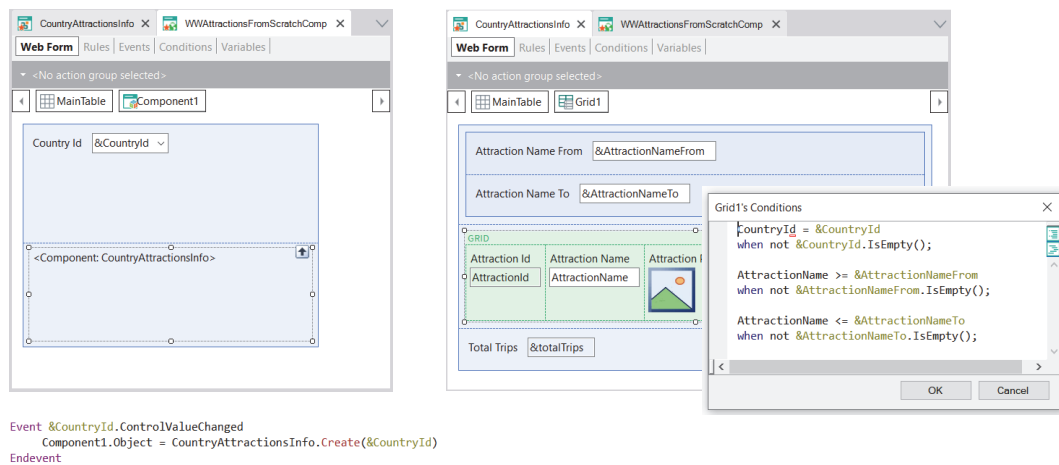
次に、元のパネルを別名で保存し、コンポーネント Web パネルに配置するすべてのコントロールをコンポーネントタイプのコントロールに置き換えます。コンポーネントに含まれているすべてのイベントを削除します。ここではコメントとして残しています。

このタイプのコントロールを配置することは、指定したものすべてをそこにロードして実行する必要があることを意味します。そこで、先ほど説明した Web コンポーネントのインスタンス `CountryAttractionsInfo` をそのコンポーネント内で作成するよう命令する必要があります。この命令は、静的な方法、つまりプロパティで指定することによって行います。コンポーネントタイプのオブジェクトを指定することで、パラメーターが送信されます。試してみましょう。

まったく同じに見えます。観光名所の名前でフィルタリングすると、完璧に機能します。

では、国でフィルタリングしたい場合はどうなるでしょうか。何も起こりません。なぜでしょうか。

コンポーネント

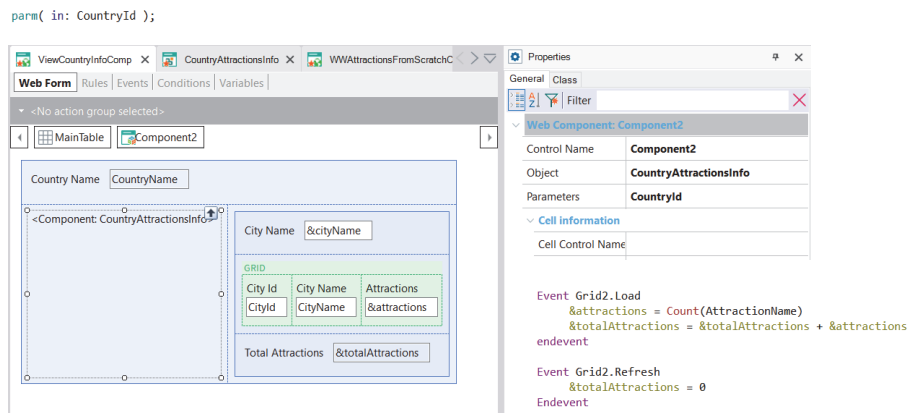


CountryId 変数は、フィルタリングされる観光名所グリッドとは異なるパネルにあります。ここでは、ControlValueChanged イベントを使用して &CountryId 変数コントロールの値の変更を検出し、コンポーネントの新しいインスタンスを作成して、変数の新しい値を渡す必要があります。

試してみましょう。

再表示して、フランスでフィルタリングしてみます。完璧です。次に、観光名所名で試します。こちらも完璧です。

コンポーネント

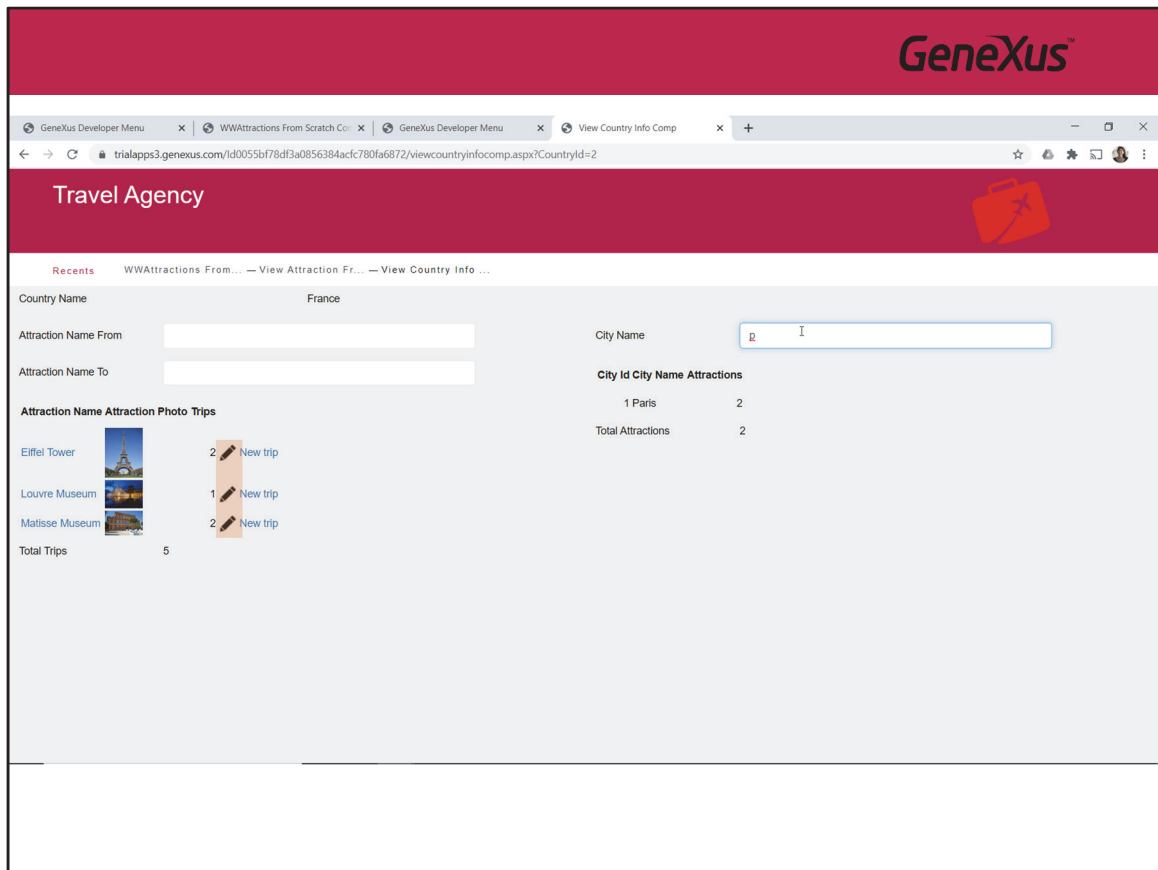


都市の情報を表示するパネルでも同様のことを行います。

このパネルを別名で保存し、このセクション全体をコンポーネントに置き換えて、このパネルでロードされるようにします。

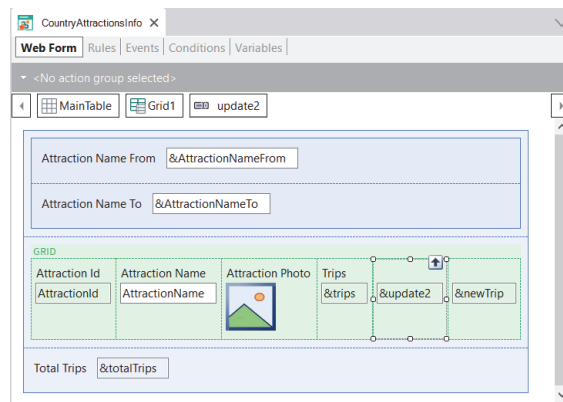
これにより、新しいパネルでは、CountryId を変数ではなくパラメーターで受け取ることになります。したがって、この Web パネルの実行内で 1 回だけ、静的な方法でコンポーネントインスタンスを作成し、それにパラメーターを渡します。このケースでは、このパラメーターは、Parm ルールで受け取った項目属性に含まれます。

元のパネルから観光名所に関連するイベントのコントロールとプログラミングを削除し、都市に関連するものだけ残しました。実行環境で確認してみましょう。



次に、更新を行うための鉛筆の画像を「UPDATE」というテキストに変更します。

Web コンポーネント



```

Event Grid1.Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent

Event Grid1.Refresh
    &totalTrips = 0
Endevent

Event Start
    &update2 = "UPDATE"
    &newTrip = "New trip"
Endevent

Event &update2.Click
    Attraction(trnMode.Update, AttractionId)
Endevent

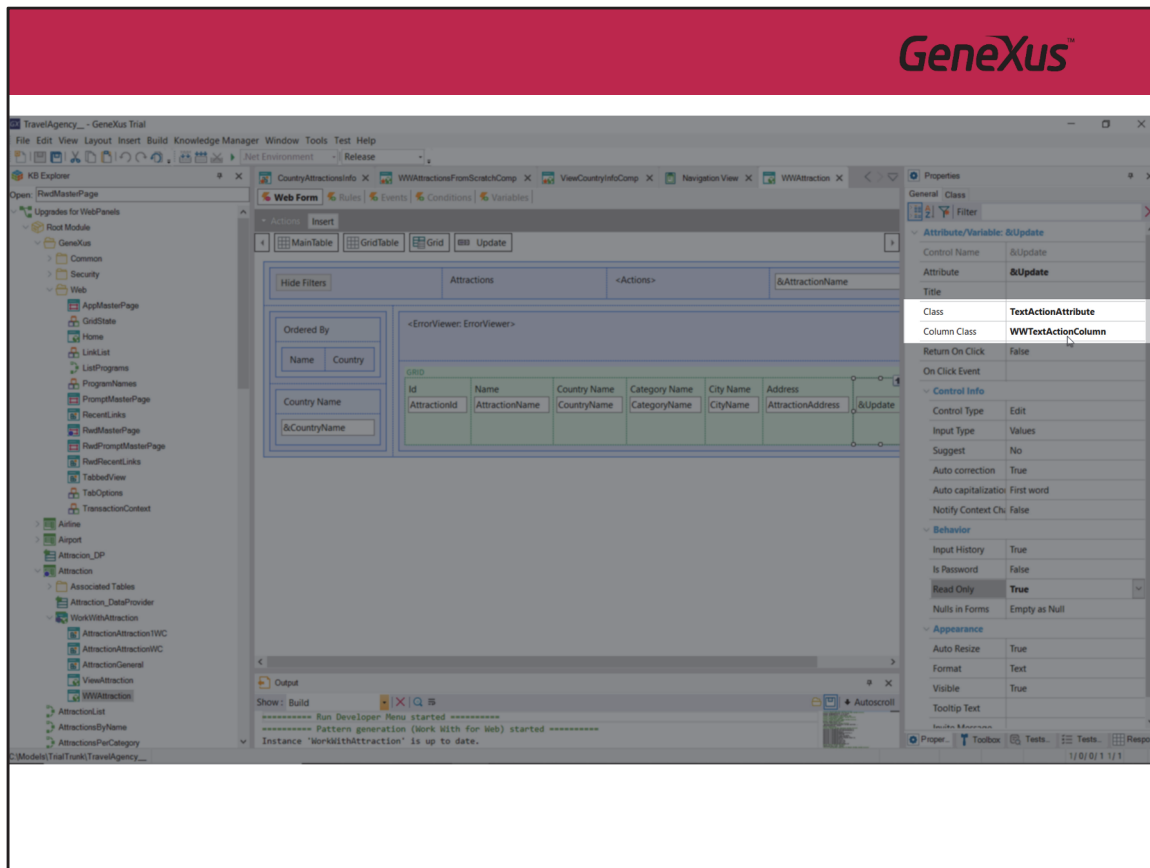
```

そのためには、コンポーネント Web パネルで、Character タイプで長さが 20 の変数を作成します。これをグリッドに挿入します。Start イベントで、「UPDATE」という値、つまりユーザーに表示する値を割り当てます。グリッドから画像を削除するため、変数への画像の割り当てを解除します。

次に、新しい変数からタイトルを削除して、読み取り専用にします。

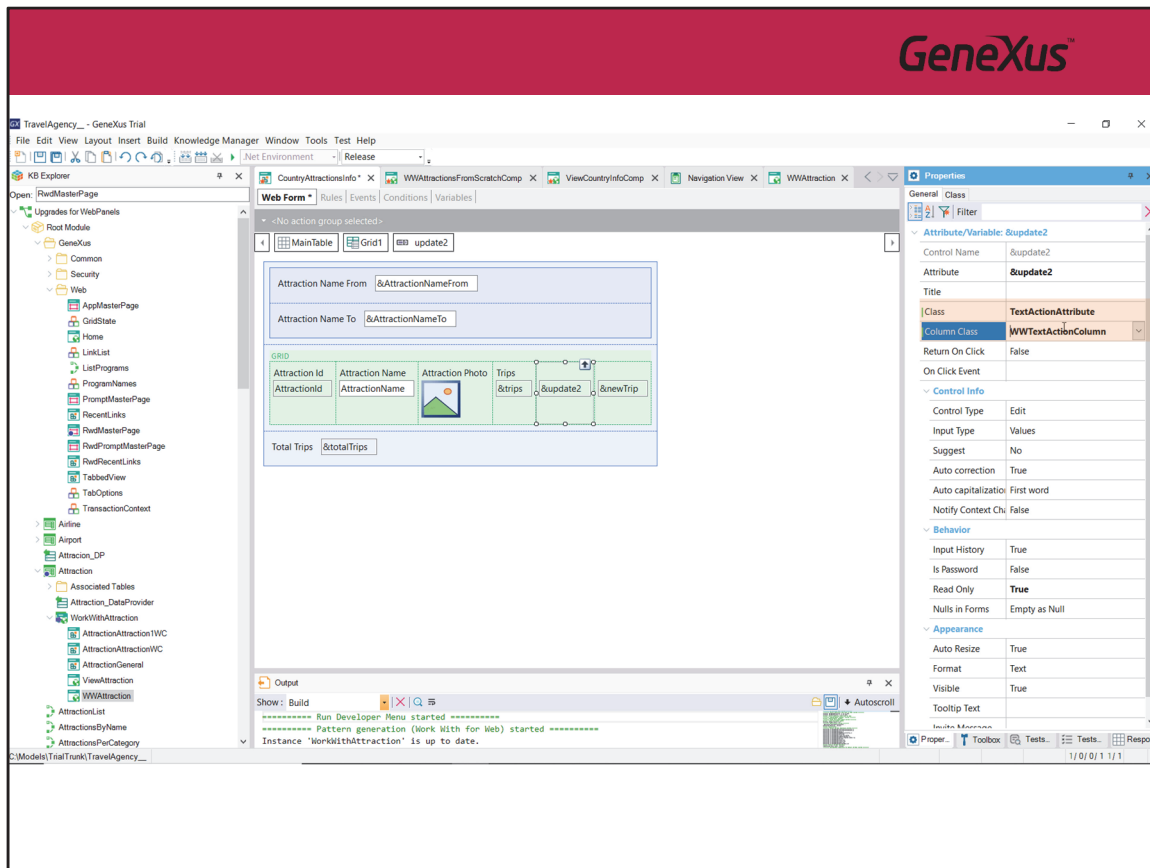
実行してみましょう。クリックは有効なイベントではないと表示されます。クリックが古い変数ではなく、新しい変数を対象とするように変更するのを忘れないでください。

その処理を行います。

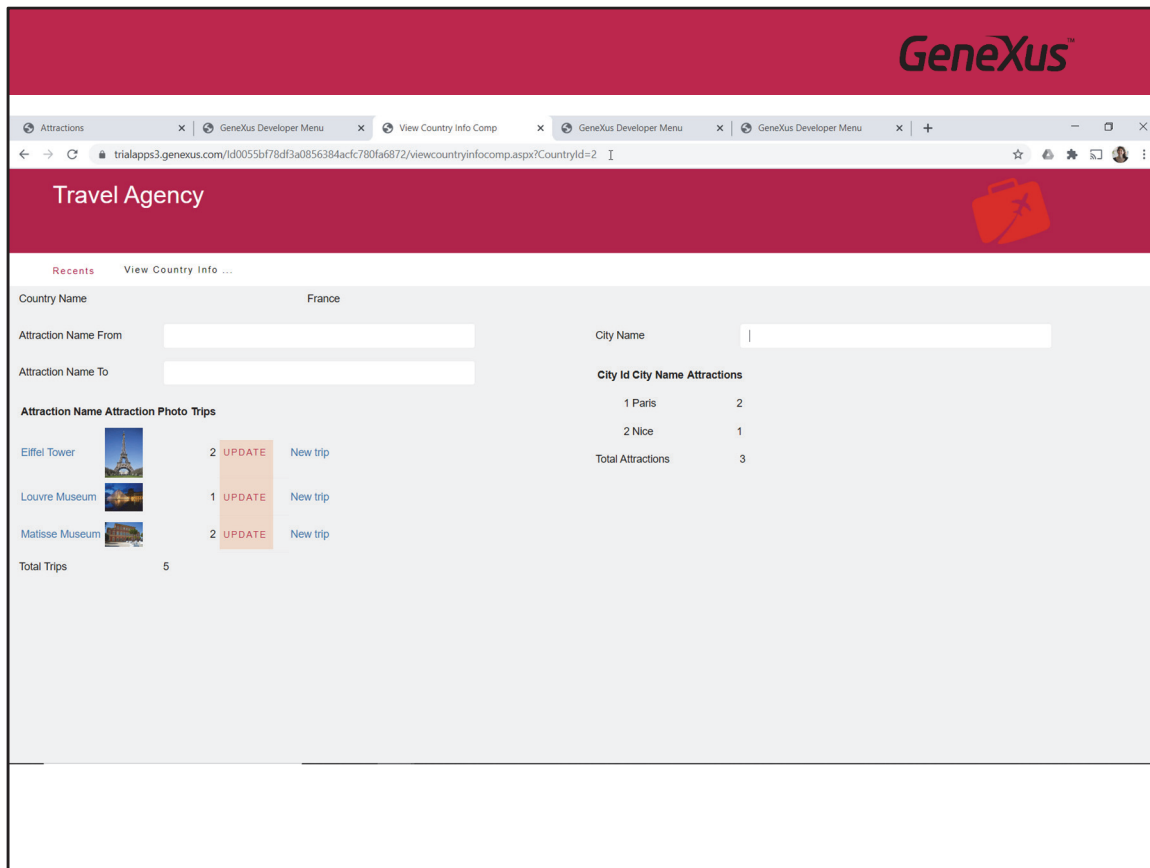


パターンでの表示結果を確認します。デザインの見栄えがはるかによいのはなぜでしょうか。

パターンによって生成された Web パネルのコントロールプロパティを調べてみましょう。確認するのは [Class] と [Column Class] の 2 つです。



Web パネルに手で挿入したコントロールのプロパティの値も見てみましょう。
値が異なります。これらにパターンの値と同じものを割り当ててみます。これ
で確認してみましょう。



結果はスライドのようになります。ここにきて、画面のデザインを操作する方法が分かってきました。

Web パネルのタイプ

マスターページ

Web ページ

コンポーネント

学習したことをまとめます。Web パネルには 3 つのタイプがあり、それらは相互に関連しています。

マスター ページ タイプは、アプリケーションのすべてのページまたはその一部に共通するレイアウトを備えています。これにより、ページごとに毎回同じことを繰り返す必要がなくなります。たとえば、メニューはここに入ります。特にこのオブジェクトのフォームには、Web ページがロードされる特別なコントロール `ContentPlaceHolder` が含まれています。

Web パネルタイプは、ここで学習しているタイプの 1 つであり、`ContentPlaceHolder` にロードされるため、特定のマスターページを 1 つだけ関連付けることができます。

コンポーネントタイプは、同じデザインとプログラミングを別のオブジェクトで再利用するのに役立ちます。コンポーネントタイプの Web パネルとして定義されたオブジェクトを別の Web オブジェクトにロードするために、静的または動的にロードできるコンポーネントタイプのコントロールが使用されます。

このように、さまざまなコンポーネントを理解し、使い分けることで、より優れたアプリケーションにすることができます。

このトピックに関しては、コンポーネントを含むパネルでイベントを実行すると何が起こるか、コンポーネントがどのように再表示されるかなど、学習すべきことはまだたくさんあります。しかし、今のところはこれで十分です。