顧客向けに重点を置いた Web 画面

Angular の使用

GeneXus™

これまで、アプリケーションのバックオフィスに重点を置いた画面を構築する方法を見てきました。具体的には、旅行代理店の従業員が会社のデータを入力および管理するために使用する部分です。次に、旅行代理店のクライアントが情報を閲覧するために使用する画面、つまりアプリケーションの顧客向け部分をデザインおよび実装する方法を説明します。

GeneXus

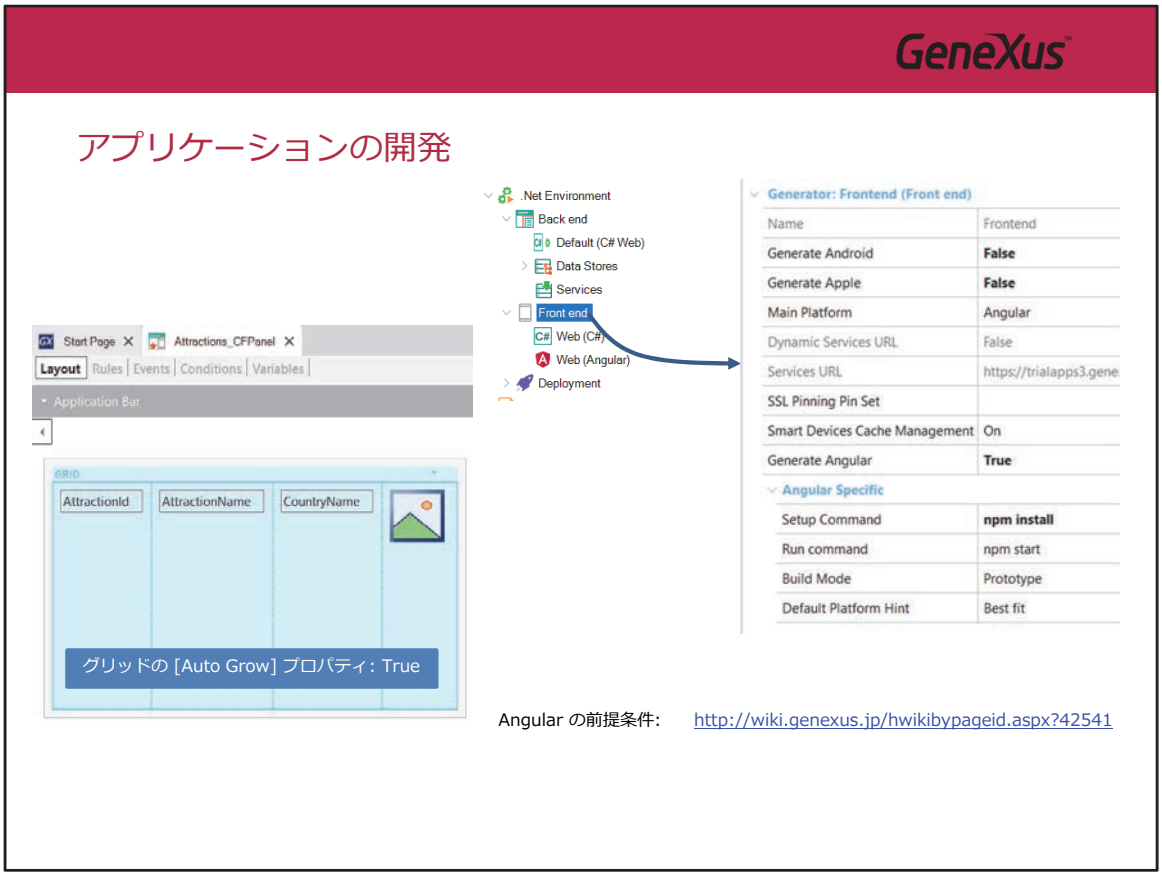
旅行代理店の要望: 顧客向けアプリケーション

Panel オブジェクト

旅行代理店の要望は、クライアント向けの Web 画面に訪問可能な観光名所を表示し、そこで情報を操作できるようにする、たとえば、画面上のデータをフィルタリングして検索結果を絞り込んだり、選択した観光名所の詳細を表示したりできるようにすることです。

Web パネルを使用して画面を構築する方法は既に説明しましたが、ここでは、Panel オブジェクトを使用して画面を実装する方法を説明します。Panel オブジェクトを使用すると、Angular でのアプリケーションの生成が可能になります。

なお、この同じ Panel オブジェクトを使用して、Android や Apple デバイス用のモバイルアプリケーションの画面を生成することもできます。



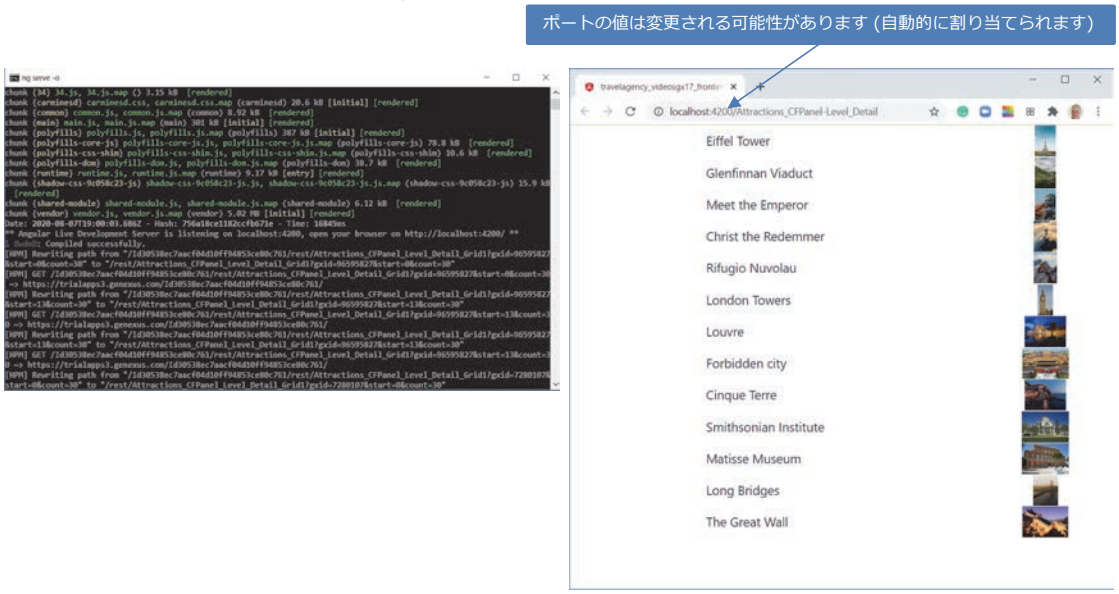
訪問可能な観光名所のリストを作成し、興味のある観光名所をクリックすると、その観光名所の詳細情報を表示できるようにします。考え方としては、前に取り上げた WWAttractionsFromScratch Web パネルに似たパネルを構築します。

FrontendAngular フォルダを作成し、Attractions_CFPanel というパネルタイプのオブジェクトを作成します。ご覧のとおり、ここにはツールバーからコントロールをドラッグできるフォームもあります。
まず、グリッドをドラッグして、AttractionId、AttractionName、CountryName、および AttractionPhoto の各項目属性を選択します。次に、AttractionId の [Visible] プロパティを False に設定し、グリッドの [Base Trn] プロパティで Attraction トランザクションを割り当てます。

次に、ジェネレーターとして Angular を割り当てます。KB エクスプローラーで、[フロントエンド] をクリックし、[Generate Angular] というプロパティ (既定値は False) があることを確認し、その値を True に設定します。また、[Generate Android] と [Generate Apple] が既定で True に設定されていますが、ここでの目的はモバイルデバイス向けの生成ではないため、いずれも False に設定します。

Angular で生成できるようにするには、画面に表示されている Wiki の記事に記載のソフトウェアをインストールする必要があります。

アプリケーションの実行

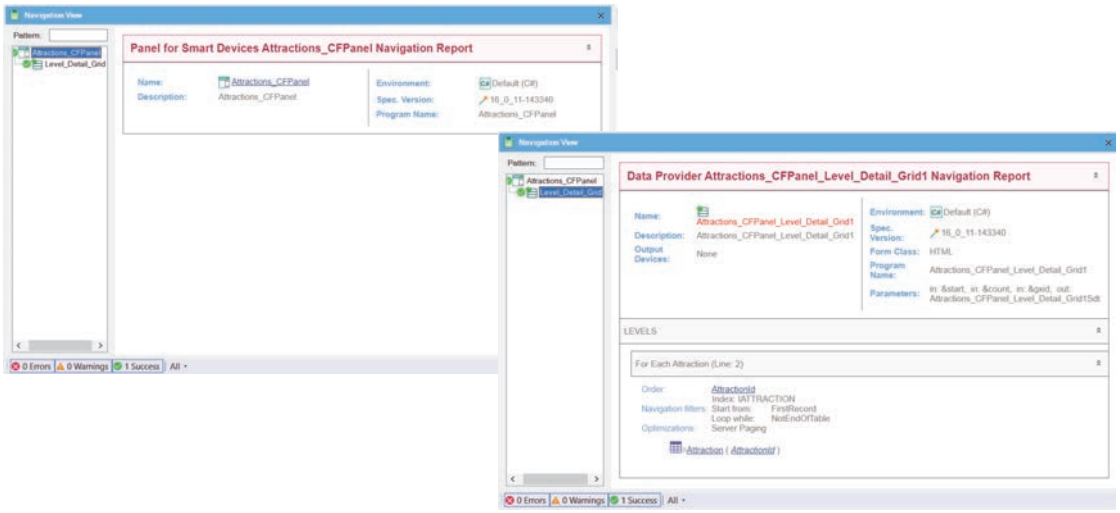


実行するには、作成したパネルをメインとして設定し、右クリックして [実行] を選択します。サーバーの実行を示すコマンドラインウィンドウが開き、数分後、アプリケーションが実行された状態でブラウザーが開きます。

デザインはかなり貧弱ですが、それはこれから対応していきます。アプリケーションは、既定では URL http://localhost:4200 で実行されます。これは、アプリケーションが自動的にインストールされたローカル開発サーバーのアドレスです。
初めて生成した Angular アプリケーションが、もう実行されています。

GeneXus

ナビゲーション表示



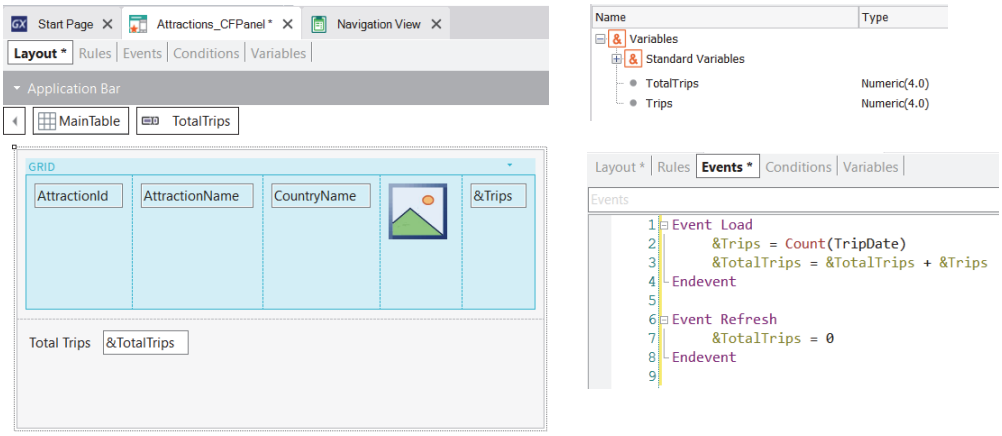
Attractions_CFPanel Panel オブジェクトのナビゲーション表示に移動すると、2 つのエントリーが表示されていることが分かります。パネル名がある方を選択すると、リストは空であり、Attraction テーブルへの参照は表示されません。これは Level_Detail_Grid1 というノードにあり、パネルのグリッドには観光名所を表示するために、Attraction テーブルが参照されるため、このノードのナビゲーション表示は Attraction ベーステーブルを持つ Web パネルで表示されるものと同じになります。

リストの Attractions_CFPanel というノードは、パネル自体に対応しており、ここには、ダイナミック コンボ ボックスのナビゲーションなど、ユーザーインターフェースの要素に関する情報が含まれています。パネルの固定部分には要素がないため、リストは空で表示されます。

Level_Detail_Grid1 というノードは、挿入したグリッドに対応しています。グリッドは挿入された観光名所の項目属性をもっており、実行時には Attraction ベーステーブルは、期待どおりに参照されることになります。レポートのタイトルには、ナビゲーションがデータプロバイダー Attractions_CFPanel_Level_Detail_Grid1 に対応していることが示されます。このデータプロバイダーは、バックエンドでサービスとして公開され、データベースにアクセスして観光名所を取得するために、パネルによって呼び出されます。

これで、Angular を使用して生成したこれらの顧客向けアプリケーションのアーキテクチャにはクライアントレベルのロジックがあること、そしてデータベース内の情報にアクセスする際にはサーバー上でサービスが呼び出されることが証明されました。

各観光名所を含むツアーの数と、全体のツアーの合計数を追加する



前に見た Work With Attractions Web パネルに近づけるために、パネルのグリッドに各観光名所を含むツアーの数を示す列を追加し、グリッドの外側にすべての観光名所のツアーを合計した数を表示します。変数 &Trips および &TotalTrips を作成し、[Label Position] プロパティを None に設定して、&Trips をグリッド内、&TotalTrips をグリッドの下にそれぞれ配置します。

観光名所を含むツアーの数を取得するには、TripAttraction テーブル全体を対象として照会する必要があります。これはデータベースに関連することであるため、サーバーでイベントがトリガーされるようにプログラムする必要があります。Panel オブジェクトでも、Web パネルと同様に、Start、Refresh、および Load イベントを扱います。

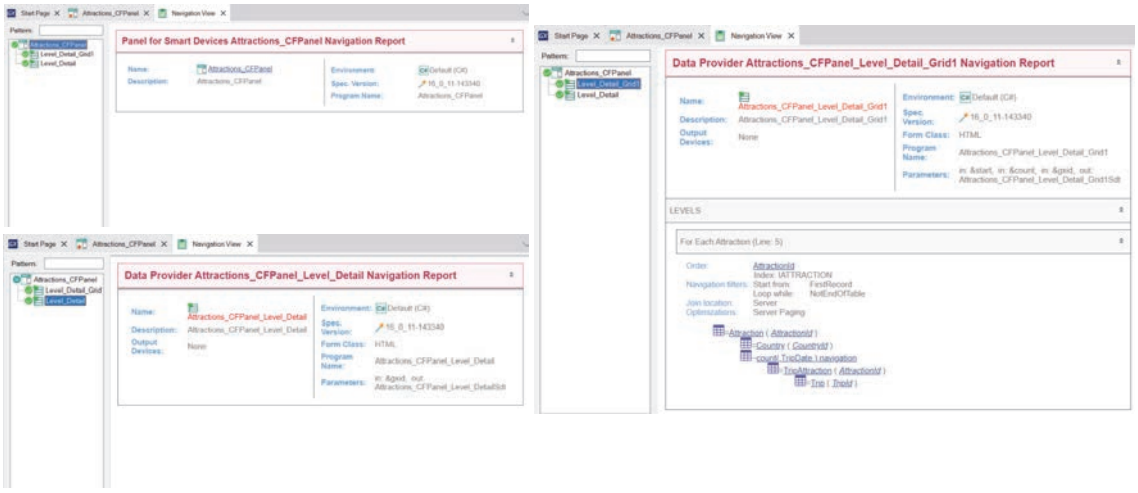
Web パネル WWAttractionsFromScratch で行ったように、イベントをプログラムします。Load イベントでは、TripDate 項目属性に基づいてツアーのレコードを計算する Count 式を使用します。前に説明したとおり、TripDate 項目属性は Trip テーブルに属していますが、GeneXus では、この式のベーステーブルとして Trip テーブルは選択されず、代わりに TripAttraction テーブルが選択されます。ナビゲーション表示に表示されているとおり、グリッドは Attraction テーブル全体を対象とすることから、グリッドの各行に対して Load イベントが 1 回トリガーされ、それに応じて、Count 式により、表示されている各観光名所を含むツアーの数が計算されます。次に、すべてのツアーの合計数を &TotalTrips で累積します。この場合も前の説明と同様のことが当てはまります。このケースでは、グリッドが 1 つしかないため汎用の Load イベントを使用しますが、Grid1.Load イベントを使用することもできます。その場合、将来的にパネルに別のグリッドを追加しても、プログラミングに変更を加えなくて済みます。

観光名所を含むツアーの数を計算する前に、ツアーの合計数をゼロに初期化する必要があります。そのため、Refresh イベントでは、Web パネルで行ったのと同様の方法で、初期化を &TotalTrips 変数に追加します。

再度実行するには、パネルを右クリックして [実行] を選択します。

GeneXus

ナビゲーション表示



ナビゲーション表示を見ると、グリッドとは異なる新しい詳細レベル Level_Detail があることが分かります。ここには、パネルの固定部分、つまりグリッドに含まれていないフォームのすべてのコントロールのロードが表示されます。

Web パネルとは異なり、Panel オブジェクトでは、固定部分はグリッドから独立しており、後で説明するように、グリッドのベーステーブルとは異なるベーステーブルを使用することもできます。

Level_Detail ノード (タイトルに記載されているデータプロバイダーを呼び出す) のナビゲーション表示は空のように見えます。これは、データベースからフィールドがロードされず、Load イベント内で更新される &TotalTrips 変数のみが存在するためです。

Level_Detail_Grid1 のナビゲーション表示に移動すると、対応するデータプロバイダーが Attraction テーブルにアクセスしていて、既定の順序である AtractionId で並べ替えられていることが分かります。また、Load イベントでトリガーされる Count 式のナビゲーションも含まれています。

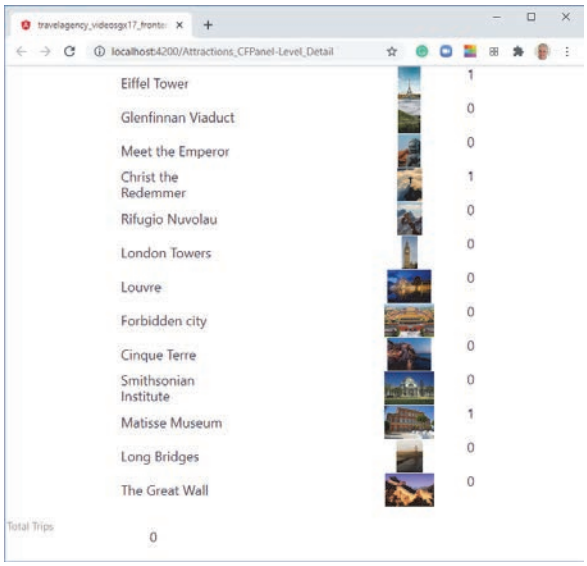
全体のツアーの合計数が誤って計算される

Layout * | Rules | **Events *** | Conditions | Variables

Events

```
Event Load
    &Trips = Count(TripDate)
    &TotalTrips = &TotalTrips + &Trips
Endevent

Event Refresh
    &TotalTrips = 0
Endevent
```



ブラウザーで実行されているアプリケーションを見ると、各観光名所を含むツアーの数は正しく表示されていますが、全体のツアーの合計数は 0 になっています。何か間違えているのでしょうか。&TotalTrips 変数は、Web パネルの場合と同様に、Load イベントで増分されます。また、一部の観光名所を含むツアーが実際に確認されているため、このイベントがトリガーされているのも間違いありません。では、なぜでしょうか。

理由は、Panel オブジェクトが Web パネルと同じように機能しないことです。Panel オブジェクトでは、固定部分はグリッドから独立してロードされます。このケースでは、&TotalTrips 変数はパネルの固定部分にあり、これが最初にロードされます。それからグリッドがロードされるため、変数が表示されたときにグリッドはロードされておらず、Load イベントもまだトリガーされていません。その結果、&TotalTrips の値はまだ追加されていません。

顧客向けアプリケーションの開発に使用される Panel オブジェクトは、クライアントデバイスに画面をロードするために、サーバー上のサービスを呼び出してデータベースにアクセスします。これらのサービスはデータプロバイダーであり、画面の固定部分とグリッド (または各グリッド) で切り離されています。

パネルの実行が開始されると、クライアントでローカルイベントがトリガーされ、固定部分をロードするために、データプロバイダーを呼び出し、サーバーでトリガーされる Start イベントと Refresh イベントを発生させます。その後、2 番目のデータプロバイダーが実行され、それによってサーバーで Load イベントが N 回トリガーされてグリッドがロードされます。

この例では、最初に Refresh がトリガーされたときに &TotalTrips 変数が初期化され、固定部分がロードされます。その後、Load イベントがトリガーされ、その時点で &TotalTrips が正しい値でロードされ、グリッドが更新されますが、固定部分はそれより前に既にロードされていて再描画されません。

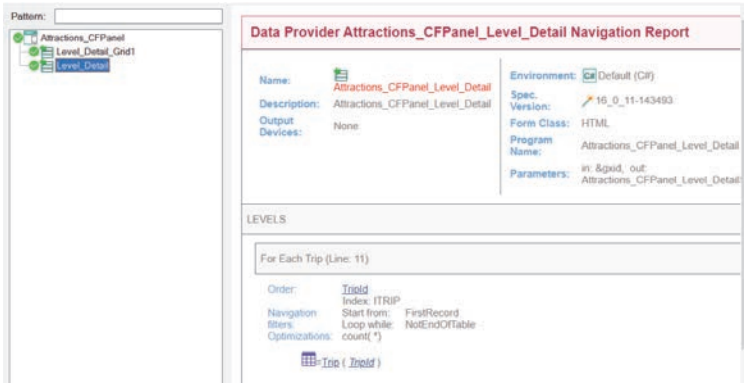
この機能が原因で、Panel オブジェクトは Web パネルのようにはプログラムできません。

イベントのトリガーとベーステーブルの決定については別の章で詳しく説明していますが、ここでは、この例を機能させるためにプログラミングを変更します。

正しいソリューション

```
Event Load
    &Trips = Count(TripDate)
Endevent

Event Refresh
    &TotalTrips = 0
    For each Trip.Attraction
        &TotalTrips += 1
    Endfor
Endevent
```



ソリューションは、Refresh イベントに、TripAttraction テーブルにアクセスして全体のツアーの合計数を計算する For Each コマンドを含めることです。Load イベントでの計算を削除することを忘れないでください。

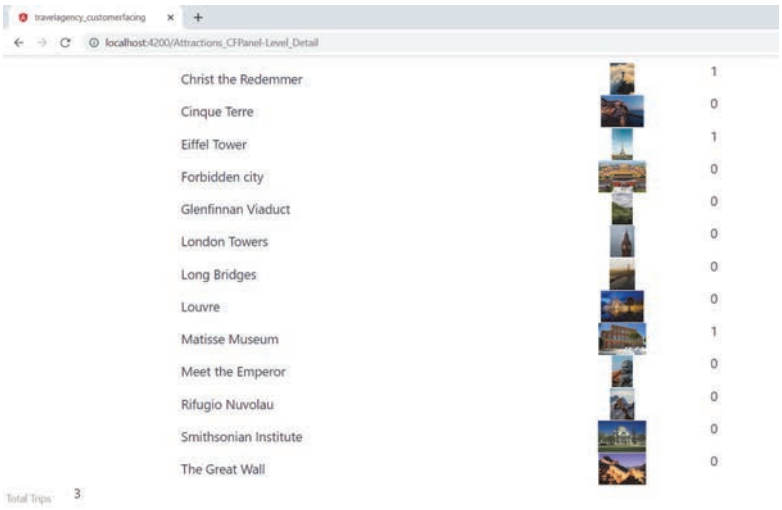
Refresh イベントで For Each コマンドを使用して更新を行う理由は、固定部分がロードされたとき、つまりグリッドがロードされる前に、Refresh イベントがトリガーされるためです。
したがって、固定部分をロードすると、&TotalTrips の値は正しい値になり、その後でのみグリッド部分が更新されます。

これを実行します。

なお、Level_Detail ノードに対応するナビゲーション表示には、TripAttraction テーブルへのナビゲーションを行う For Each コマンドが含まれています。

GeneXus

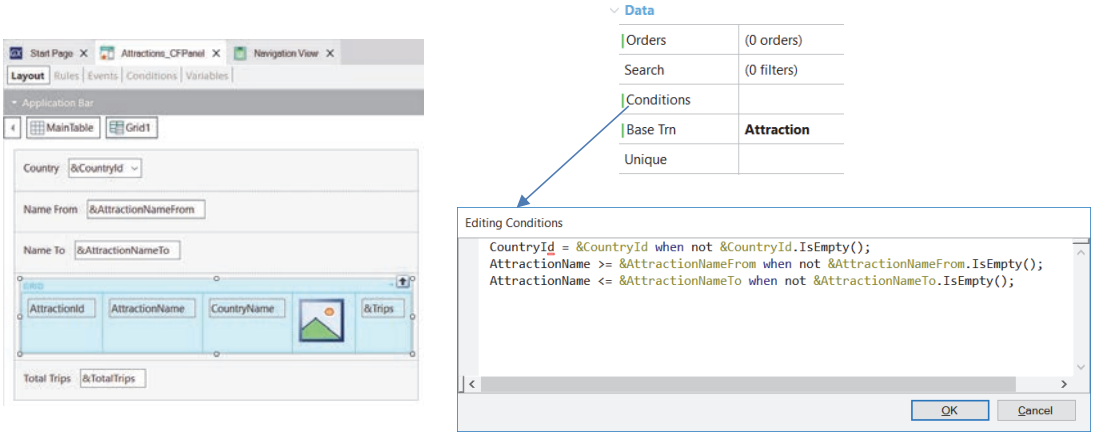
全体のツアーの合計数が正しく計算される状態での実行



ブラウザーで、全体のツアーの合計数が正しく表示されていることが分かります。

GeneXus

国名と観光名所名によるフィルタの追加

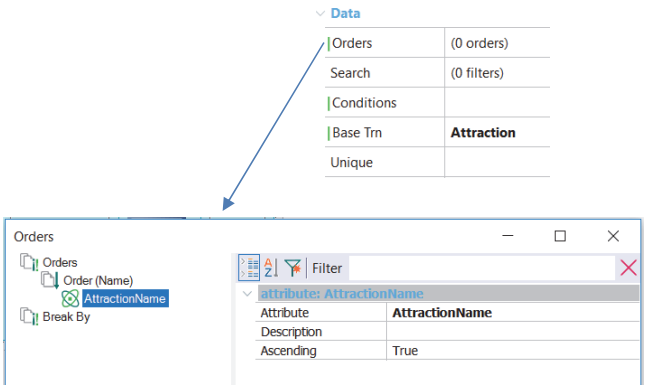


Work With Attractions パネルに必要なものを揃えるには、国の識別子と観光名所の名前によるフィルタを追加する必要があります。

グリッドの上に &CountryId 変数をダイナミック コンボ ボックスとして追加し、[Item Descriptions] プロパティに CountryName を指定し、[Empty Item] プロパティを True に設定します。また、&AttactionNameFrom 変数と &AttractionNameTo 変数を追加して、観光名所を名前でフィルタリングします。

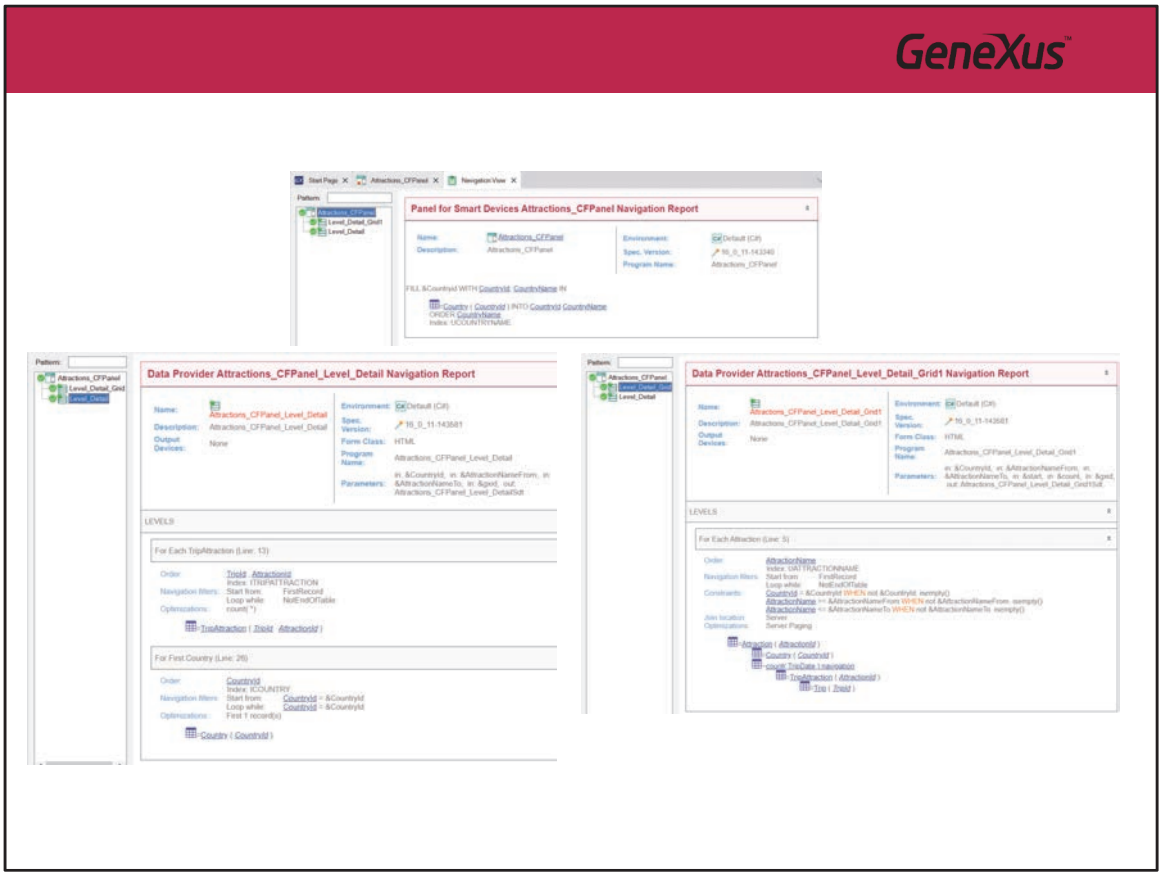
Web パネルでは、グリッド条件にフィルタを追加しますが、ここでも同じことができます。国でフィルタリングし、さらに観光名所の名前でフィルタリングする条件を定義してみましょう。

観光名所のリストを名前で並べ替える



また、[Order] プロパティを使用して、グリッドを観光名所の名前で並べ替えるように設定できます。その場合、[Orders] を右クリックし、名前 (「Name」など) を付けてから、もう一度右クリックして AttractionName 項目属性を開きます。

複数の項目属性による並べ替えを定義することも、異なる基準による並べ替えを作成することもできます。[Break By] では、グリッドのレコードをグループ化することができます。たとえば、観光名所を国名でグループ化する場合などに使用します。

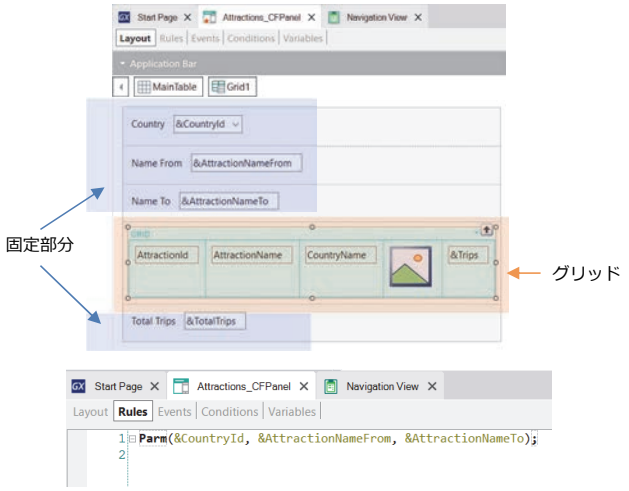


パネルを右クリックして [ナビゲーションを表示] を選択すると、パネル自体のナビゲーション表示で、ダイナミック コンボ ボックスに入力するために Country テーブルへのアクセスが行われることが分かります (「FILL &CountryId WITH CountryId, CountryName IN」と表示されます)。

Level_Detail ノードのナビゲーション表示には、データベースからデータを取得するためにサーバーで呼び出されるデータプロバイダーのレポートが表示されます (GeneXus によって自動的に作成されますが、ナレッジベースには表示されません)。これは、パネルの固定部分をロードするために必要になります。このデータプロバイダーにより、サーバーで Start イベントと Refresh イベントがトリガーされます。リストには、Refresh イベントでプログラムした For Each コマンドが表示されます。このコマンドは、TripAttraction テーブルにアクセスしてツアーの合計数を計算するものです。また、Country テーブルへのアクセスが行われて、ダイナミック コンボ ボックスで選択された CountryId によってフィルタリングされることも分かります。

グリッドをロードするデータプロバイダーに対応するナビゲーション表示を見ると、観光名所が AttractionName の順序で照会されること、そして制約として定義済みのフィルタが表示されていることが分かります。グリッドにベーステーブルが存在する場合は、このデータプロバイダーにより、Web パネルの場合と同様に、ロードするグリッドの各行に対して内部的に Load イベントが 1 回実行され、情報がパネルに返されて表示されます。

フィルタを適用してコンテンツを再表示する




```
Event Load
    &Trips = Count(TripDate)
Endevent

Event Refresh
    &TotalTrips = 0
    For each Trip.Attraction
        &TotalTrips += 1
    Endfor
Endevent

Event &CountryId.ControlValueChanged
    Grid1.Refresh()
Endevent

Event &AttractionNameFrom.ControlValueChanged
    Grid1.Refresh()
Endevent


Event &AttractionNameTo.ControlValueChanged
    Grid1.Refresh()
Endevent
```

前に説明したとおり、パネルの固定部分はグリッドのロードから切り離してロードされます。サーバーでサービスとして公開されているさまざまなデータプロバイダーが呼び出され、データベースへのアクセスを通じて各部分の情報が取得されます。

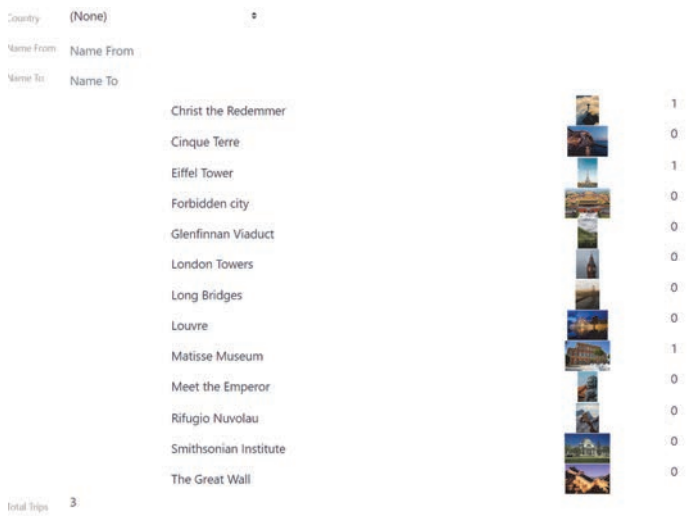
フィルタの値を変更した場合は、グリッド情報を再表示する必要があります。そのためには、グリッドの Refresh メソッドを追加する必要があります。これにより、サーバーの Refresh イベントと Load イベントがトリガーされるようになります。次に、グリッドが再ロードされ、プログラムされた条件が適用され、想定どおりにフィルタリングされた結果が表示されます。

グリッドの Refresh メソッドは、フィルタ変数の値を変更した後に呼び出す必要があるため、各変数の ControlValueChanged イベントを使用してメソッドを呼び出します。このようにすると、値を変更した後、フィールドを離れると、対応するイベントがトリガーされ、最終的にグリッドのコンテンツが再表示されます。

ただし、このアーキテクチャでは、ページのロード回数をできるだけ少なくすることが目的であるため、データキャッシュ、つまり以前に保存された情報の取得が常に優先されることを考慮する必要があります。新しいデータが必要であることをサーバーに認識させるには、サーバーに送信される URL を変更することによって、それが新しいページであり、対応する情報を取得してクライアントに送信する必要があることを認識させます。
そのために、フィルタで使用される変数の値を含む Parm ルールを追加しました。このようにすると、変数の値が変更された場合に、ページが新しいデータで更新されます。

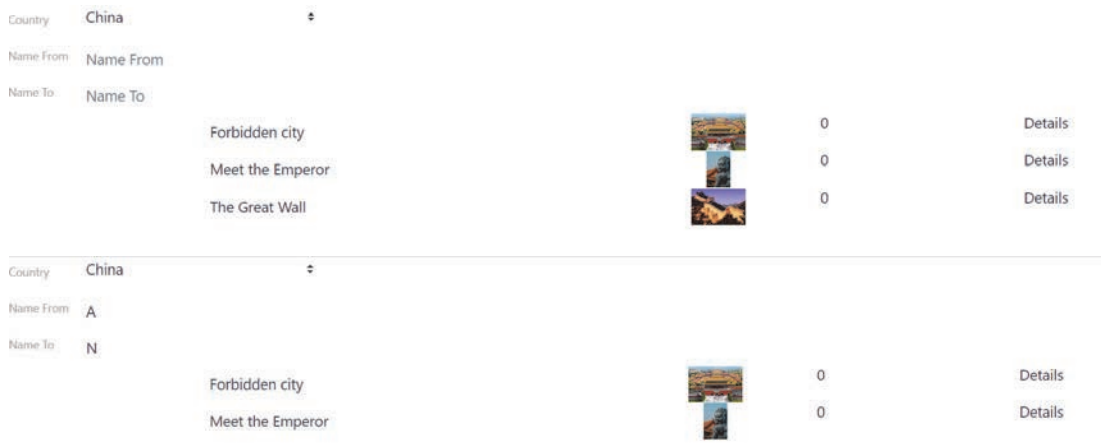
ここまでの内容をテストするため、ここで実行してみましょう。

新しいフィルタと順序での実行



追加したフィルタが上部に表示され、観光名所が想定どおりに名前で並べ替えられていることに注目してください。

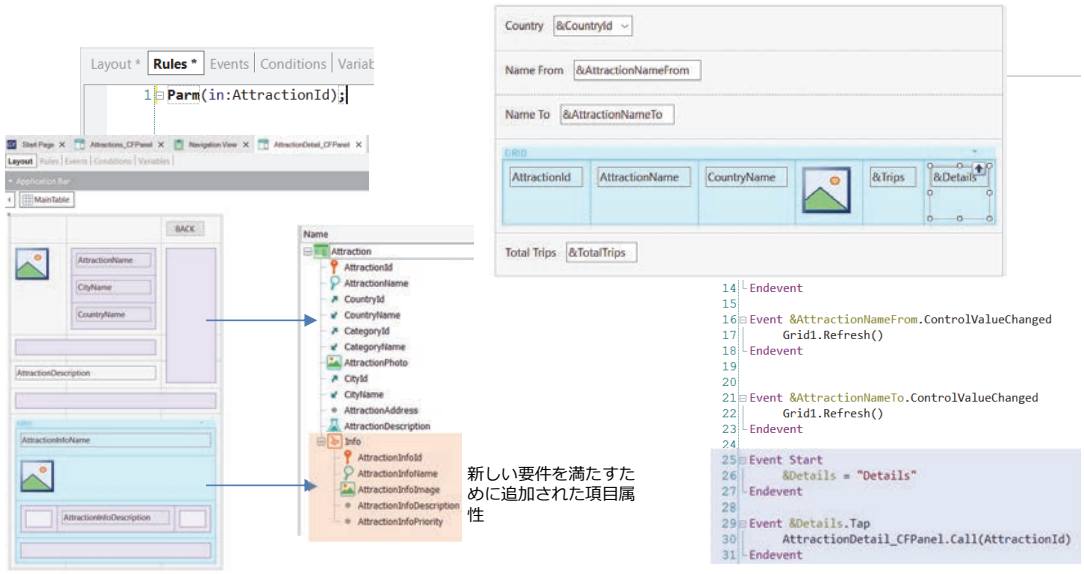
新しいフィルタと順序での実行 (続き)



中国でフィルタリングし、中国の観光名所が表示されることを確認します。

ここで、名前が A ～ N で始まる観光名所を表示するように指定すると、その条件に合致する中国の観光名所のみが表示されます。

観光名所の詳細を実装する



次に、旅行代理店の最後の要望に対応します。それは、リストに表示された観光名所をクリックすると、その観光名所の詳細が表示されるようにすることです。このために、AttractionDetail_CFPanel という別の Panel オブジェクトを使用します。これは既に作成されているものを紹介します。

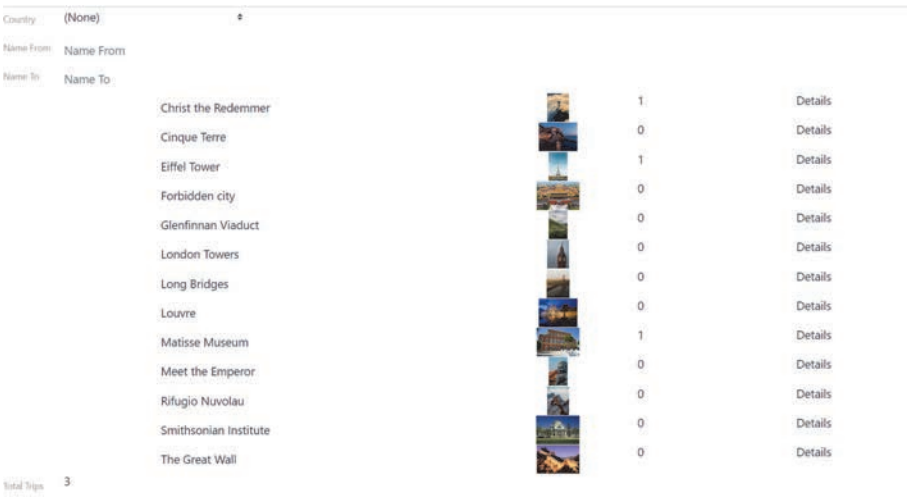
ルールでは、入力パラメーターとして AttractionId 項目属性を使用して Parm ルールを定義します。
これにより、パラメーターで渡された観光名所、つまり観光名所リストのパネルで選択した観光名所の情報のみを表示できます。

フォームに、AtractionPhoto、AtractionName、CityName、および CountryName の各項目属性を追加しました。その下に、AttractionDescription 項目属性を挿入しました。次に、観光名所の詳細を表示するために、グリッドを挿入し、AttractionsInfoName、AttractionInfoImage、および AttractionInfoDescription の各項目属性を選択しました。また、配置を整えるためにいくつかのテーブルコントロールを挿入しました。
さらに、観光名所のリストに戻れるように、Return を呼び出す [BACK] ボタンを右上隅に追加しました。

次に、Attractions_CFPanel パネルに移動し、Character タイプの &Detail 変数をグリッドに追加します。Start イベントで「Details」というテキストを割り当てます。
次に、グリッド変数を右クリックして [イベントへ移動] を選択し、タップで AttractionDetail_CFPanel を呼び出すよう記述し、パラメーターとして AttractionId を渡します。

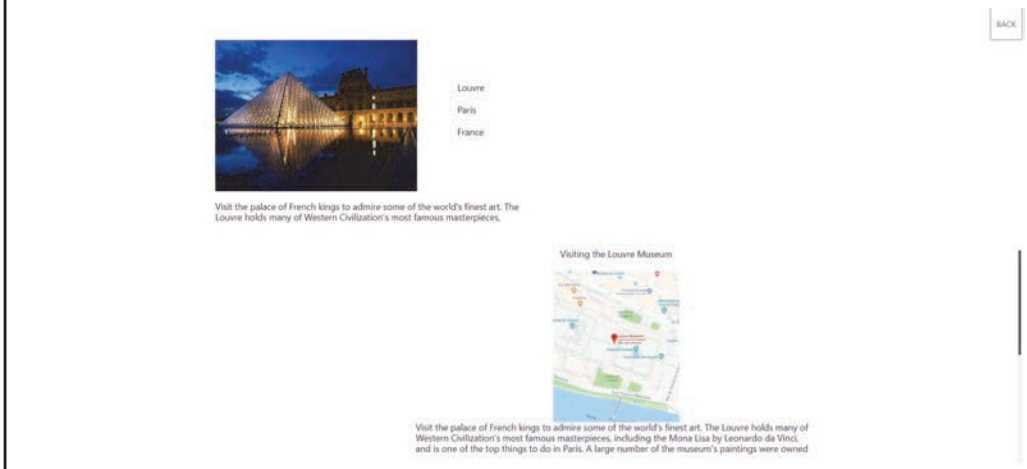
実行してみましょう。

観光名所の詳細を扱う実行



ルーブル美術館の詳細情報を確認します。対応する行の [Details] をクリックします。

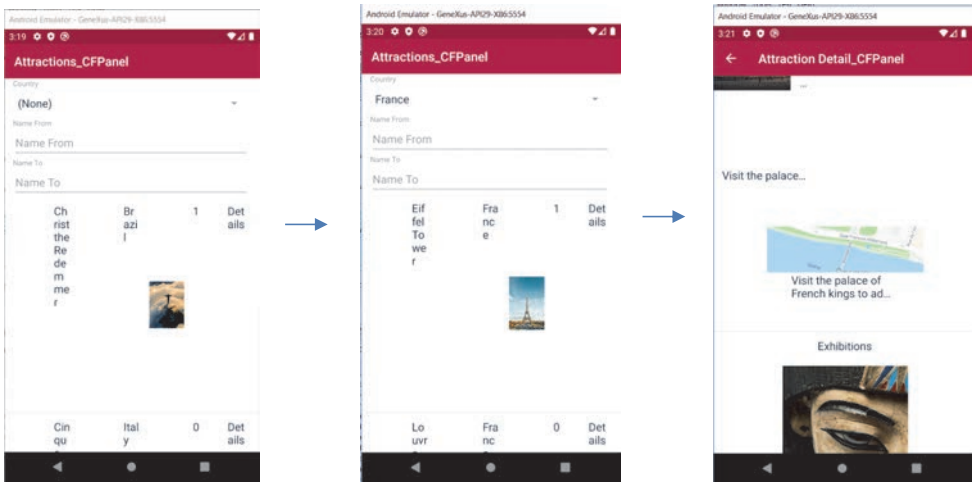
観光名所の詳細を扱う実行 (続き)



パネルが開き、観光名所の詳細が表示されます。ここでは、地図と鑑賞可能な展示の情報を見ることができます。

Android ネイティブアプリケーションの生成

Android の前提条件: http://wiki.genexus.jp/hwikibypageid.aspx?14449



この章の冒頭で触れたとおり、必要に応じて、構築した Panel オブジェクトを使用してモバイルアプリケーション用の画面を生成することができます。これを試してみましょう。

Android の言語で生成できるようにするには、画面に表示されている Wiki の記事に記載のソフトウェアをインストールする必要があります。

KB エクスプローラーで [フロントエンド] ノードをクリックし、[Generate Android] プロパティを True に設定し、[Main Platform] プロパティを Android に設定します。
次に、メインオブジェクト Attractions_CFPanel を実行します。

Android エミュレーターが開き、Attractions_CFPanel パネルに観光名所のリストが表示されることが分かります。
パネルをデザインしたときには、デスクトップコンピューターの画面上で実行される Web システムの画面を想定していたので、スマートフォンの画面サイズに合わせてレイアウトを定義する必要があるのは明白です。

しかし、ここではデザインのことは省略し、正しく機能するかどうかを確認しましょう。フィルタでフランスを選択すると、フランスの観光名所のみが表示されます。ここで、ルーブル美術館の [Details] をクリックすると、美術館の詳細を示す画面が開き、想定どおりに地図と展示の情報が表示されます。

この章では、最初に Panel オブジェクトについて理解し、Angular フレームワークでアプリケーションを生成しました。また、作成したオブジェクトを使用して、ネイティブの Android 言語でアプリケーションを生成できることを確認しました。

次は、クライアントレベルおよびサーバーレベルでのイベント管理について詳しく学習します。