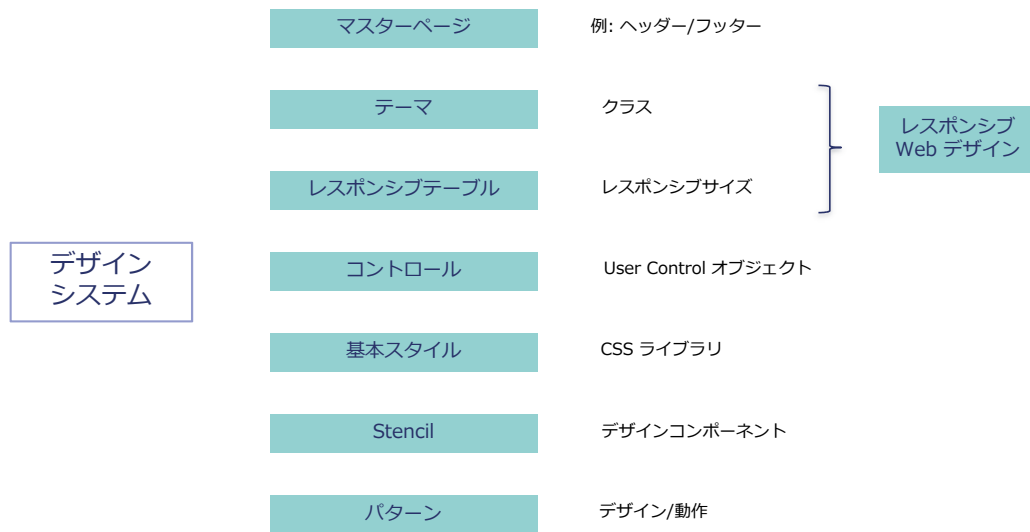


# GeneXus のデザインシステム

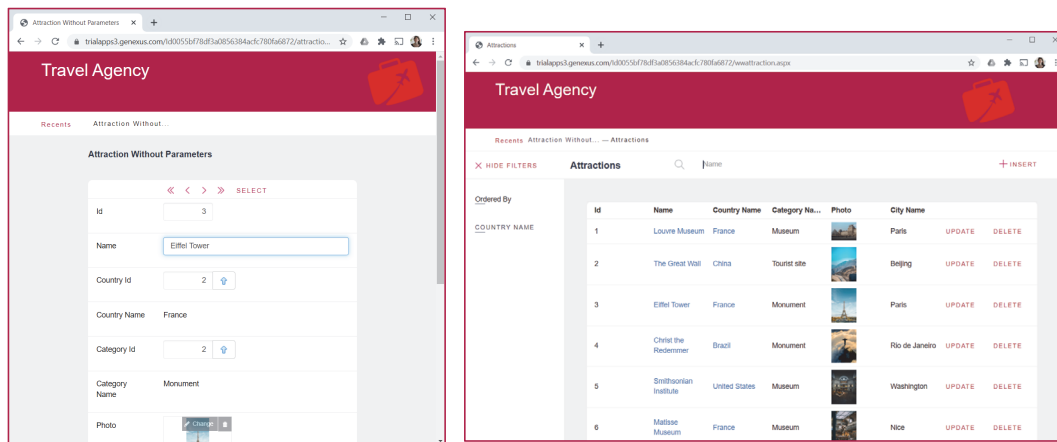
既定

*GeneXus*<sup>™</sup>

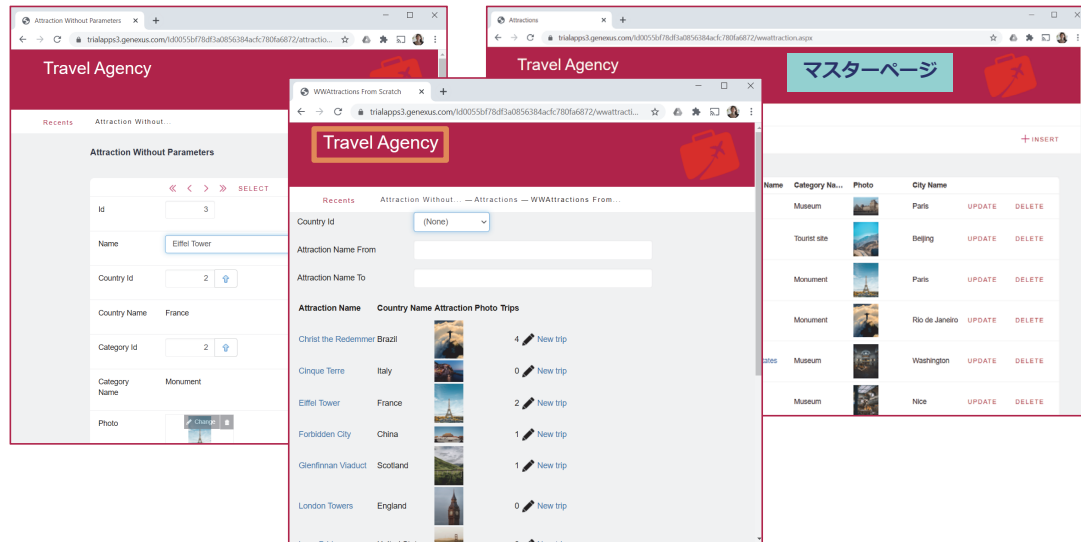


この章では、アプリケーション開発におけるデザインシステムの実装に関わる GeneXus の要素についてそれぞれ簡単に説明します。

## バックオフィス

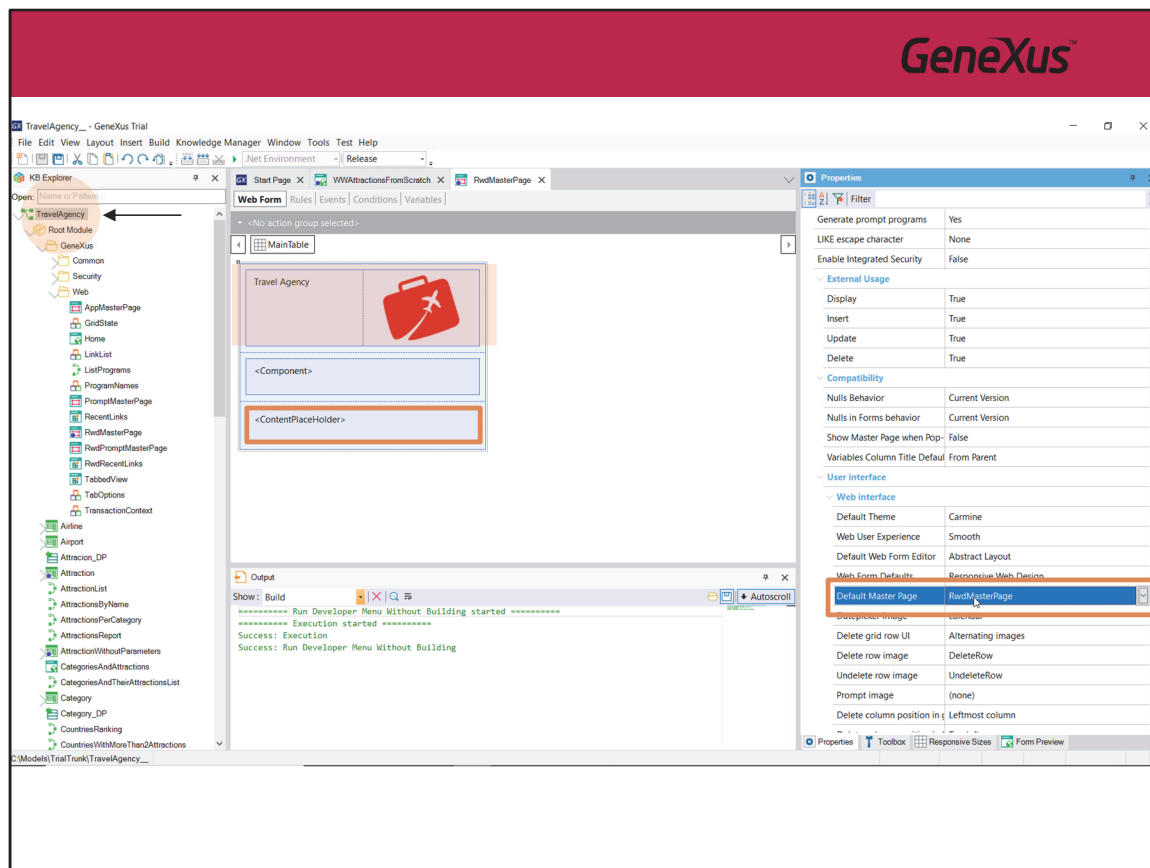


使用しているトランザクションや、Work With パターンによって作成されるパネルに注目すると、人が意識しなくても一貫性や統一性を維持するエレメントが存在することが分かります。つまり、既定のデザインシステムの観点で、GeneXus により、基本的なことであれ何かしらの処理があらかじめ行われていることを意味します。



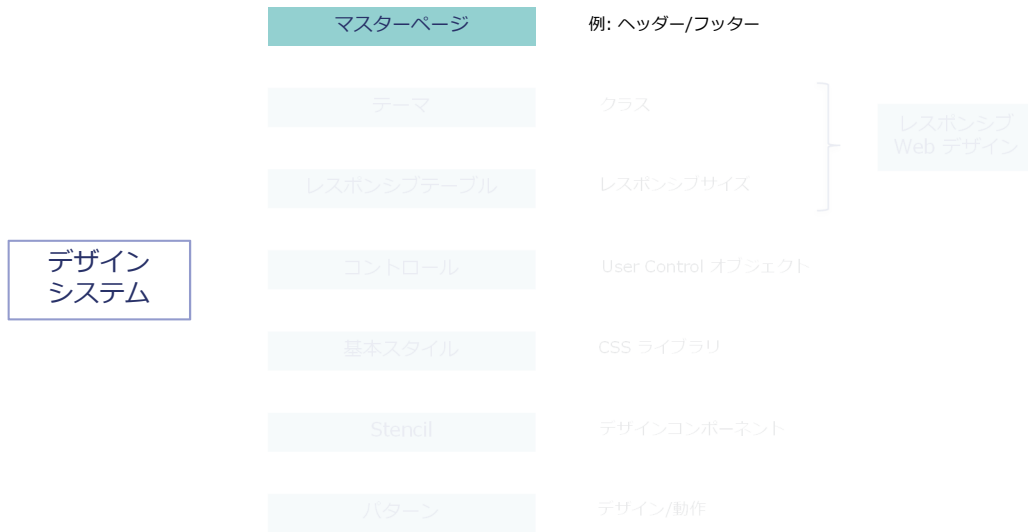
たとえば、すべての画面に共通のヘッダーがあります (このスライドの Web パネルのように一から作成した画面でも同様です)。この例ではカスタマイズを行い、既定のテキストを「Travel Agency」に変更し、**画像を別のものに置き換えました。**



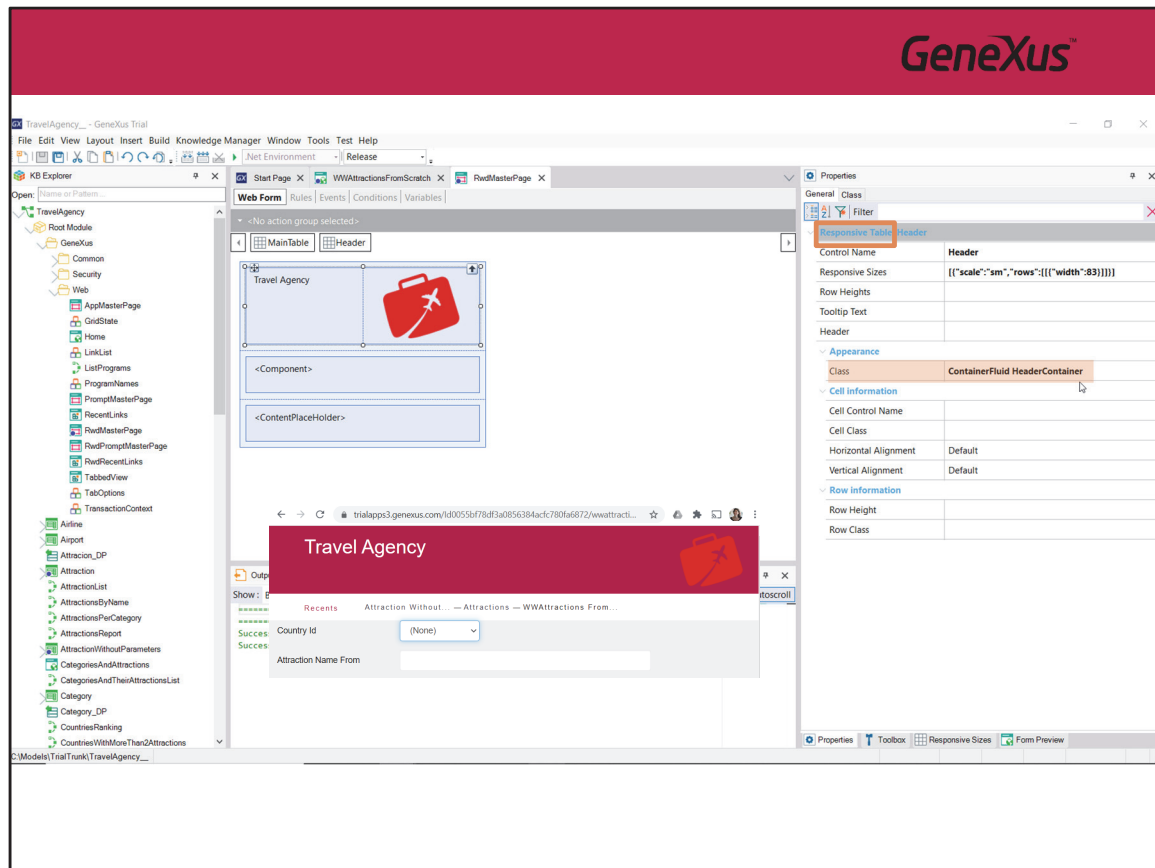


これを実現するには、Web の Master Page オブジェクトを使用します (ここで変更内容を確認できます)。実行時において、アプリケーションの各画面は、この ContentPlaceHolder コントロールにロードされることを学習してきました。

ナレッジベースのバージョンレベルでは、既定でこの事前定義済みのマスターページが指定されており ([Default Master Page] プロパティ)、前述のように、作成するすべての Web オブジェクトに適用されます。

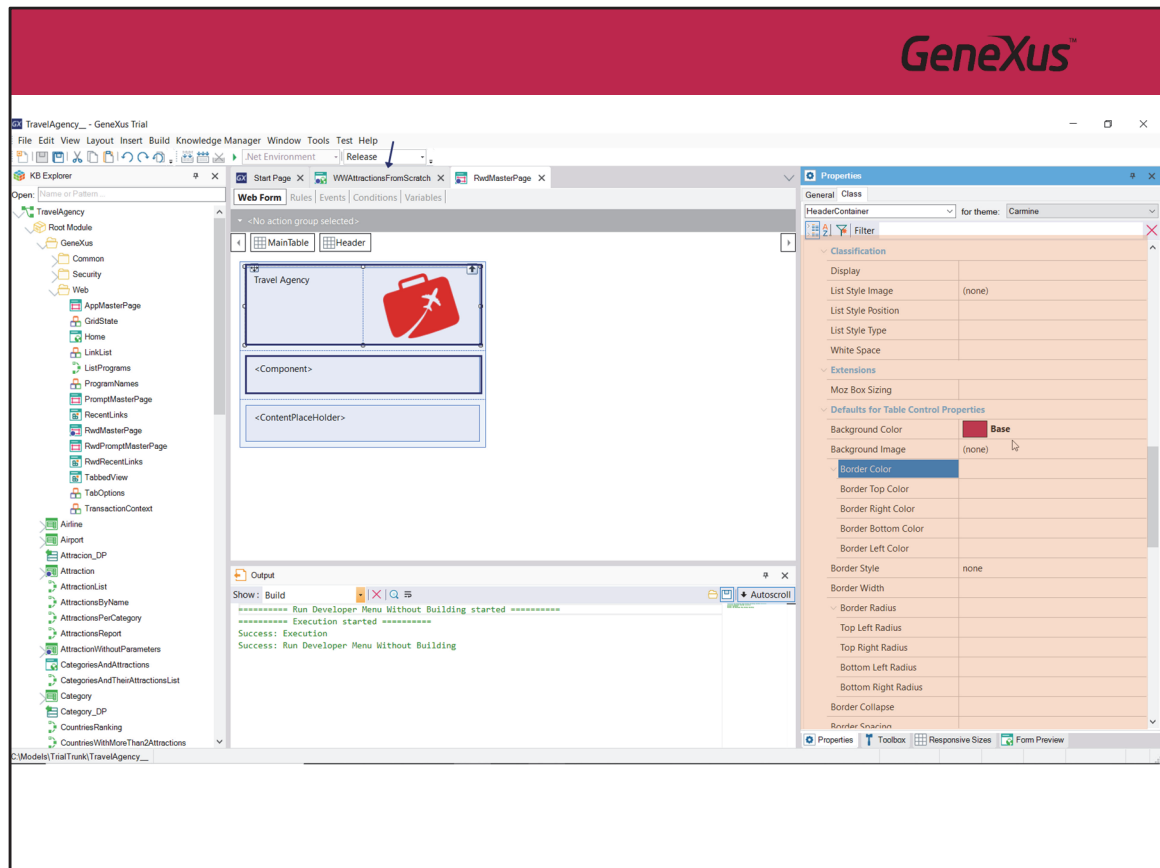


このように、GeneXus には、ヘッダーやフッターなど、ページ間で共有する共通の情報を一元化するためのオブジェクトが用意されています。



加えて、このアプリケーションでは、ベースカラーとして赤色を使用しています。この色はヘッダーだけでなく、ユーザーに提示するボタンやその他のアクションにも使用されています。この設定はどこで行うのでしょうか。

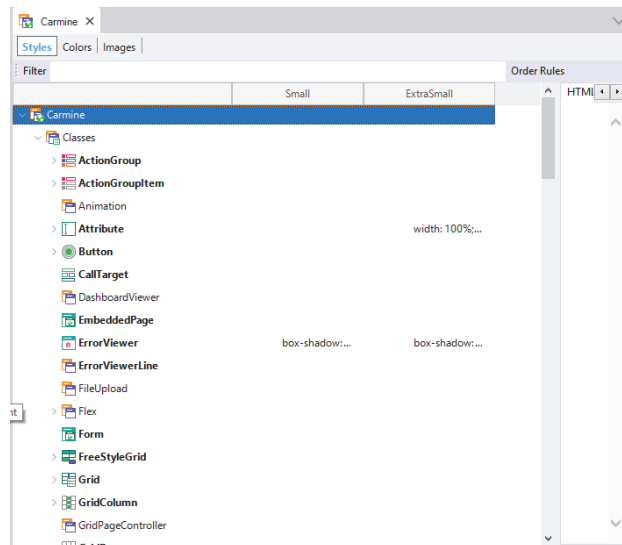
このテーブルの背景は実行環境では赤色であることが分かっていますが、このコントロールの [全般] プロパティ ウィンドウ タブではそのように表示されていません。このコントロール (この例ではレスポンス テーブル タイプ) には、クラスがペアで定義されています。これらが「Appearance」グループの下にあるのは偶然ではありません。



[クラス] タブを見ると、このテーブルに関連付けられている 2 目目のクラスのプロパティが編集されていることが分かります。ここで、背景色の赤色が定義されていました。

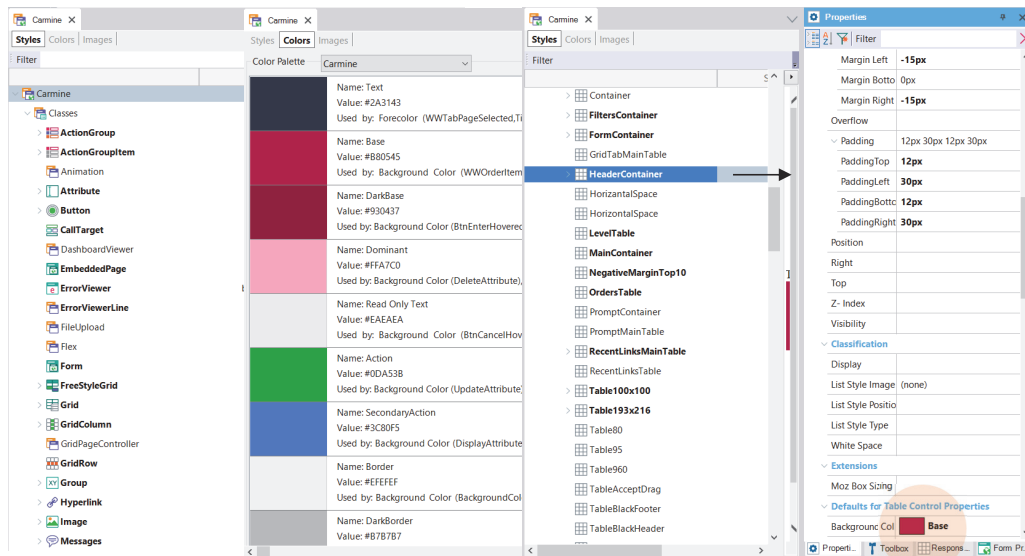
クラスは、このタイプのコントロール (この例ではテーブル) のデザインを一元化するのに役立ちます。この例では、同じフォーム内または別のフォーム内にあるほかのテーブルで、同じ定義を共有できることが分かります。たとえば、これらのコントロールの背景色を赤から青に変更する場合は、コントロール (つまりテーブル) ごとに変更する必要はなく、クラスで直接変更するだけで、そのクラスに関連付けられているすべてのコントロールに変更が反映されます。

## テーマ



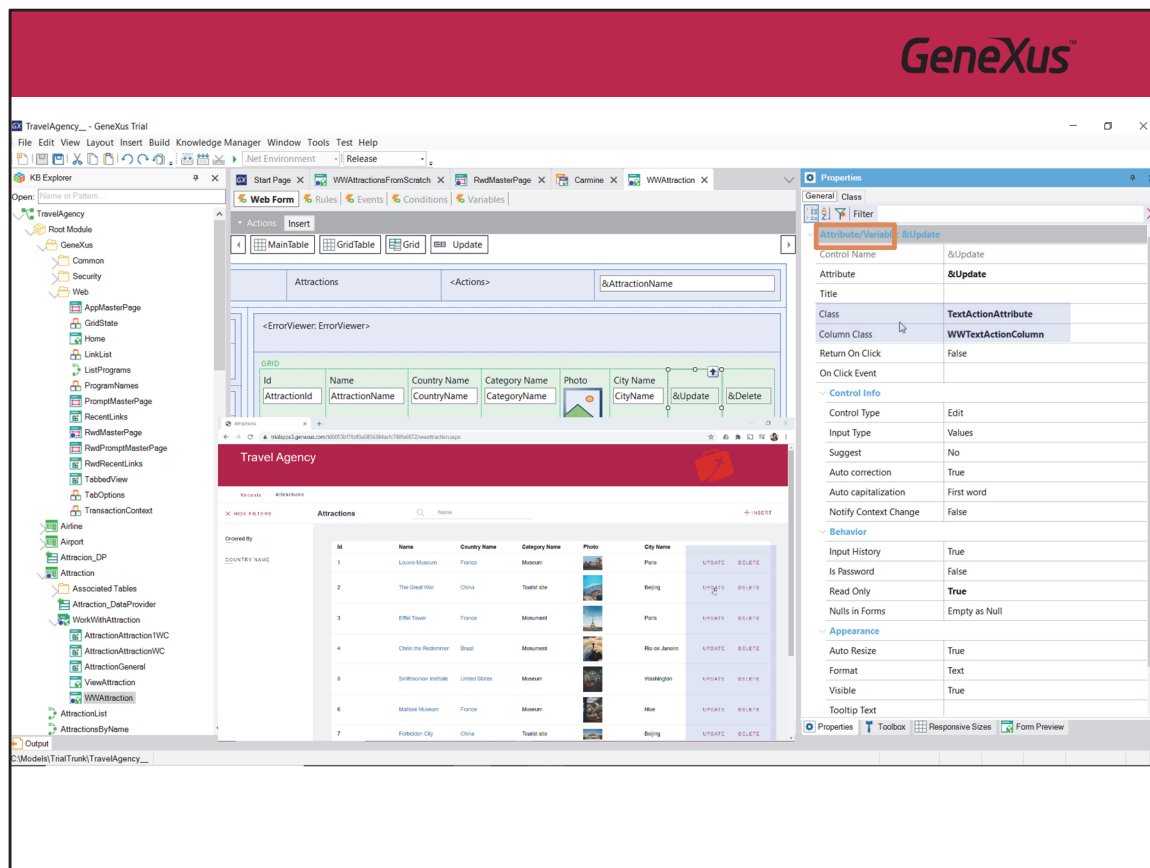
すべてのクラスの設定はどこで行うのでしょうか。それは Theme オブジェクト内です。

事前定義済みのマスターページと同様に、Carmine という事前定義済みのテーマが用意されています。これはこれまでに何度か Web パネルに関連付けられているのを見てきました。



こちらを開くと、コントロールタイプ別にクラスがグループ化されています。すべてがここに一元化されています。[Colors] エレメントには、テーマと同じ名前のカラーパレットが表示されます。ここでベースカラーは設定されています。

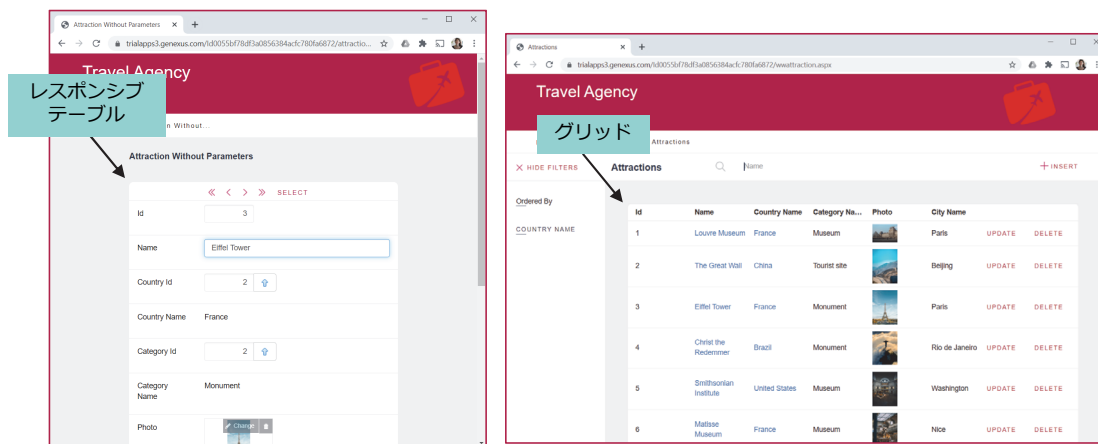
Table クラスのサブクラスを確認すると、HeaderContainer があります。これはマスターページのテーブルに関連付けられていた 2 つのクラスの中の 1 つです。一方への変更内容はもう一方にも反映されます。先ほど見たのはこのページへのショートカットでした。



Work With パターンの更新/削除のオプションは実行環境ではベースカラーで表示されますが、これを見ると、コントロールのタイプが項目属性/変数であることが分かります。また、割り当てられているクラスはパターンによって事前定義済みです。ここでプロパティを編集することができます。その場合、変更は中央の Theme オブジェクトに対して行うことになり、結果として、このクラスが割り当てられているほかのすべての項目属性/変数コントロールも変更されることになります。

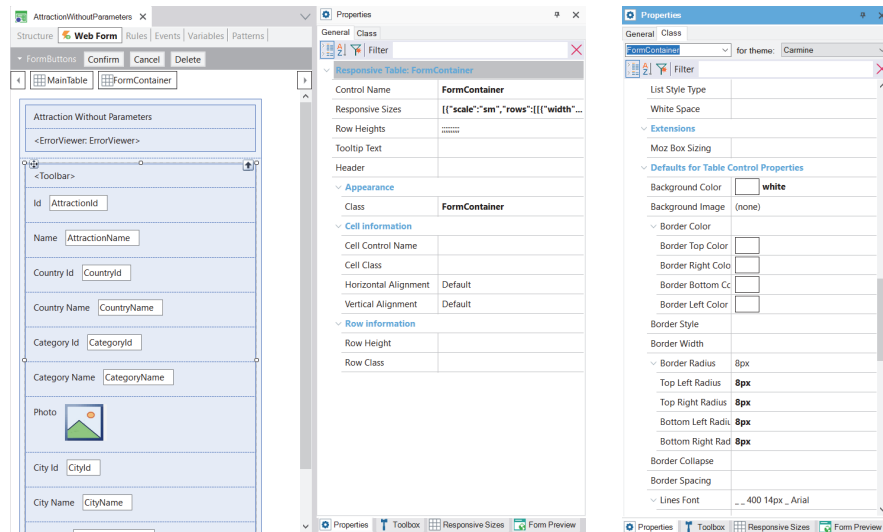
つまり、フォーム内のすべてのコントロールには、単独で設定可能な個別のプロパティに加えて、割り当てられているクラスに含まれるプロパティもあります。後者のプロパティは、デザインの側面に関連するもので、同じタイプの複数のコントロールに共通です。

コントロールがグリッド列でもあるこの特殊なケースでは、それに関連する別のクラス、つまり列のデザインや動作を規定するクラスが表示されることに留意してください。

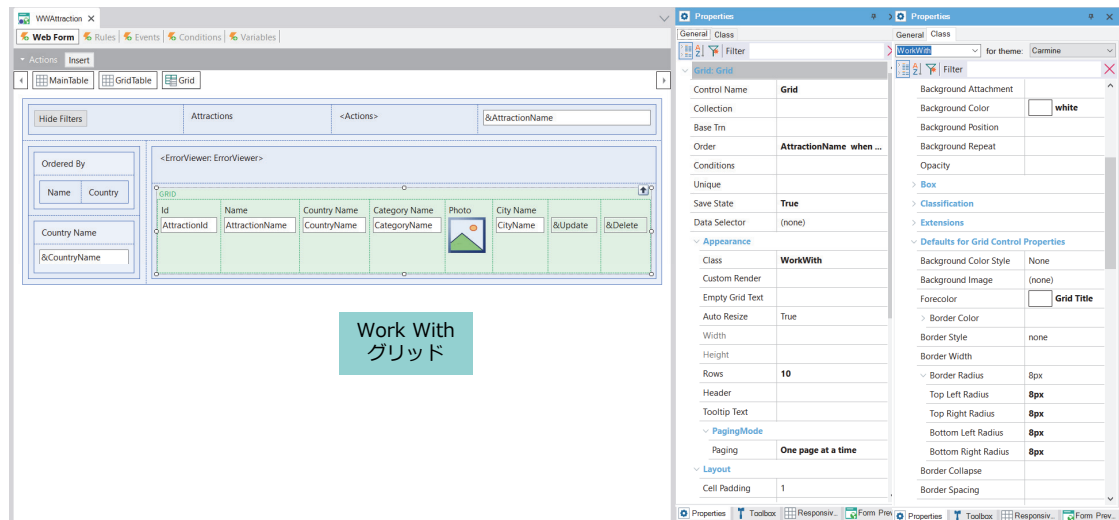


トランザクションおよび Work With パネルの事前定義済みデザインをもう一度見ると、主要なデータは灰色地に白で表示され、角が丸みを帯びていることが分かります。

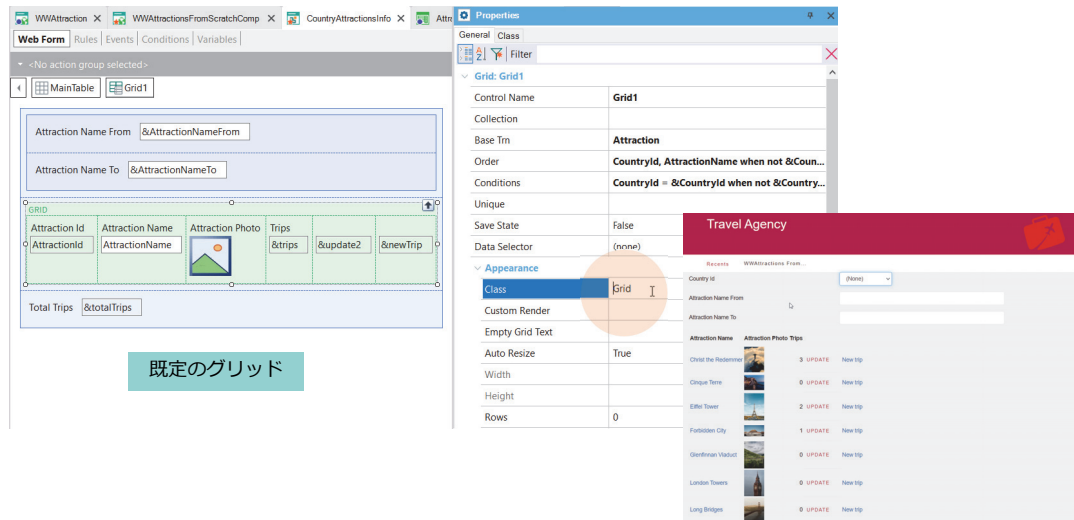


レスポンス  
テーブル

前ページのスライドの画面のようになるのは、トランザクションの Web Form 上のテーブルに、白の背景色と 4 つの角の半径、[Border Radius] を指定したクラスが割り当てられているためです。

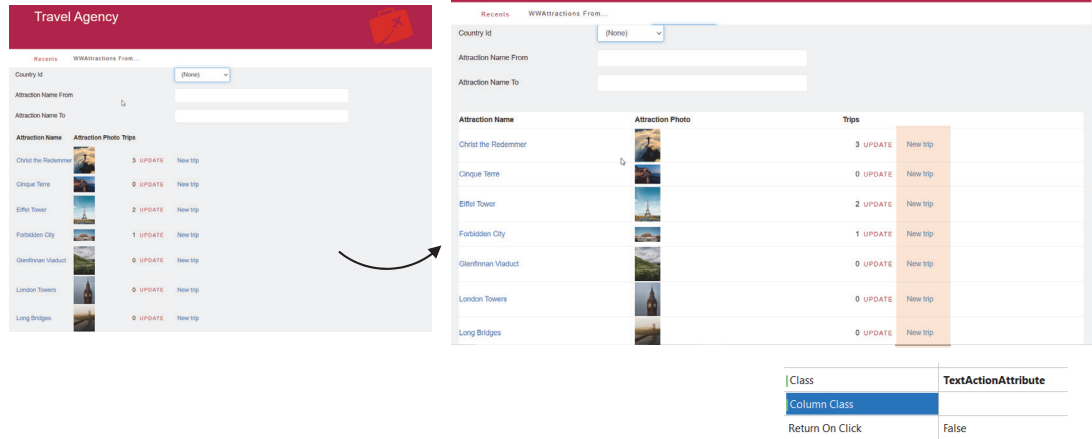


Work With の一覧画面のグリッドには、同様に白の背景色と同じ半径値を指定した別のクラスが割り当てられています。



一方、独自に作成した Web パネルのグリッドには、このような設定を一切行っていない Grid というクラスが割り当てられています。そのため、実行時には、すべての列が横に並んで表示され、背景色は既定の灰色になります。

## クラス: WorkWith

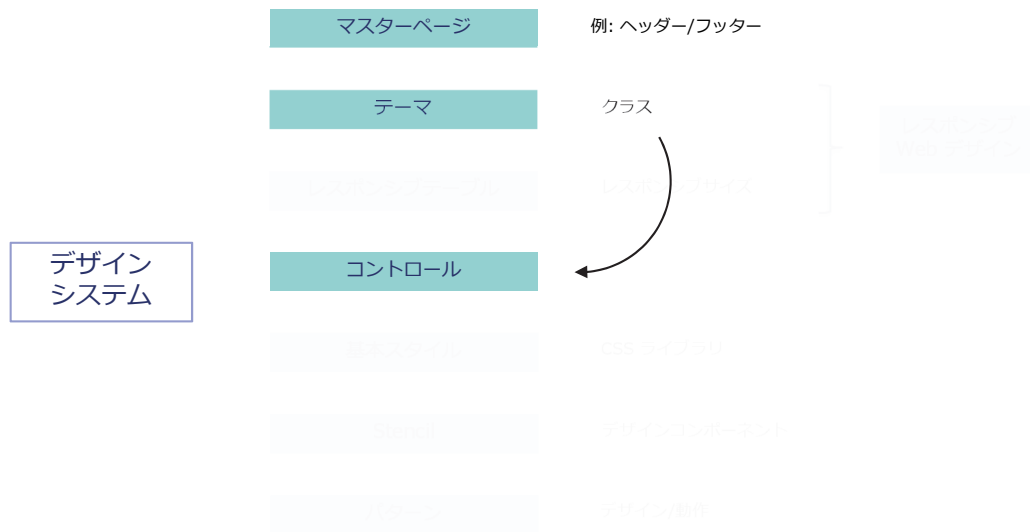


The screenshot shows the 'Travel Agency' application. On the left, there is a 'WorkWith' class interface with a list of attractions. Each attraction has an 'UPDATE' button. An arrow points from this 'UPDATE' button to the 'UPDATE' button in the table on the right. The table on the right is a 'Trips' table with columns for 'Attraction Name', 'Attraction Photo', and 'Trips'. The 'Trips' column has an 'UPDATE' button for each row.

Class	TextActionAttribute
Column Class	
Return On Click	False

このクラスを WorkWith グリッドと同じクラスに変更するとどうなるでしょうか。

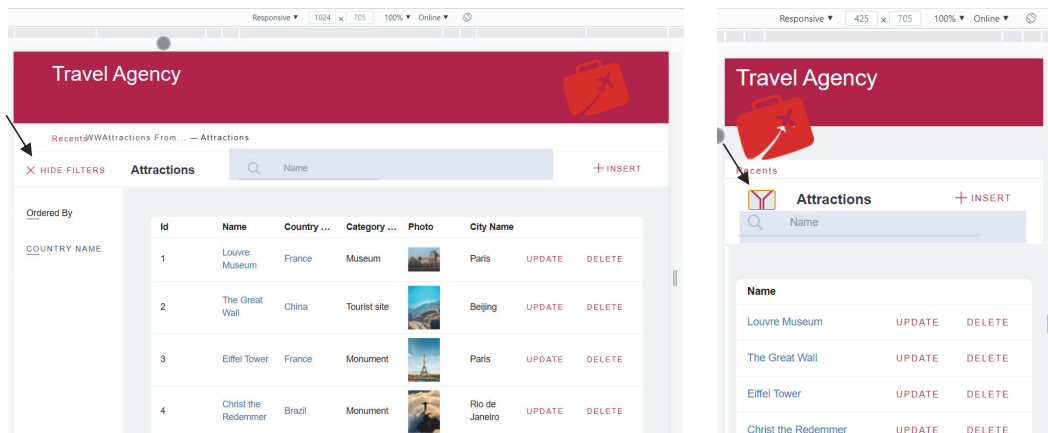
デザインシステムの一貫性を維持するために、このグリッドアクションもほかと同じようにします。更新アクションについては画像からテキストに Work With に合わせて変更しました。



ここまでの説明から、デザインシステムの大部分は Theme オブジェクトで、クラス (事前定義済みクラスと独自に追加するクラス) を使用して指定するものと推論することができます。フォーム内の各コントロールにはこれらのクラスを割り当てます。クラスによってデザインを抽象化し、特定のロジックを確立することで、各コントロールが担う役割に応じて統一化を図ることができます。

したがって、デザインにおいて、コントロールに必要なクラスを特定し、適用することが不可欠です。

## レスポンス Web デザイン (RWD)



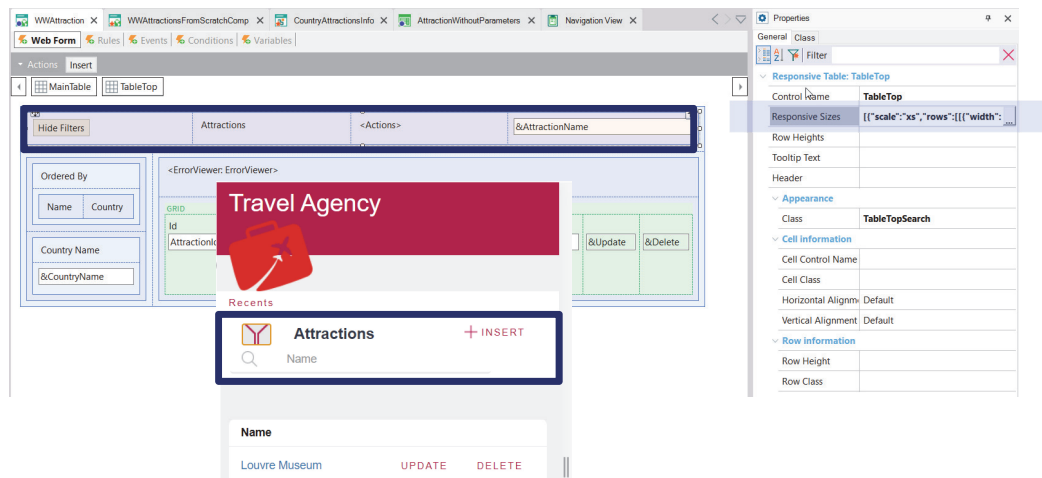
ここで、事前定義済みの設定に戻ります。一部のブラウザーには表示モードがあり、この例では F12 キーを押すと、異なる画面サイズでの表示を確認できます。

観光名所の Work With の画面をスマートフォンのサイズに縮小すると、フィルタの表示が変わります。ユーザーがフィルタとして観光名所の名前を入力するためのフィールドは、タイトルの横ではなく下に表示されます。また、最近使用したリンクは、展開された状態ではなく、ドロップダウンメニューとして表示されます。マスターページの画像は、横ではなく下に表示されます。さらに、画面幅の縮小に伴い、名前と更新/削除アクションの列を除くほぼすべての列が非表示になります。

このような動作は「レスポンス」と呼ばれるもので、それを考慮した Web デザインをレスポンス Web デザインと呼びます。現在、画面サイズに適應できないデザインは容認されにくくなっています。

レスポンスネスの一部はレスポンステーブルによって実現されます。

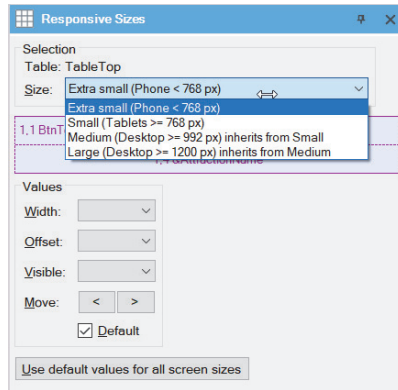
## レスポンシブテーブル



たとえば、パターンによって作成したこの観光名所の Work With のテーブルは、4 つのコントロールが含まれるレスポンシブテーブルです。最初のコントロールは追加フィルタに対応しています。2 つ目のコントロールはテキストブロックに対応しています。3 つ目のコントロールは挿入アクションに対応し、最後のコントロールはフィルタ変数に対応しています。

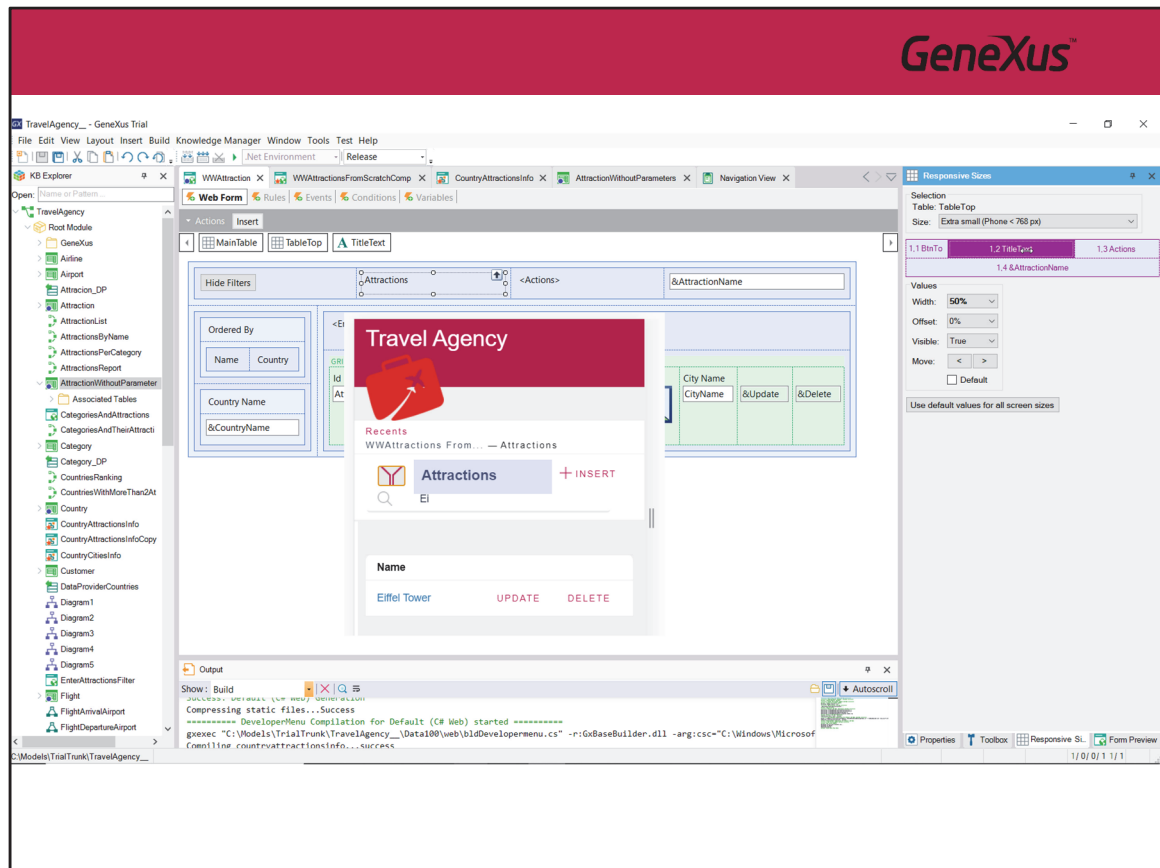
テーブルのプロパティを編集するときに、[レスポンシブサイズ] というセクションがあります。

## レスポンシブテーブル

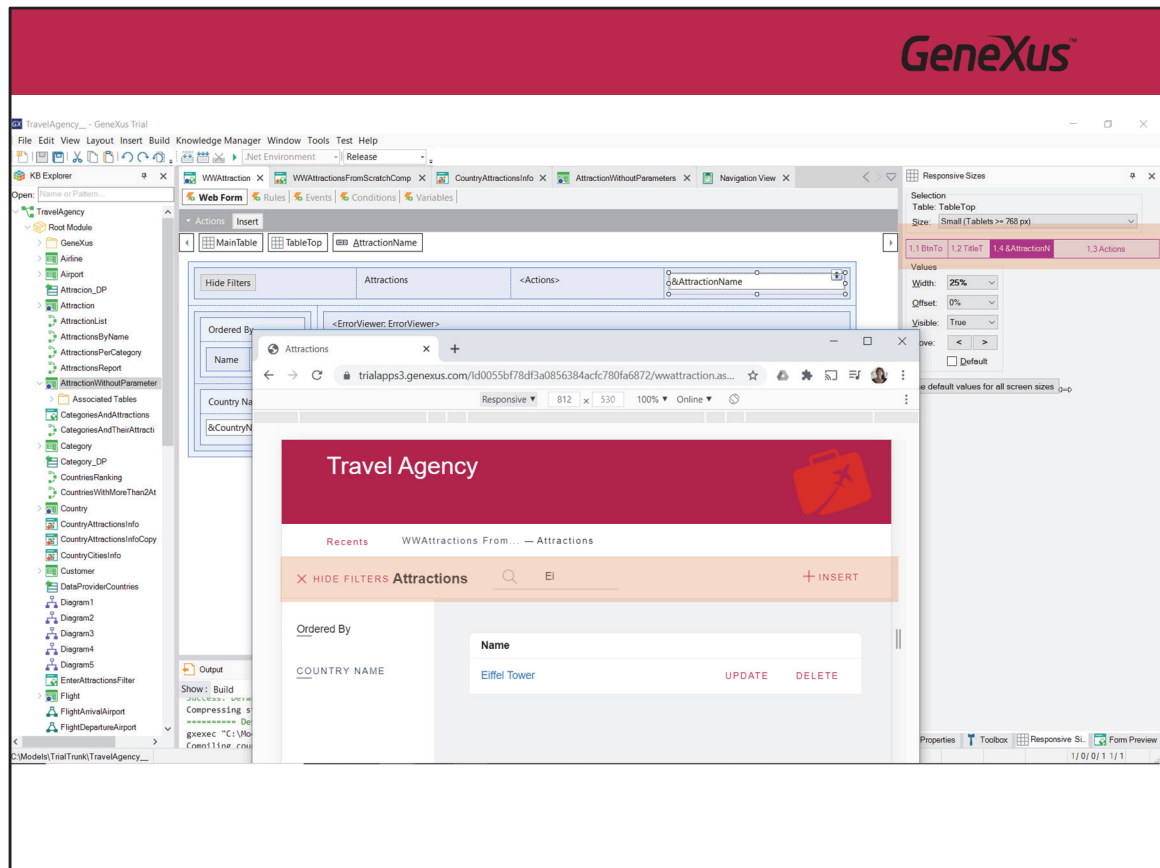


ここでは、4つの画面サイズを基準としたテーブルコンテンツの表示方法を決定します。**極小**はスマートフォン向けで、**小**はタブレット向けです。**中**は1200ピクセル未満のノートブックまたはPC画面向けで、**大**はそれ以上のサイズの画面向けです。





ここで、サイズを**極小**に設定すると、フィルタのボタンが先に表示され、画面幅の17%を占めます。その右側に、幅の50%を使用してタイトルが表示され、最後に残りの33%を使用してアクション、ここでは挿入アクションのみが表示されます。その下に、フィルタ変数が100%の幅で表示されます。

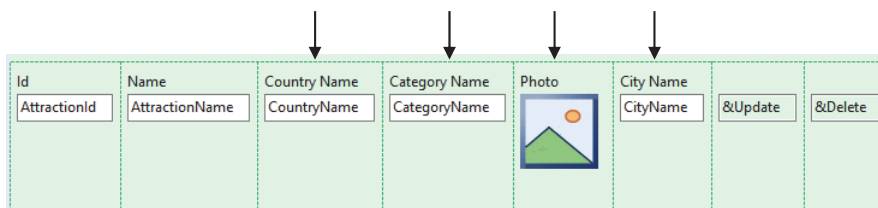



これに対して、サイズが小の場合、ボタン、タイトル、変数、アクションの順で横  
に一行で並びます。

## レスポンシブ Web デザイン (RWD)

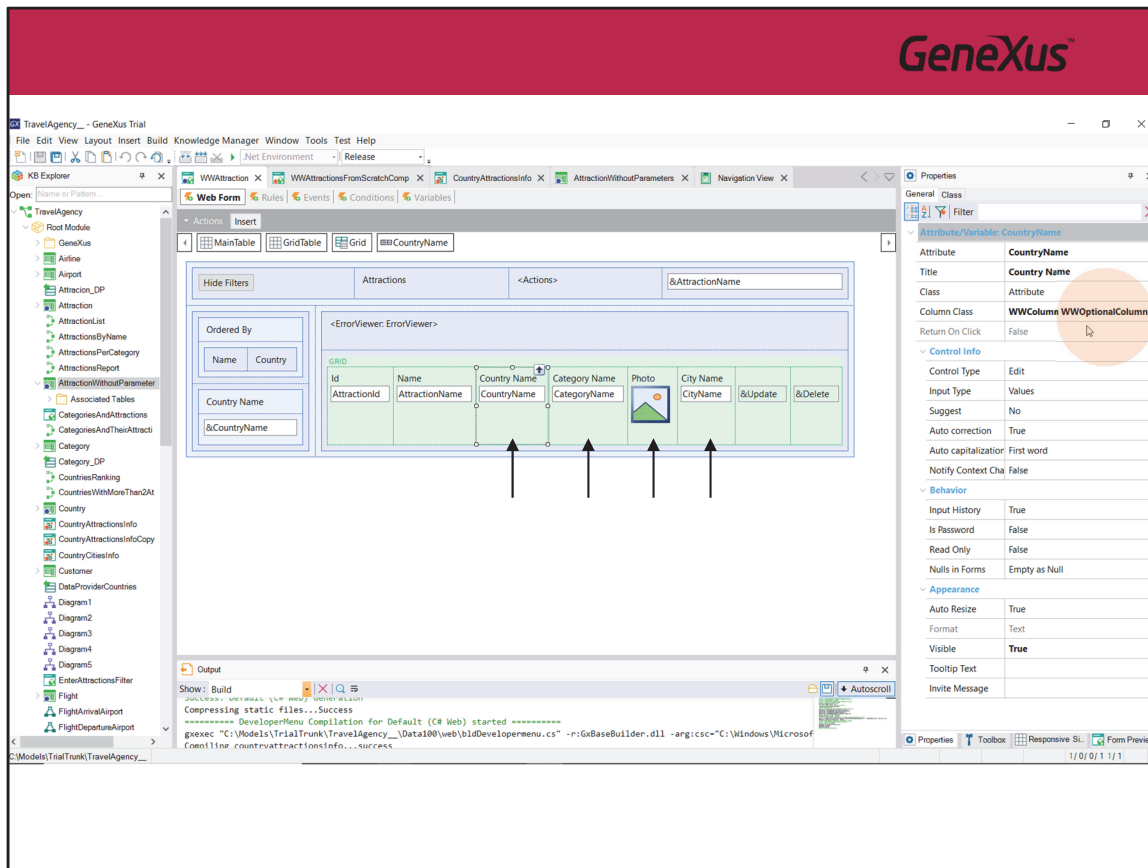
- レスポンシブテーブルのレスポンシブサイズ
- テーマルール: 小、極小、既定

例: 極小サイズまたは小サイズの場合にグリッド列を非表示



Id	Name	Country Name	Category Name	Photo	City Name		
AttractionId	AttractionName	CountryName	CategoryName		CityName	&Update	&Delete

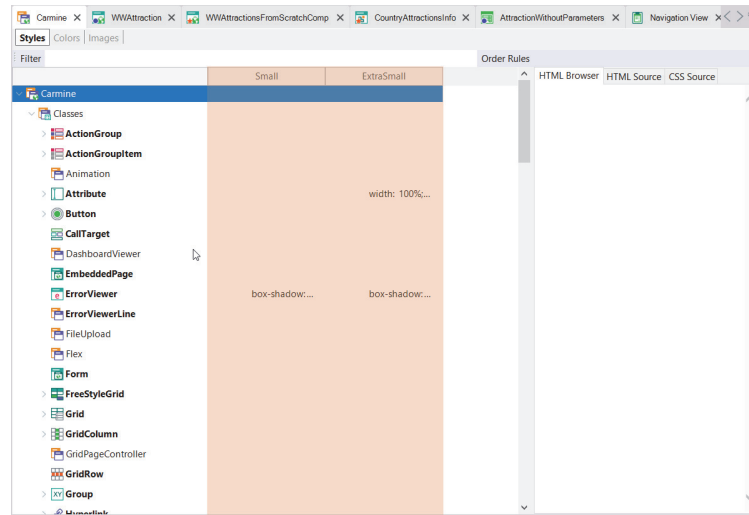
前述のとおり、レスポンシブネスの重要で概略となる部分は、これらのテーブルおよびプロパティを使用することによって実現されます。  
一方、詳細部分は**クラス**によって実現されます。



画面サイズに応じてグリッド列を非表示にする例を見てみましょう。

観光名所名の列のクラスは WWColumn ですが、その他の列についてはクラス WWOptionalColumn が追加されています。

## レスポンシブ Web デザイン: テーマルールの列

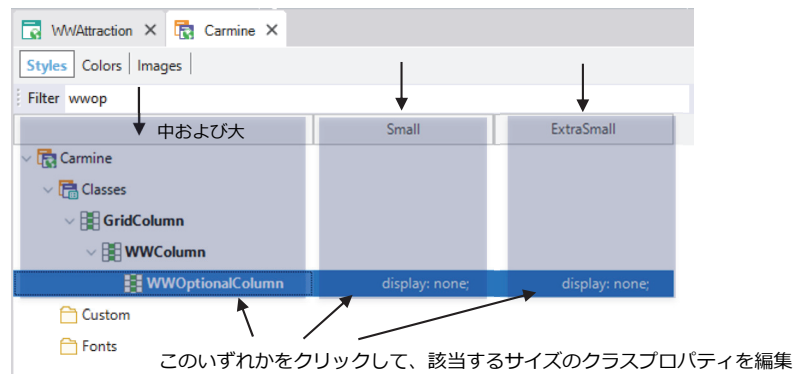


テーマを選択すると 2 つの列が表示され、それぞれの画面サイズに応じてクラスのプロパティの値を変更できます。

既定では、小と極小の 2 つのみが設定されています。既定の列の値は中および大に対応しているため、画面サイズは実際は 3 つあります。

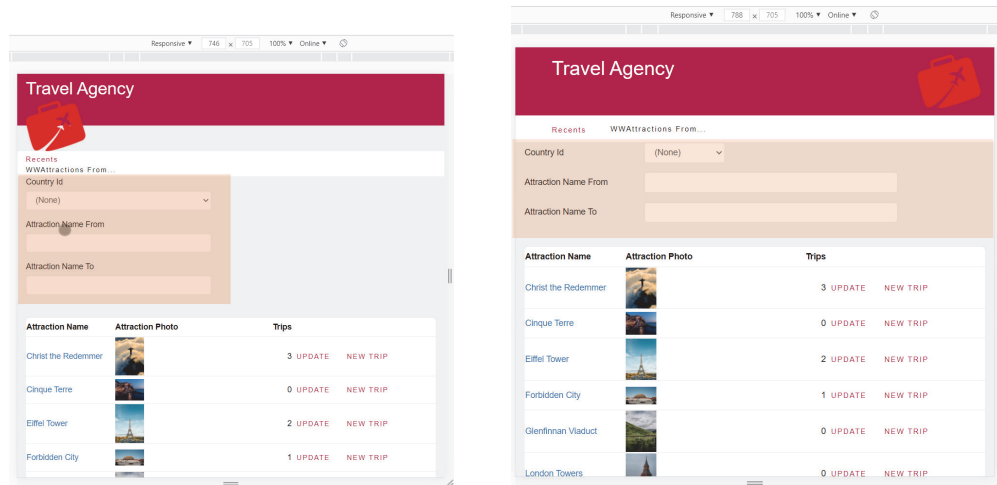
## レスポンシブ Web デザイン: テーマルールの列

- テーマルール: 小、極小、既定



したがって、このクラスの [Display] プロパティに何も指定しなくても、実行時には中と大の画面サイズに対して表示されることになります。小と極小については、値は none になっています。つまり、これらは表示されません。

## Web パネルのレスポンスブネスを高める方法



一から実装した Web パネルにはこのグリッドレスポンス動作がありません。画面サイズを縮小しても同じ列が表示されます。

唯一の事前定義のレスポンス設定はマスターページから継承されるもので、極小の場合に画像が右側ではなく下部に表示され、最近使用したリンクのリストがメニューとして表示されます。この Web パネルでは、フィルタのフィールドが幅の 100% を占め、ラベルが上に表示されていますが、小サイズ以上ではラベルが左側に配置されます。

## 小サイズまたは極小サイズで列を非表示にする方法







**Travel Agency**

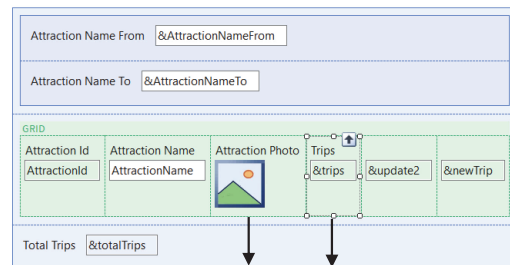
Recents WWAttractions From....

Country Id (None) ▾

Attraction Name From

Attraction Name To

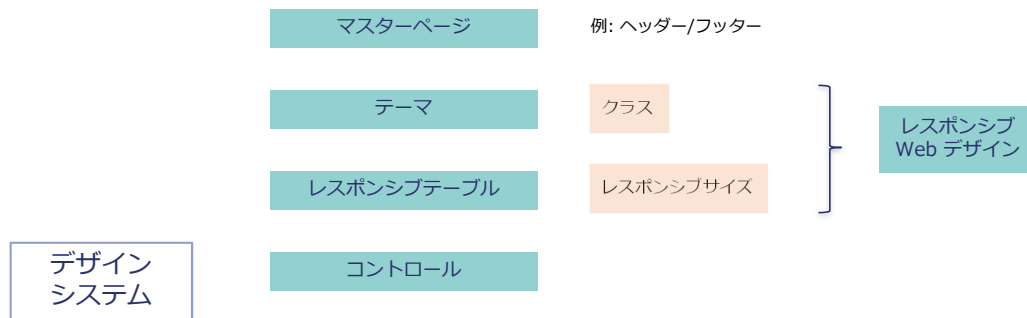
Attraction Name	Attraction Photo		Trips	
Christ the Redemmer			3	UPDATE NEW TRIP
Cinque Terre			0	UPDATE NEW TRIP
Eiffel Tower			2	UPDATE NEW TRIP
Forbidden City			1	UPDATE NEW TRIP
Glenfinnan Viaduct			0	UPDATE NEW TRIP
London Towers			0	UPDATE NEW TRIP



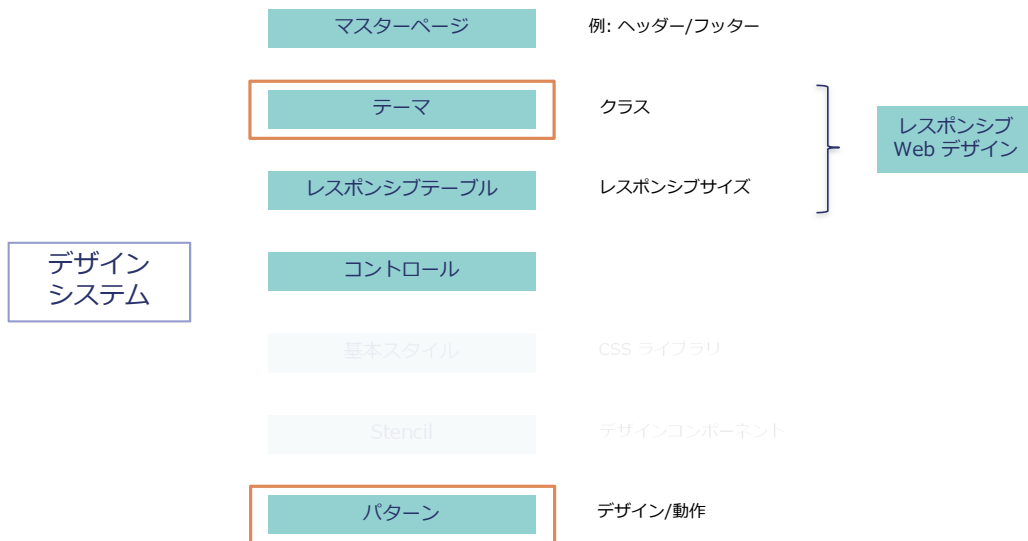
列のクラス: WWOptionalColumn

小 (small) サイズの Web パネルのグリッドに、観光名所名とアクション以外の列を表示しないようにするには、前述のように列のクラスを変更します。

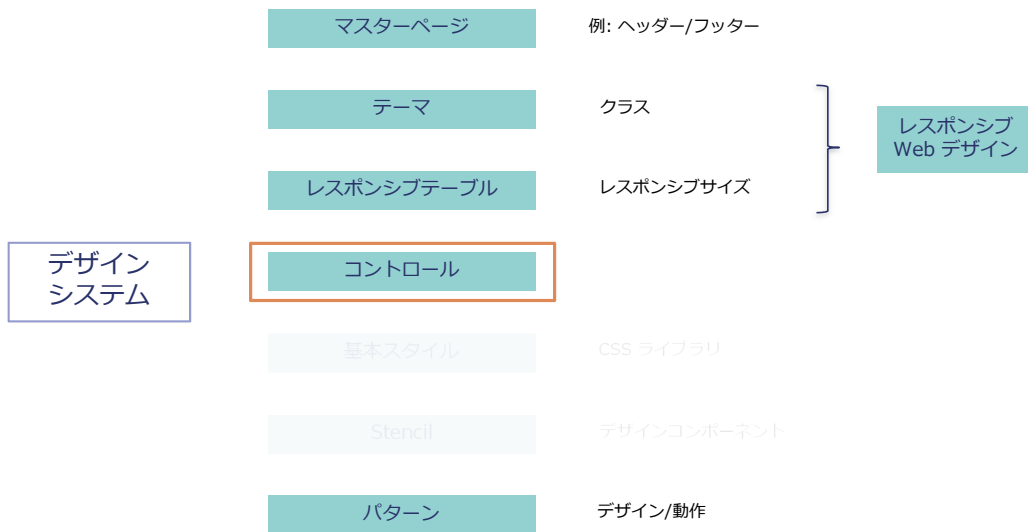




要約すると、デザインシステムのもう 1 つの重要な部分はレスポンシブネスに関係します。レスポンシブネスを実現するには、レスポンシブテーブルのコントロールの位置と可視性を変更するとともに、(テーマの 3 つの列を使用して) サイズに応じてクラスプロパティを変更します。



これらすべてを実現するために行う必要のある作業はほとんどありませんでした。Work With パターンとテーマによって自動的に処理されたことを複製するだけでした。このように、GeneXus には、Web パネルに使用できる基本的な事前定義済みデザインシステムが用意されています。



別の要素としてコントロールがあります。

## グリッド: デザインと動作

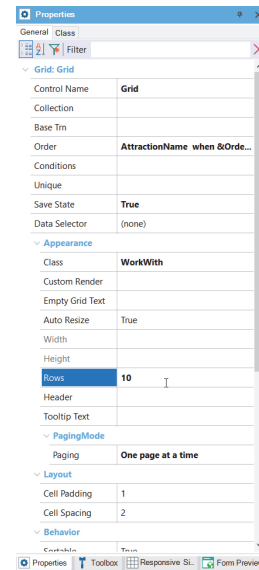
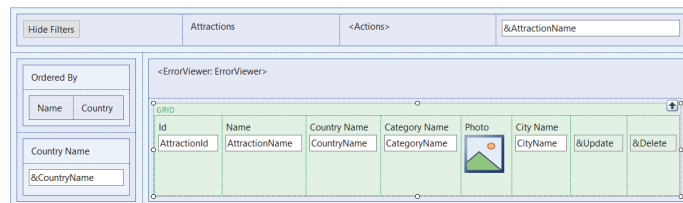
The image shows two side-by-side screenshots of a web application interface. The left screenshot, labeled 'Web パネル: グリッドのページングなし', shows a list of attractions with a 'Total Trips' count of 10 at the bottom. The right screenshot, labeled 'Work With Web パネル: グリッドのページングあり', shows a 'Travel Agency' interface with a table of attractions. The table has columns for Id, Name, Country, Category, Photo, and City Name. It lists three attractions: London Towers, Cinque Terre, and Long Bridges. Each row has 'UPDATE' and 'DELETE' buttons. A pagination bar at the bottom of the table shows navigation controls. An arrow points to the pagination bar in the right screenshot.

Web パネル: グリッドのページングなし

Work With Web パネル: グリッドのページングあり

たとえば、Work With Web パネルを複製した Web パネルには、Work With に存在するページングがありません。

## グリッドのページング



グリッドのプロパティを確認すると、[Rows] プロパティがあり、値が 10 になっています。また、[PagingMode] プロパティの値はこうなっています。作業中のグリッドも同様にしましょう。

## グリッドのページング

Attraction Name From: &AttractionNameFrom

Attraction Name To: &AttractionNameTo

Attraction Id	Attraction Name	Attraction Photo	Trips	&update2	&newTrip
---------------	-----------------	------------------	-------	----------	----------

Total Trips: &totalTrips

Properties - Grid1

General Class

Control Name: Grid1

Collection:

Base Trm: Attraction

Order: CountryId, AttractionName w...

Conditions: CountryId = &CountryId whe...

Unique:

Save State: False

Data Selector: (none)

Appearance

Class: WorkWith

Custom Render:

Empty Grid Text:

Auto Resize: True

Width:

Height:

Rows: 0

Header:

Tooltip Text:

Layout

Cell Padding: 1

Cell Spacing: 2

Behavior

Sortable: True

Allow Drop: False

Properties - Grid1

General Class

Control Name: Grid1

Collection:

Base Trm: Attraction

Order: CountryId, AttractionName w...

Conditions: CountryId = &CountryId whe...

Unique:

Save State: False

Data Selector: (none)

Appearance

Class: WorkWith

Custom Render:

Empty Grid Text:

Auto Resize: True

Width:

Height:

Rows: 10

Header:

Tooltip Text:

PagingMode

Paging: One page at a time

Layout

Cell Padding: 1

Cell Spacing: 2

Behavior

値 0 は、すべての明細行をロードすることを意味します。[Paging] プロパティが Work With と同じ値の場合、[Rows] の値を 10 に変更することで、ページングを行い、一度に 10 件のレコードをデータベースから取得するよう指示することになります。

## グリッドのページング

Travel Agency




RecentWWAttractions From...

Country Id

(None) ▾

Attraction Name From

Attraction Name To

Attraction Name	Attraction Photo	Trips		
<a href="#">Rifugio Nuvolau</a>		0	UPDATE	NEW TRIP
<a href="#">Smithsonian Institute</a>		1	UPDATE	NEW TRIP
<a href="#">The Great Wall</a>		0	UPDATE	NEW TRIP

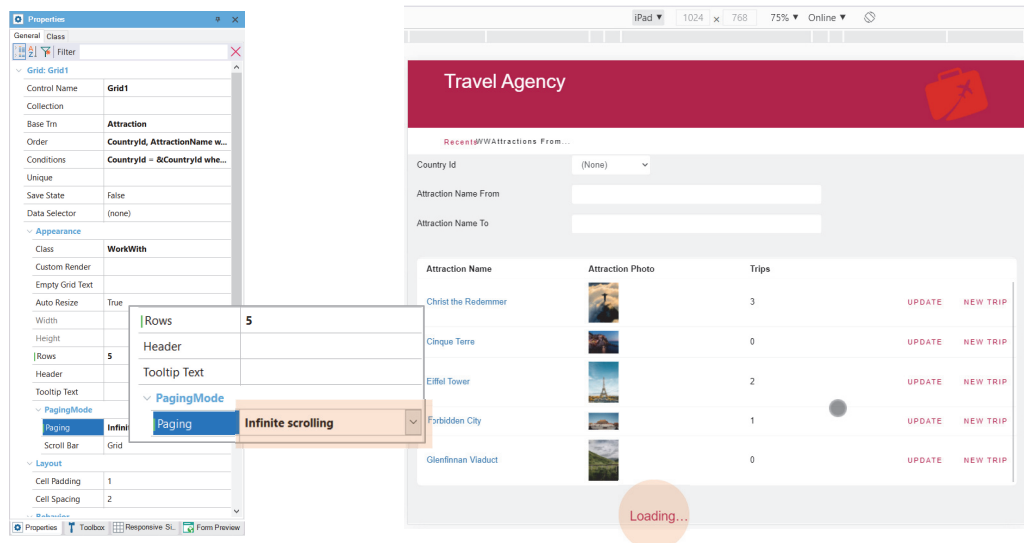
« < > »

Total Trips1

このオブジェクトを生成し、実行してみます。

すべての観光名所がロードされていた画面を再表示すると、ページングオプションが有効になります。

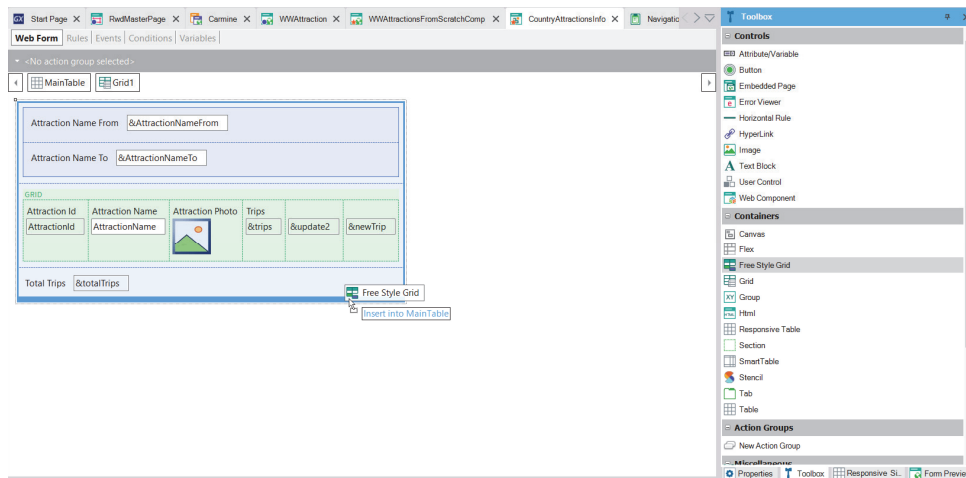
## グリッドのページング: 無限スクロール



次に、ページサイズを 5 行に変更し、[Paging] プロパティの値を「Infinite Scrolling」に変更して試してみます。見やすいように表示をタブレットに変更します。グリッドに 5 行がロードされていますが、次のページを表示するためのボタンがありません。ただし、スクロールすると、「Loading」というメッセージが表示されます。次の 5 行が検索され、画面にロードされます。残りの行も同様です。



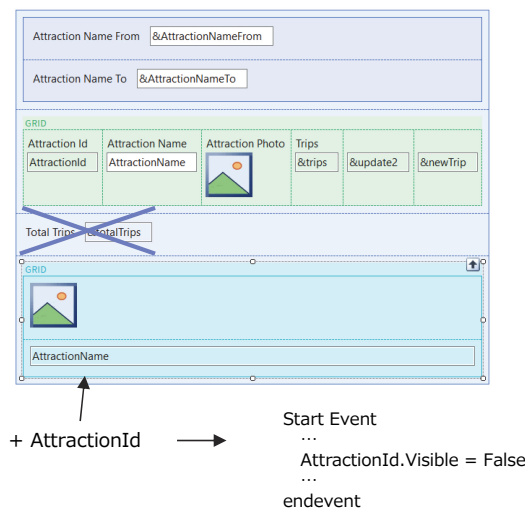
## フリー スタイル グリッド



これまでは、情報を表のような構成で表示していました。観光名所をもっと柔軟な形で表示することはできないでしょうか。

たとえば、写真と名前だけを上下に表示したいとします。このような場合のために、フリー スタイル グリッドが用意されています。これをフォームの下部にドラッグし、最初に項目属性 AttractionName と AttractionPhoto を配置して、写真が上になるように移動します。ラベルは削除します。

## フリー スタイル グリッド

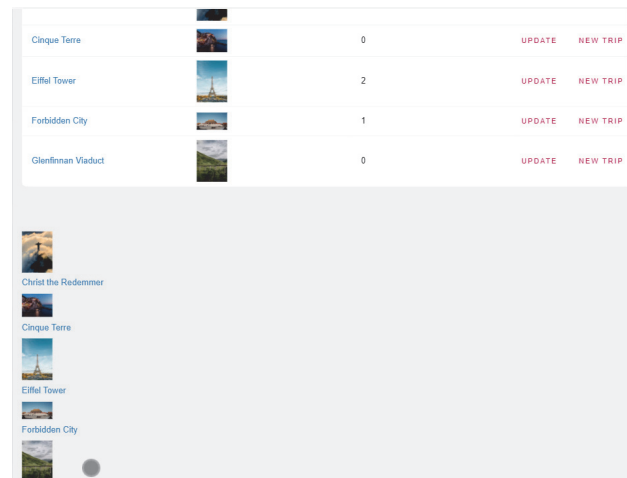


このグリッドには、標準のグリッドと同じように、ベーストランザクション、ソート順、抽出条件を指定するプロパティがあります。もう 1 つのグリッドからコピーします。

不要な変数は削除します。

AttractionId を追加していませんでした。この項目属性は、もう 1 つのグリッドと同様に、非表示にするにしてもロードする必要があるものです。これは後でパラメーターとして送信するためです。

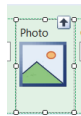
## フリー スタイル グリッド



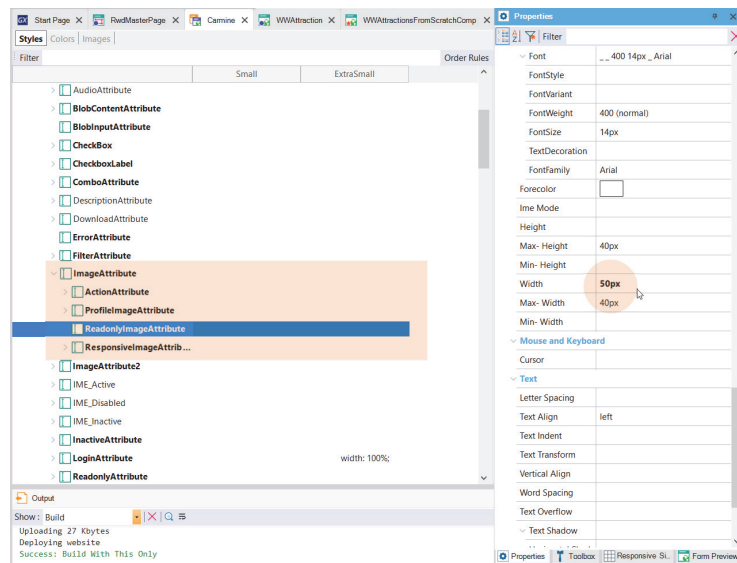
それでは試してみます。画面を再表示します。

標準グリッドの下に新しいグリッドが表示されます。デザインはまだ適用されていません。画像はもっと大きいほうがいいでしょう。なぜこのように小さいのでしょうか。

## 画像の拡大



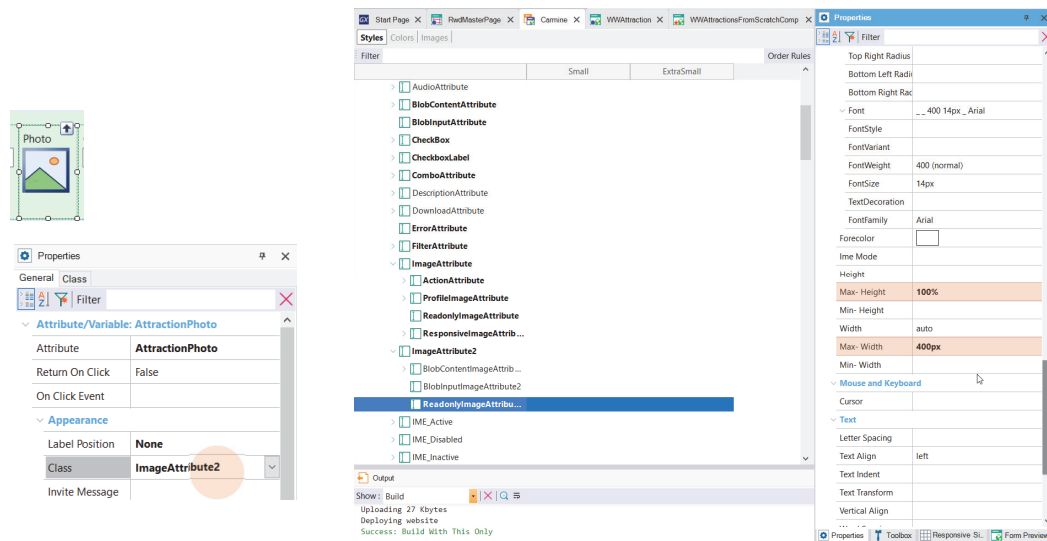
Properties	
General	Class
Attribute/Variable: AttractionPhoto	
Attribute	AttractionPhoto
Title	Photo
Class	ImageAttribute
Column Class	WWColumn WWOOptionalColu...
Return On Click	False



標準グリッドとフリー スタイル グリッドの項目属性コントロールのクラスを見ると、既定で同じクラス `ImageAttribute` が設定されていることが分かります。これは項目属性が `Image` データタイプであるからです。テーマで確認すると、一連のサブクラスがあり、その中に `ReadonlyImageAttribute` があります。これが、実行時に写真に適用されるクラスです。この場合の項目属性コントロールは読み取り専用であるからです。このように、プロパティでは親クラスが設定されていても、実際にはこの子クラスが適用されます。そのプロパティを確認すると、画像の幅に 50 ピクセルが指定されています。画像が小さかった理由が分かりました。

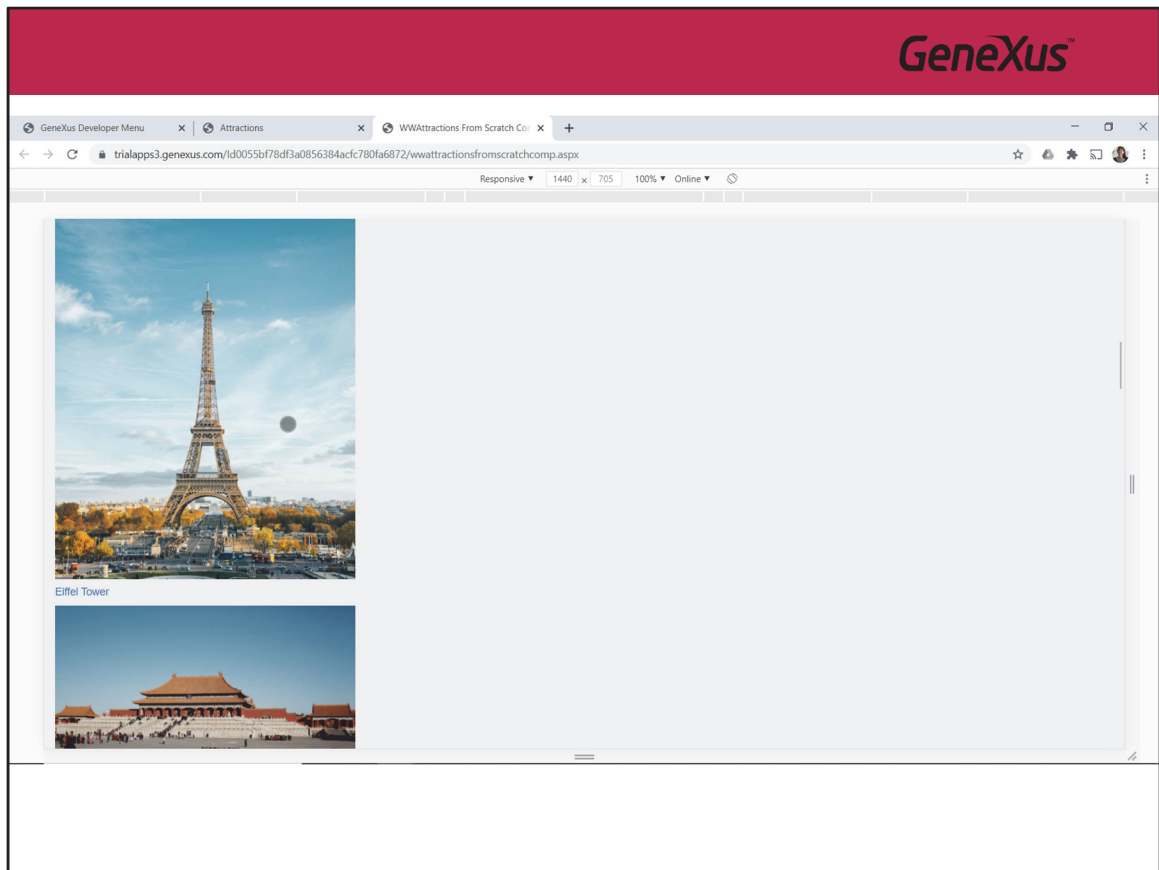
ここで、このプロパティを変更し、たとえば幅を 400 ピクセルにした場合、表示は期待どおりになります。しかし、その影響で、同じクラスのほかのコントロールも変更されます。具体的には `Work With` 内の画像です。

## 画像の拡大

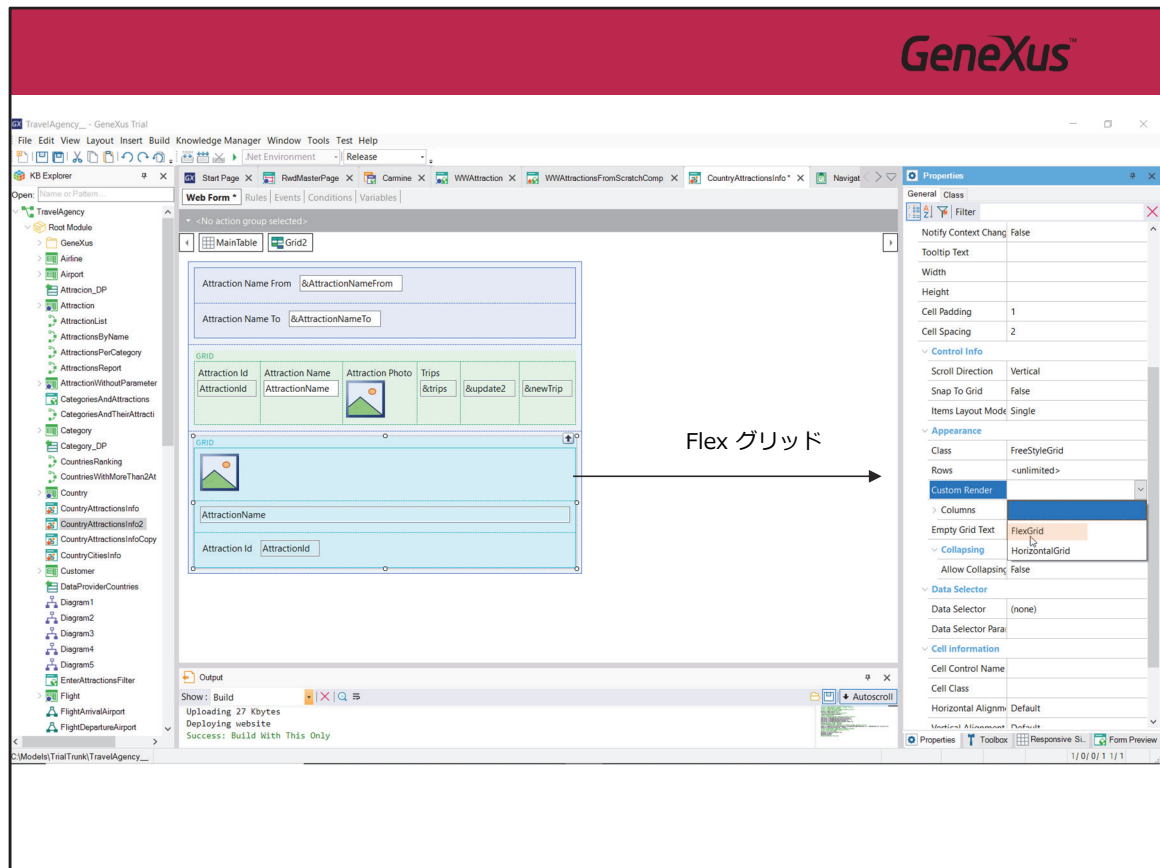


それは望んでいることではないため、同じレベルに別のクラス `ImageAttribute2` を作成しました。これは一連のサブクラスとともに自動的に作成されます。ここで `ReadOnly` のプロパティを変更します。幅が 400 ピクセルを超えない限りは画像が元のサイズで表示されるように設定しています。超える場合は、このサイズまで縮小します。

2 つ目のグリッドの `AttractionPhoto` コントロールのクラスを新しいほうに変更します。

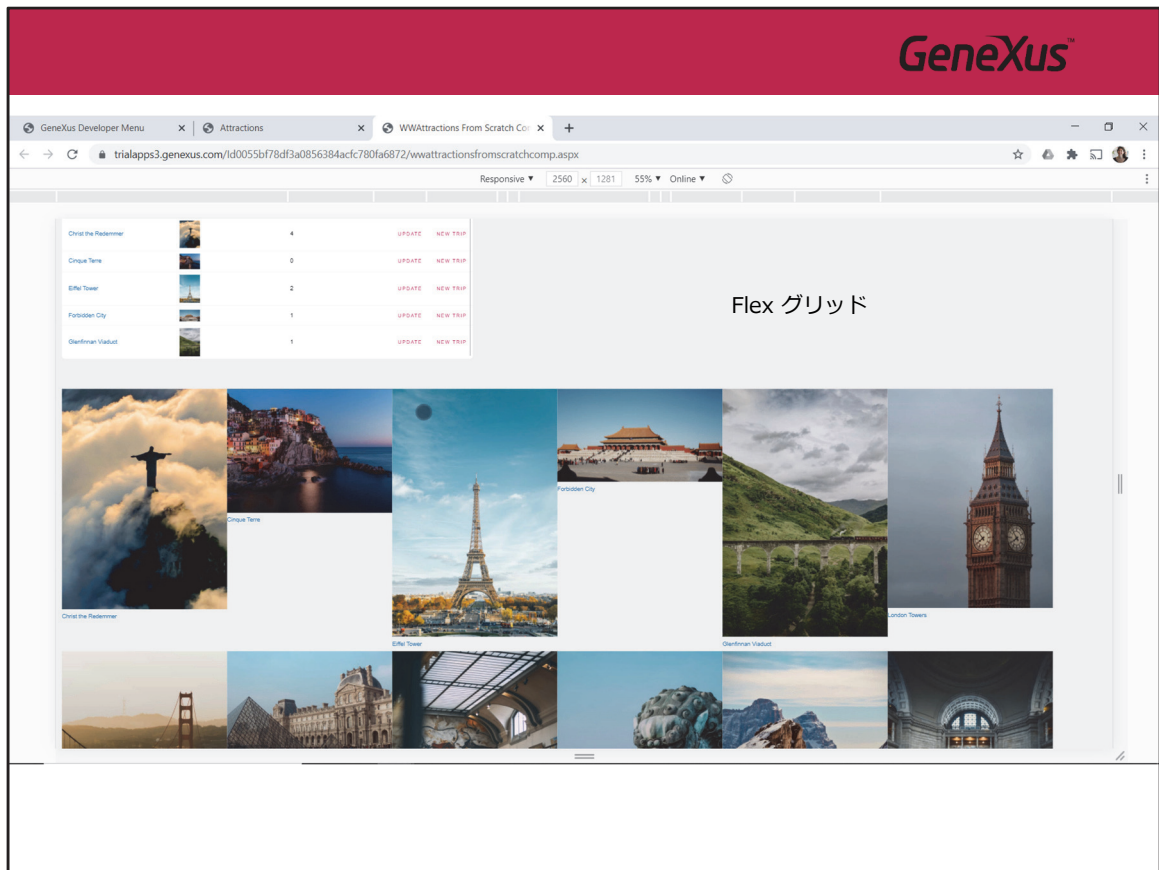


実行してみましょう。違いがはっきりと分かります。



ここで、画面のスペース全体を使って観光名所を表示するとします。  
写真はそれぞれサイズが異なることが分かっています。

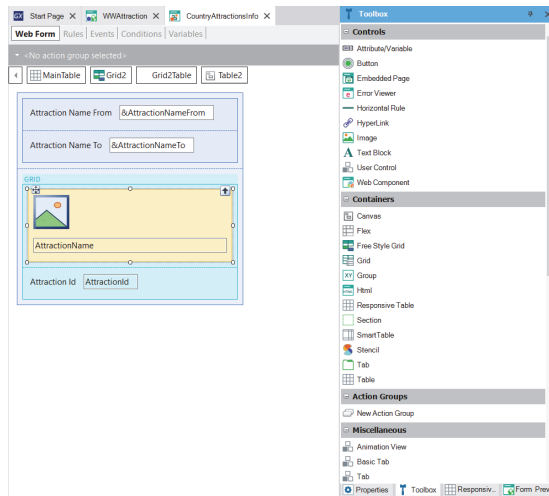
グリッドのレンダリングを変更し ([Custom Render] プロパティ = FlexGrid)、  
コンテンツがパズルのように画面上の空きスペースを埋めるようにします。試して  
みましょう。



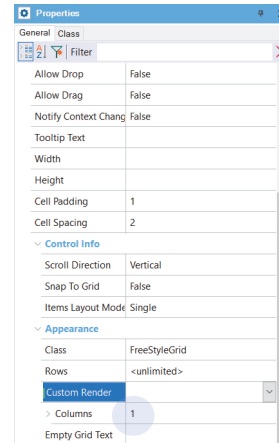
画面サイズに応じて、また最大幅 400 ピクセルを考慮に入れて、観光名所がスペース全体を使って横方向に配置されました。



## 重ねて表示するためのキャンバス



### 既定のフリー スタイル グリッド

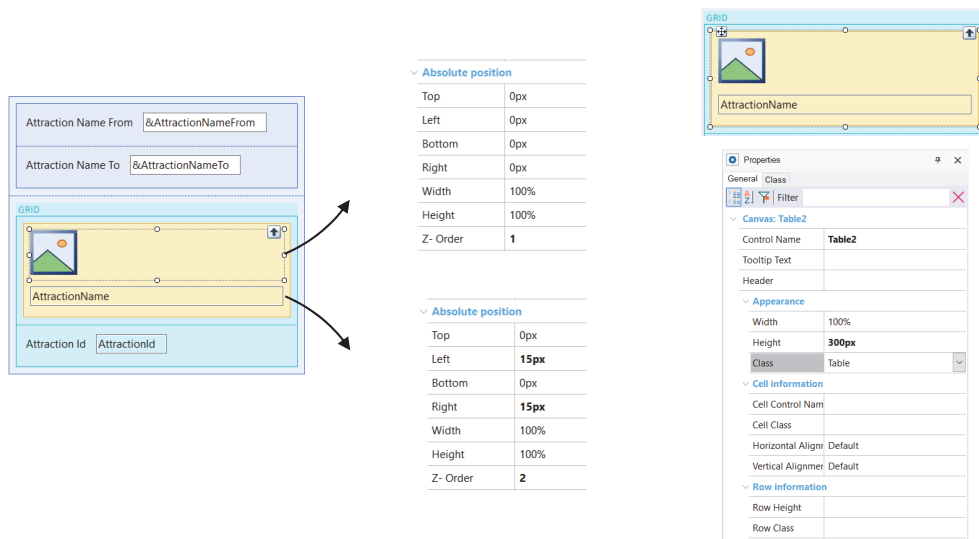


次に、観光名所の名前を画像に重ねて表示する方法を考えます。

最初に標準グリッドを削除します。ここには示していませんが、関連付けられたイベントも削除しています。

次にキャンバスコントロールをフォームに挿入します。これは、コントロールを重ねて表示できるタイプのテーブルです。ここに、重ねて表示する2つのコントロールを挿入します。フリー スタイル グリッドのレンダリング ([Custom Render] プロパティ) は既定値に戻し、観光名所 (写真と名前) が1行に1つ表示されるようにします。

## 重ねて表示するためのキャンバス



The diagram illustrates the configuration of a canvas control in GeneXus. It shows a visual representation of the canvas with a grid, a table of absolute positions for the canvas and its content, and a screenshot of the Properties window.

**Canvas Absolute Position**

Absolute position	
Top	0px
Left	0px
Bottom	0px
Right	0px
Width	100%
Height	100%
Z-Order	1

**Content Absolute Position**

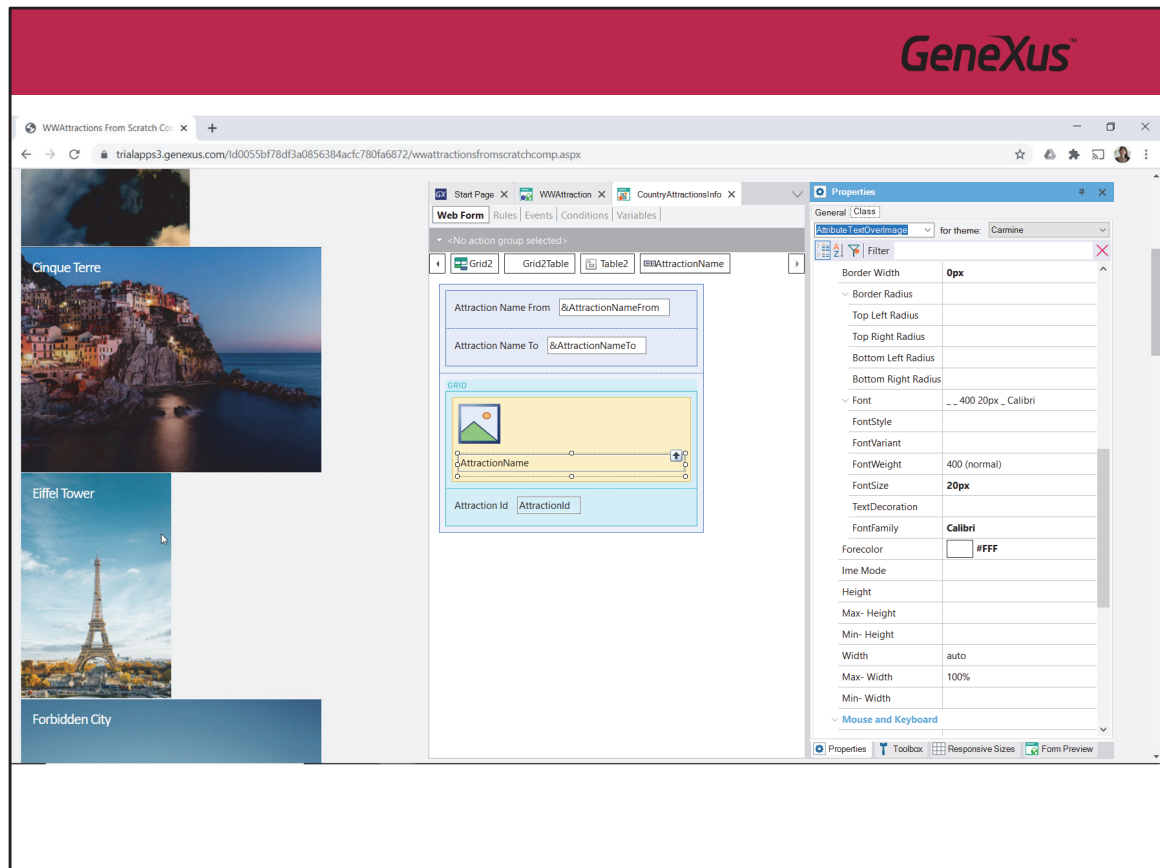
Absolute position	
Top	0px
Left	15px
Bottom	0px
Right	15px
Width	100%
Height	100%
Z-Order	2

**Properties Window**

Properties window showing the configuration for the Canvas control (Table2).

- Control Name: Table2
- Tooltip Text:
- Header:
- Appearance:
  - Width: 100%
  - Height: 300px
  - Class: Table
- Cell information:
  - Cell Control Nam:
  - Cell Class:
  - Horizontal Alignm: Default
  - Vertical Alignmer: Default
- Row information:
  - Row Height:
  - Row Class:

このコントロールをキャンバス内に配置すると、これらのプロパティが表示され、コントロールのテーブルに対する相対的な位置、占めるスペース、属するレイヤーを設定できます。レイヤー以外は既定の値のままにします。「1」と設定するのは、画像を最背面のレイヤー、つまり名前の後ろに配置するためです。観光名所名はレイヤー 2、つまりレイヤー 1 の前に配置します。また、このコントロールは左余白を 15 ピクセルにし、写真の縁との間に余白をつくります。キャンバステーブルのプロパティには、テーブルが含まれるコントロールに対する相対的な高さと幅を指定するプロパティがあります。ここでは、幅は 100% を占める設定にし、高さは 300 ピクセルにします。

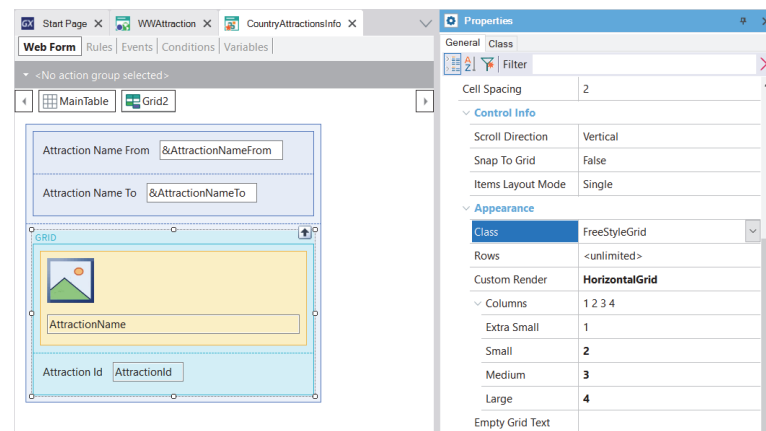


最後に、AttractionName コントロールのデザインを変更し、見た目を整えます。

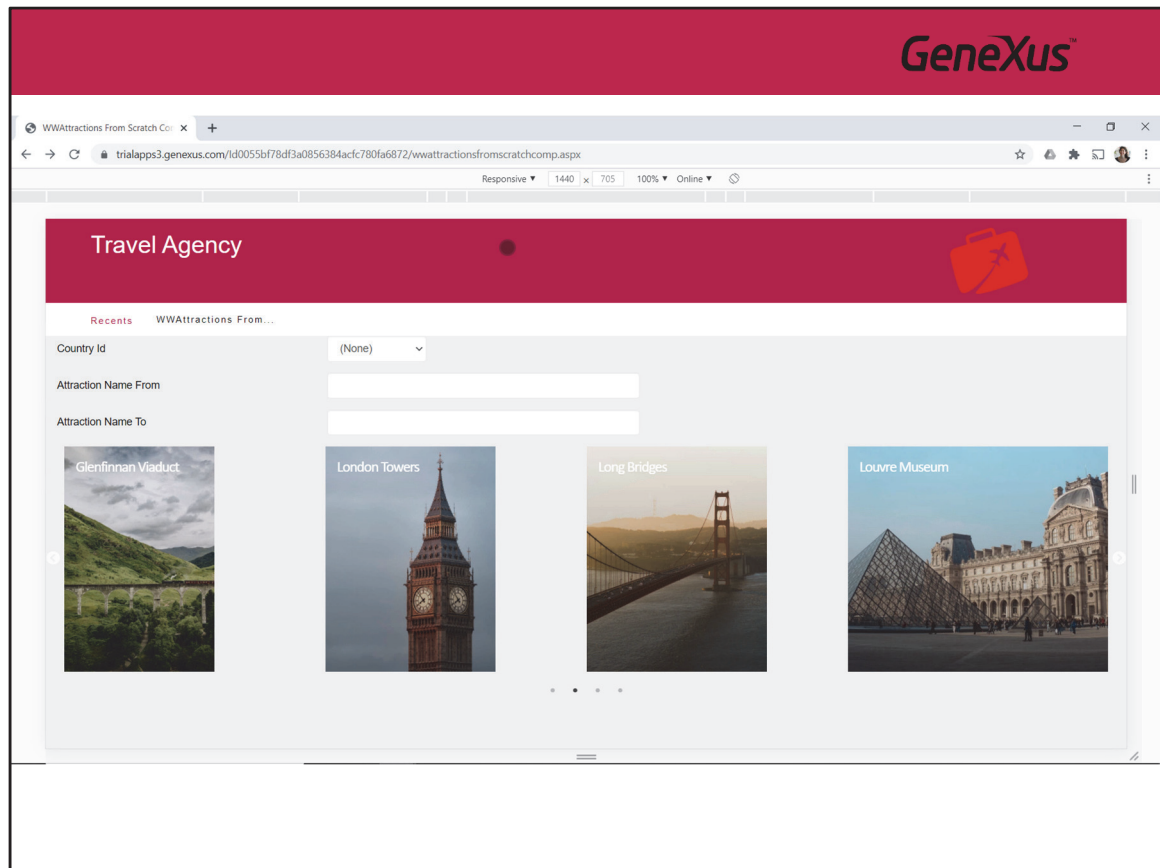
既定では Attribute クラスが関連付けられていますが、テーマに作成したこの名前のもので変更します。プロパティで、フォントの色を白、フォントタイプを Calibri、サイズを 20 ピクセルにします。

実行してみましょう。エッフェル塔の名前をクリックすると、詳細情報が表示されます。

## 水平グリッド



グリッドコントロールのさまざまな可能性を見てきましたが、最後にレンダリングを変更して ([Custom Render] プロパティ = HorizontalGrid)、水平グリッドとして表示します。たとえば、スマートフォンのサイズでは 1 列のみ、小サイズでは 2 列、中サイズでは 3 列、大サイズでは 4 列にします。



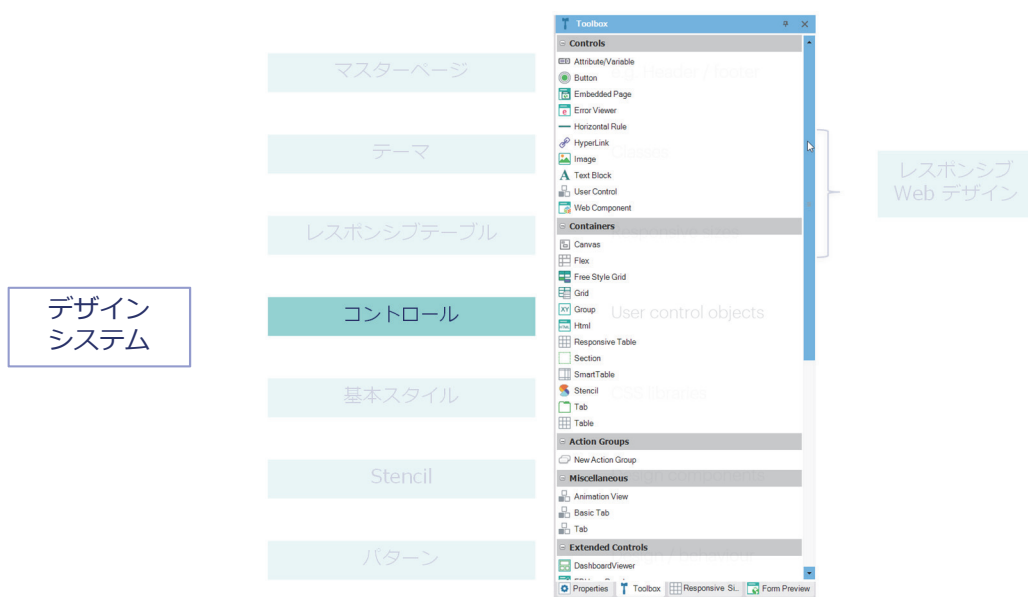
F12 キーを押します。大サイズでは 1 ページに 4 つの観光名所が表示されます。  
中サイズの場合は 3 つ、  
タブレットの場合は 2 つ、  
スマートフォンの場合は 1 つになります。

この場合、同じサイズの画像を挿入する必要があります。

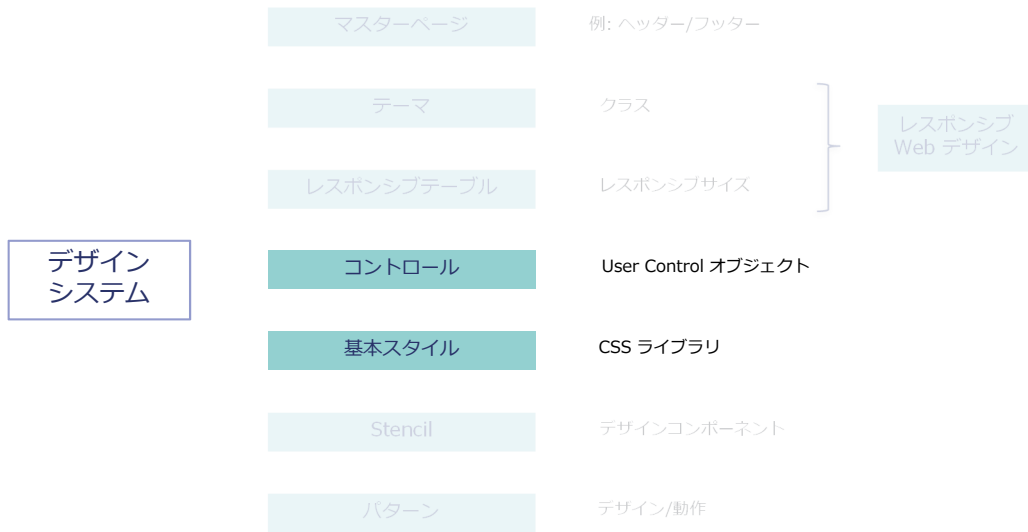
テーマのコントロールやクラスでできることの概要を紹介しました。



ここまでに取り上げた機能以外にも、GeneXus の強力なデザインシステムを活用するうえで重要な役割を果たすものがあります。それらについては、このレベルでは紹介のみに留めておきます。

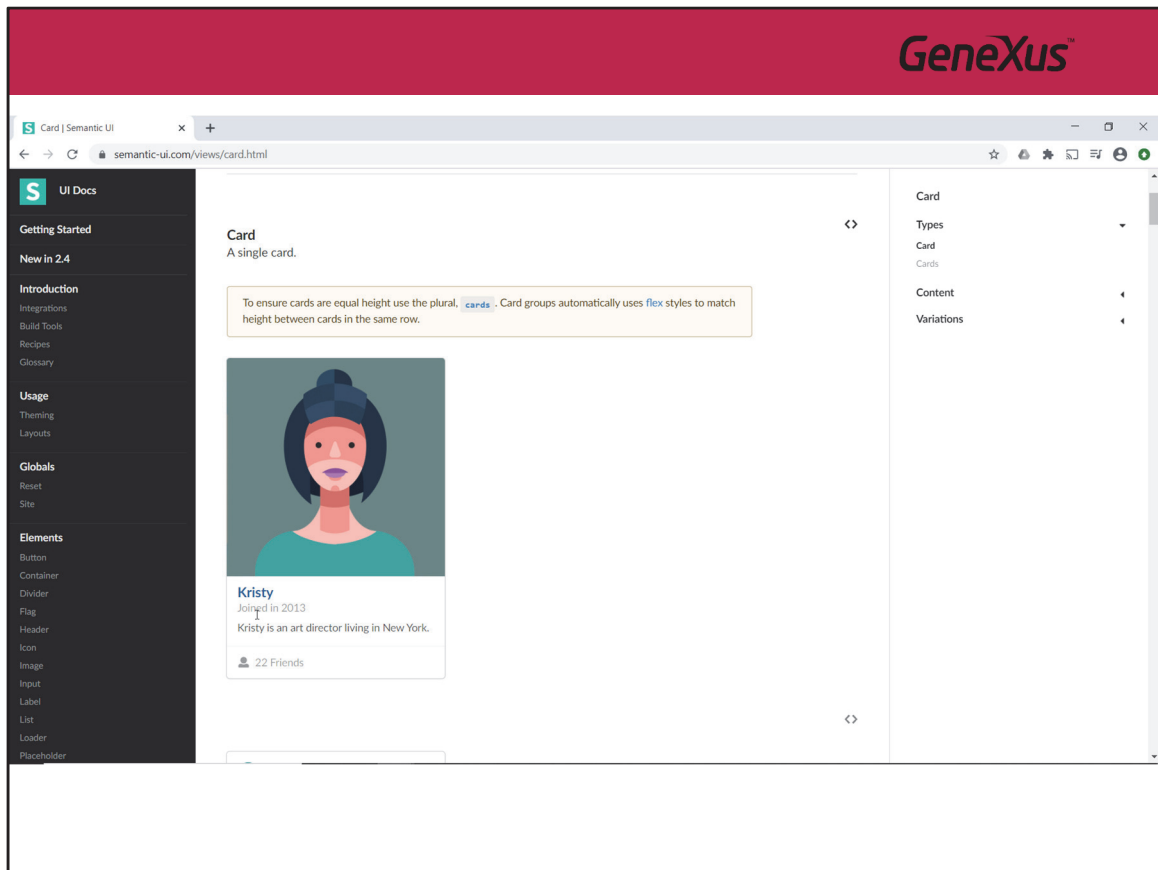


フォームは、GeneXus のツールボックスに用意されているコントロールを所定の方法で使用できるだけではありません。

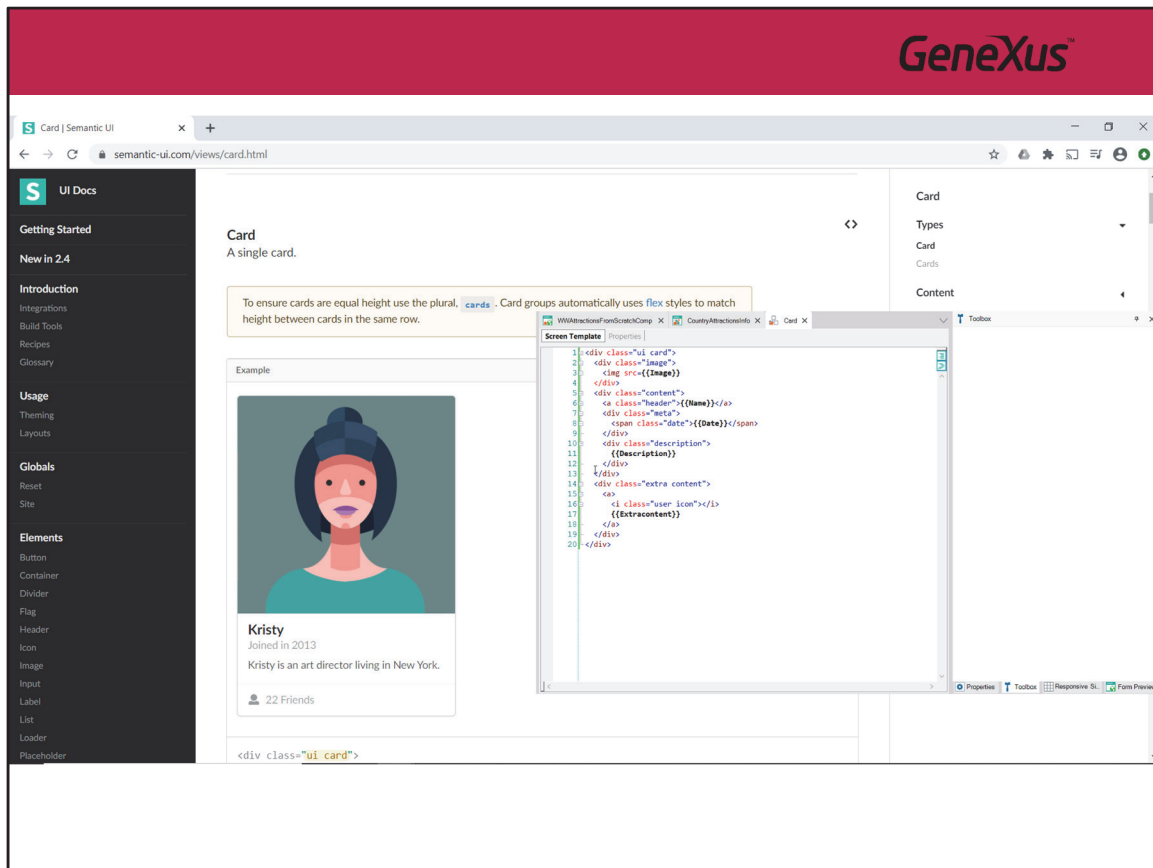


そのほかに、ユーザーコントロールを独自に定義することもできます。ユーザーコントロールは、各種コントロールを CSS ライブラリとともに提供するプラットフォームからコピーして使用できます (CSS ライブラリはテーマに似ていますが、この場合、コントロールのスタイルを指定するために使用します)。

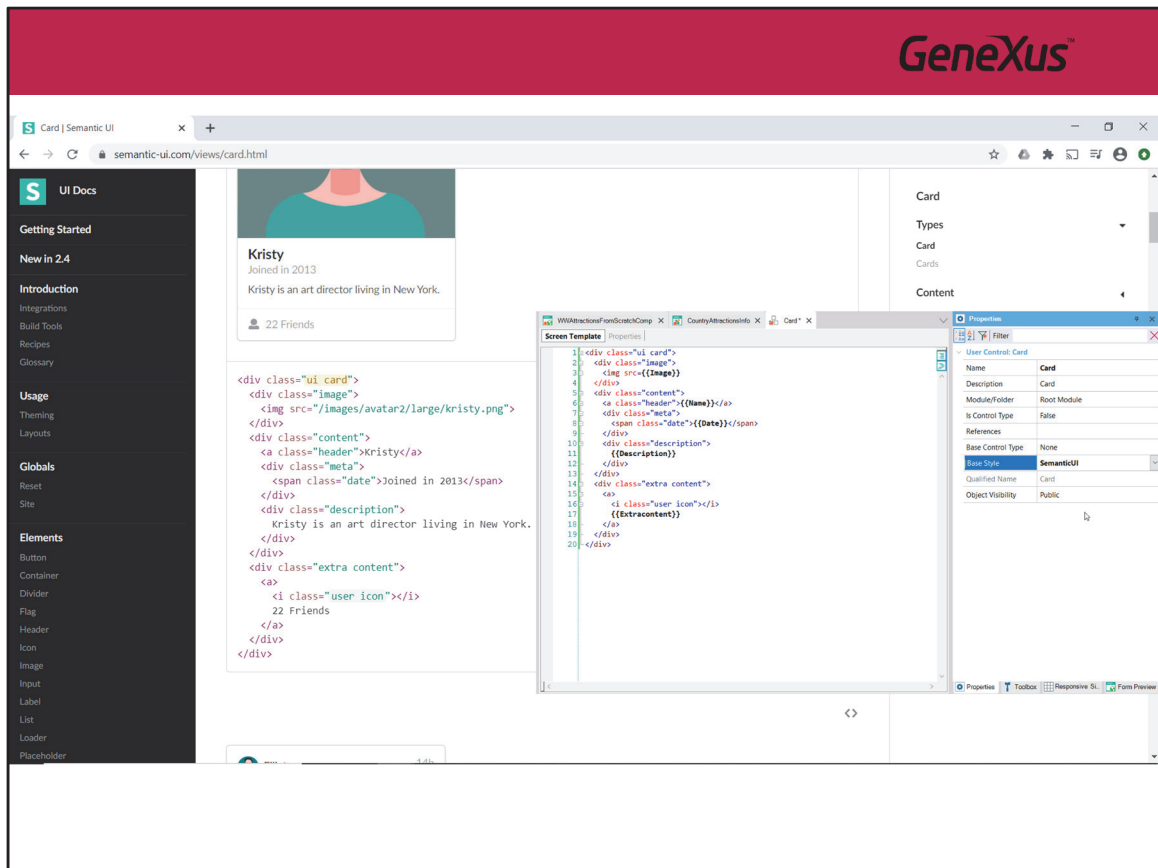




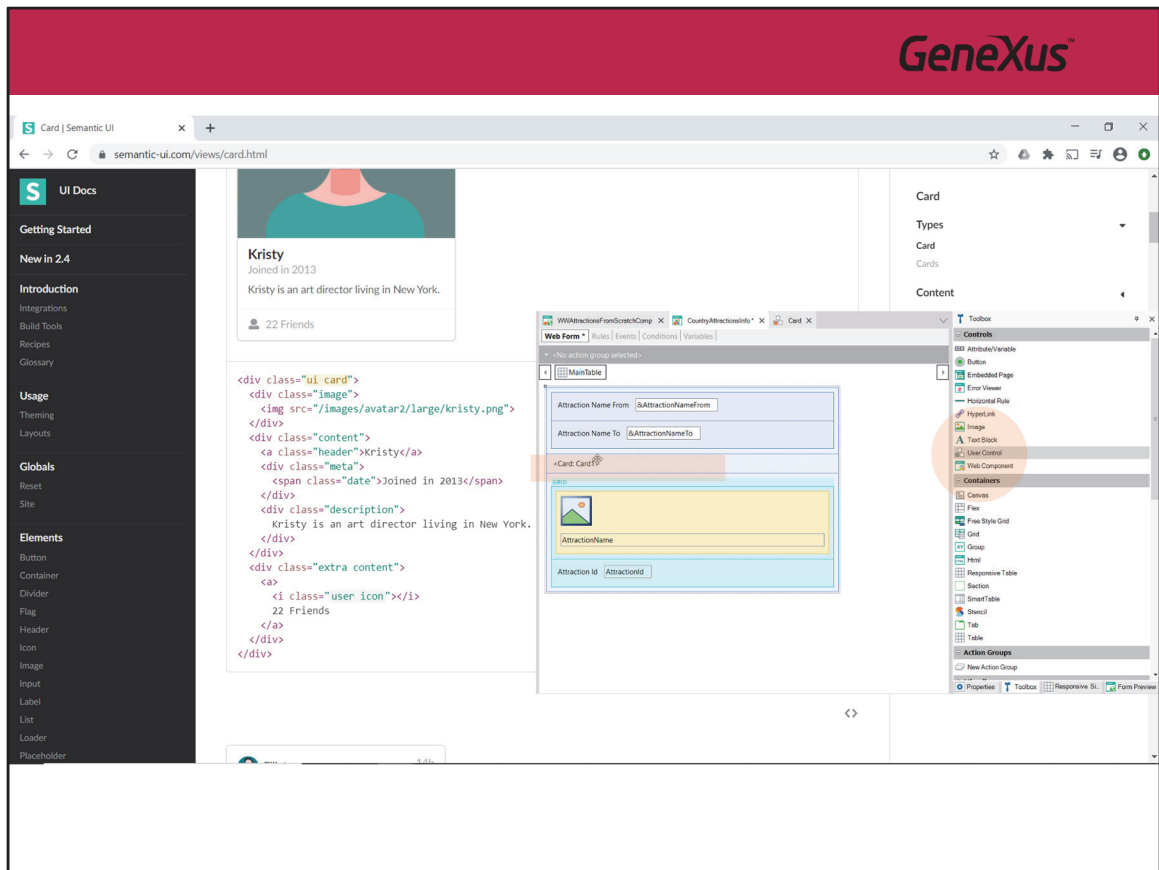
例として、Semantic UI があります。このようなカードコントロールの使用を考えているとします。



そのためには、(同名の) User Control オブジェクトを作成し、HTML コードをコピー アンド ペーストし、定数のデータ部分を SDT のメンバー名のようなものに変更する必要があります (これにより、コントロールを動的にする、つまり該当するコントロールがデータベースなどのデータ (指定可能) とともに動的にロードされるようにします)。

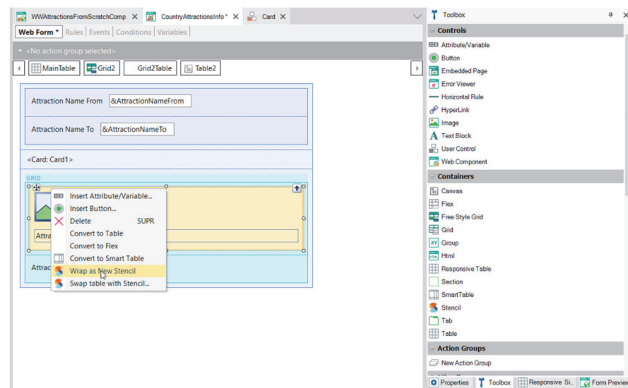


また、CSS、つまりクラスデザインの取得元を指定し、その他の処理を行います。



これにより、ユーザーコントロールをツールボックスから利用できるようになります。

デザイン  
システム



レスポンス  
Web デザイン

Stencil

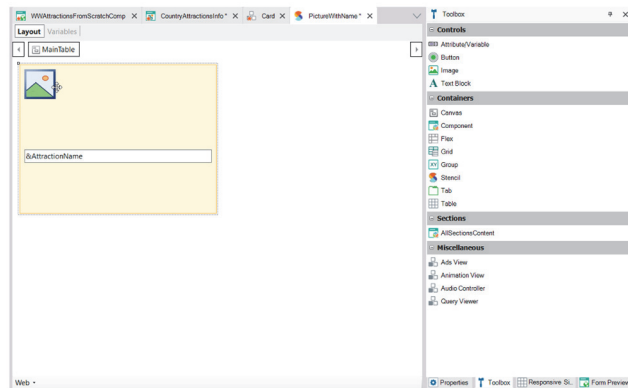
デザインコンポーネント

パターン

デザイン/動作

さらに、GeneXus には、Stencil という機能も用意されています。この機能を使用すると、画面の同じ部分 (一連のコントロール) のデザインを複数の画面で繰り返し使用できます。

デザイン  
システム



レスポンス  
Web デザイン

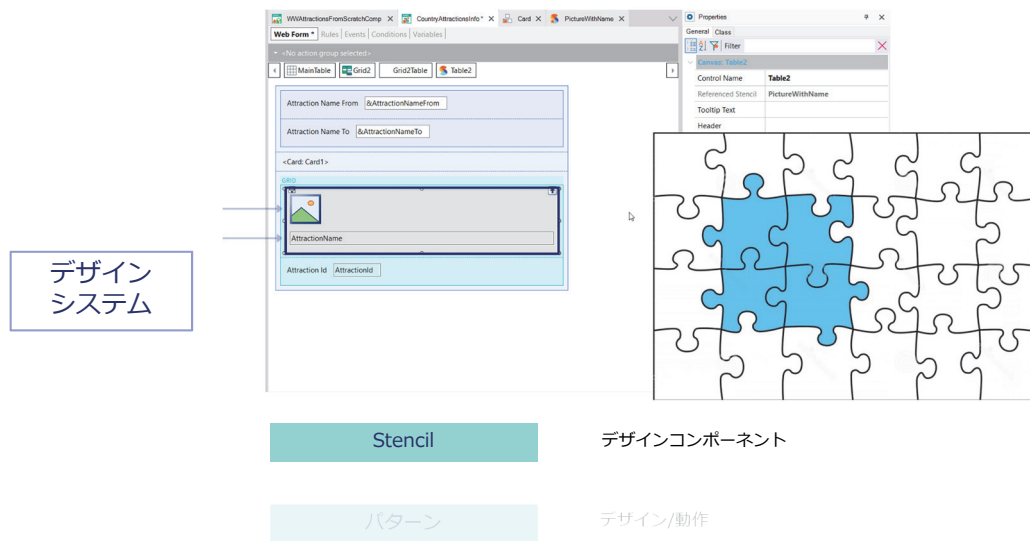
Stencil

デザインコンポーネント

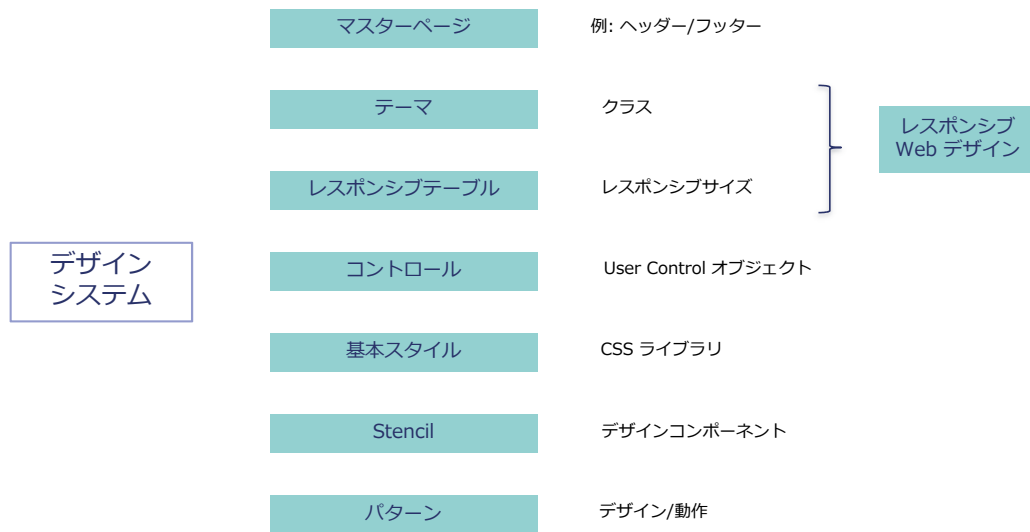
パターン

デザイン/動作

これにより、デザインを高度に抽象化することができます。



もしも開発者が、Web アプリケーションの画面のことを、小さなピース (コントロール) の集まりとして成り立っているパズルのように考える場合、Stencil は、一連のコントロールをグループ化し、そのデザインを多くの画面で繰り返し使用できるようにするための手段です。つまり、複数の小さな部分からなる 1 つのパーツを定義します。



ここでは、GeneXus におけるデザインシステムの実装に関わる重要な要素について説明しました。