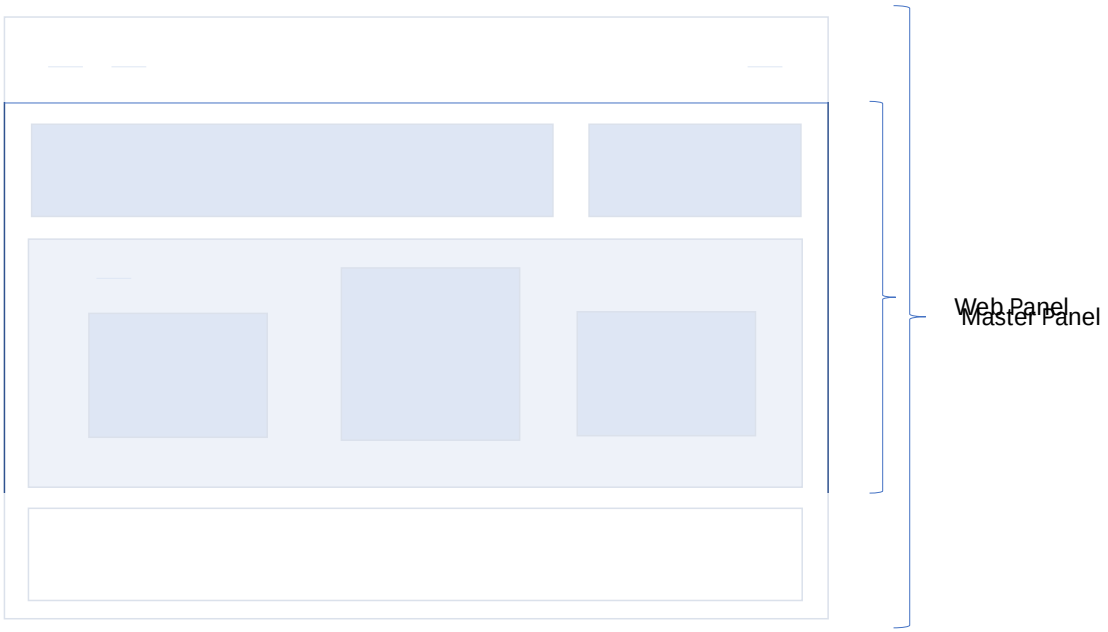
Eventos globales

Interacción entre componentes de una misma pantalla

GeneXus

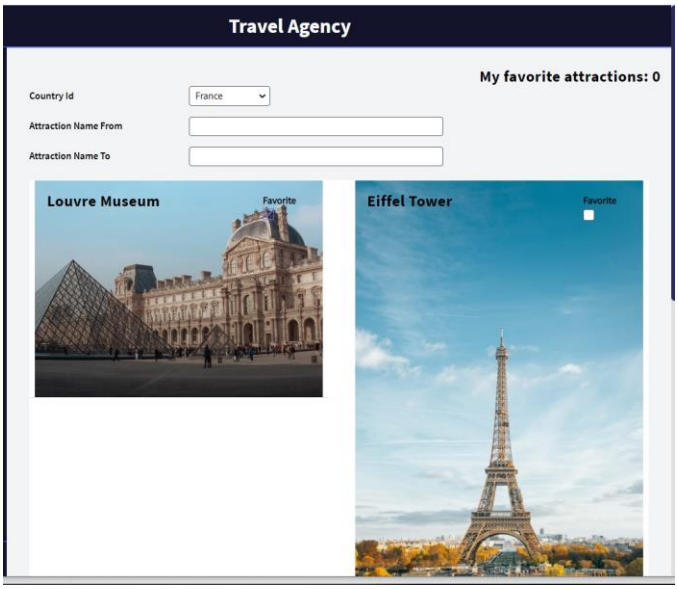
Screen composed by many screens



Como ya sabemos, el uso de componentes nos permite reutilizar funcionalidades ya desarrolladas.

Por ejemplo, es el caso de un web panel como el que ya hemos visto, donde se listaban las atracciones turísticas que una agencia de viajes ofrecía visitar. No nos ocuparemos aquí del diseño, que sin duda puede ser mejorado.

My favorite attractions



Estamos viendo una pantalla no demasiado compleja, que presenta una parte fija que viene de la master page, y otra parte que es propia. Podemos filtrar por país, y también por nombre de atracción.

Pro lo que nos importa en este momento es observar que hemos agregado la posibilidad de marcar atracciones turísticas como favoritas. Por ejemplo, marco esta y esta otra, o sea, hemos marcado dos atracciones como favoritas.
Sin embargo, si observamos aquí arriba, está indicando un cero, como si no hubiéramos marcado ninguna como favorita.

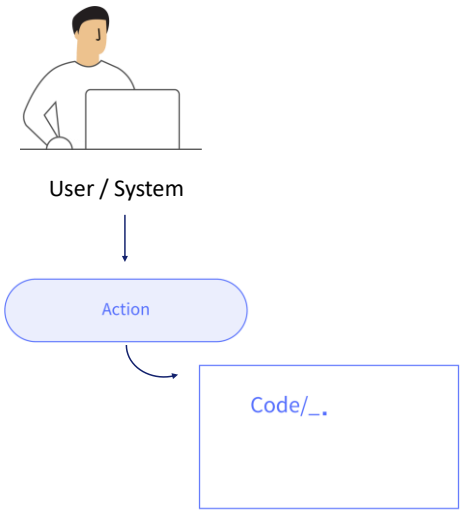
Refresquemos la pantalla... y ahora si vemos el 2.
Pretendemos también que, si desmarcamos una atracción favorita, este número se actualice indicando 1

Entonces, ¿cómo conseguimos que este valor se actualice automáticamente según se marque o se desmarque una atracción turística como favorita?

Global Events

Bien. Vamos a introducir entonces el concepto de Evento global.

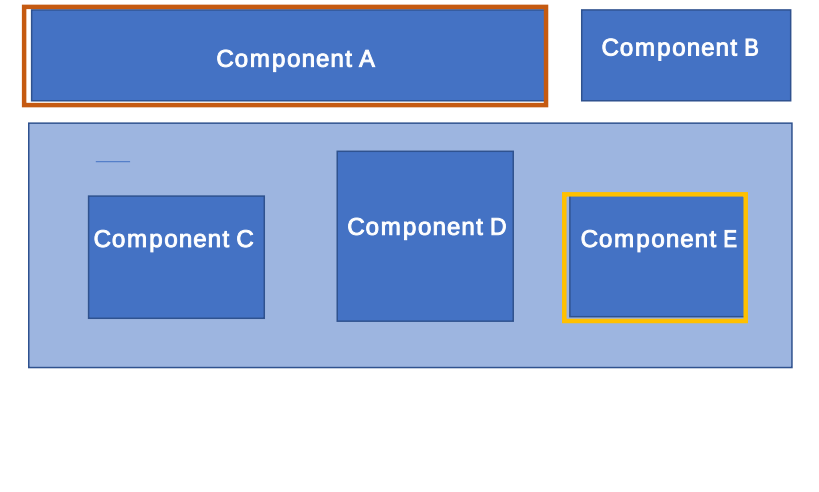
Events



Local event

Generalmente, en objetos interactivos como es el caso de los objetos web panels, los eventos son acciones manejadas por el programa y desencadenadas por el usuario. Los eventos suelen ser locales para el programa en el que se definen.

Events



En cambio, los eventos globales permiten definir eventos que aplican a todos los componentes de una aplicación.

Mientras que los eventos locales se desencadenan en respuesta a una acción del usuario, los eventos globales son código que permanece inactivo hasta que es invocado desde otro evento, desde cualquier otro componente.

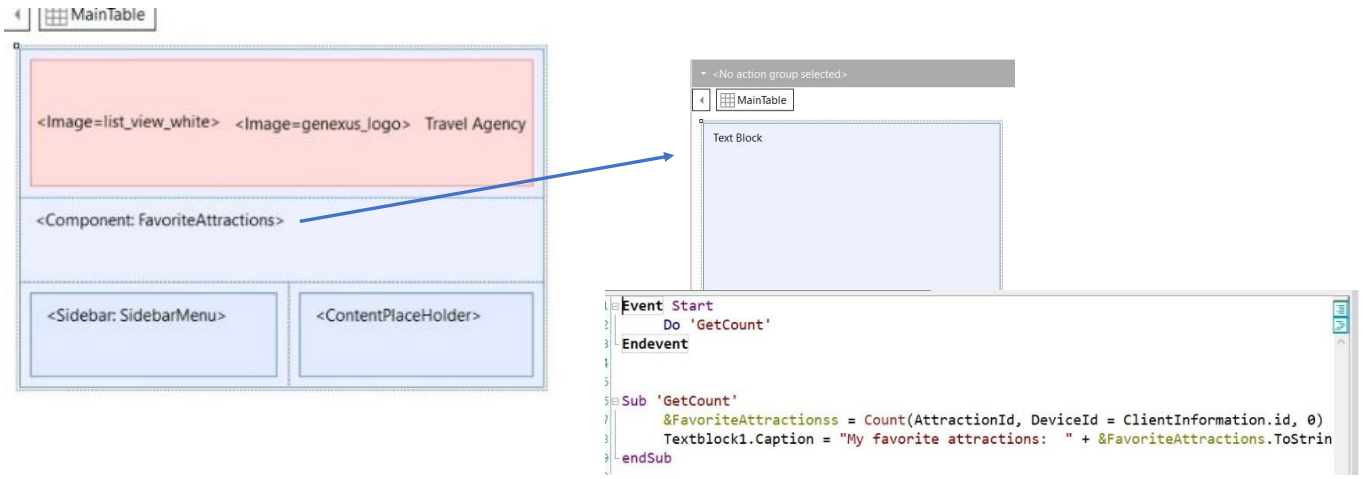
En la imagen que vemos, un evento global definido por ejemplo en el "Componente E" podría invocarse desde el "Componente A". Cualquier combinación es posible, independientemente del nivel de anidamiento de los componentes que forman la pantalla.

My favorite attractions

Transaction: FavoriteAttraction

Name	Type	Description	Formula	Nullable
FavoriteAttraction	FavoriteAttraction	Favorite Attraction		
DeviceId	DeviceId	Device Id		No
AttractionId	Id	Attraction Id		No

Master page: MasterUnanimoSidebar



Vamos a GeneXus a ver este Web panel.
Vemos, primero que nada, que tiene una master page asociada, que es la default. Por lo que, el contenido de su pantalla se cargará dentro del content place holder de dicha master page.
Pero además hemos agregado otro componente para mostrar la cantidad total de atracciones turísticas que el usuario marcó como favoritas.

Para guardar las atracciones que un usuario marca como favorita deberíamos contar con una tabla de usuarios, para poder indicar atracciones favoritas por usuario. Si no queremos aún trabajar con usuarios (por ejemplo, porque aún no le aplicamos GAM y estamos todavía resolviendo si vamos a tener una tabla propia de usuarios o no) entonces podemos provisoriamente mantener favoritos por instancia de browser.
Para ello creamos esta transacción con este atributo, al que le hemos especificado este nuevo dominio que también hemos creado nosotros, con el mismo tipo de datos que esta propiedad...
ClientInfomation es un external object del módulo GeneXus, cuya propiedad Id permite identificar al dispositivo cliente que está ejecutando, tanto sea en forma nativa como web.

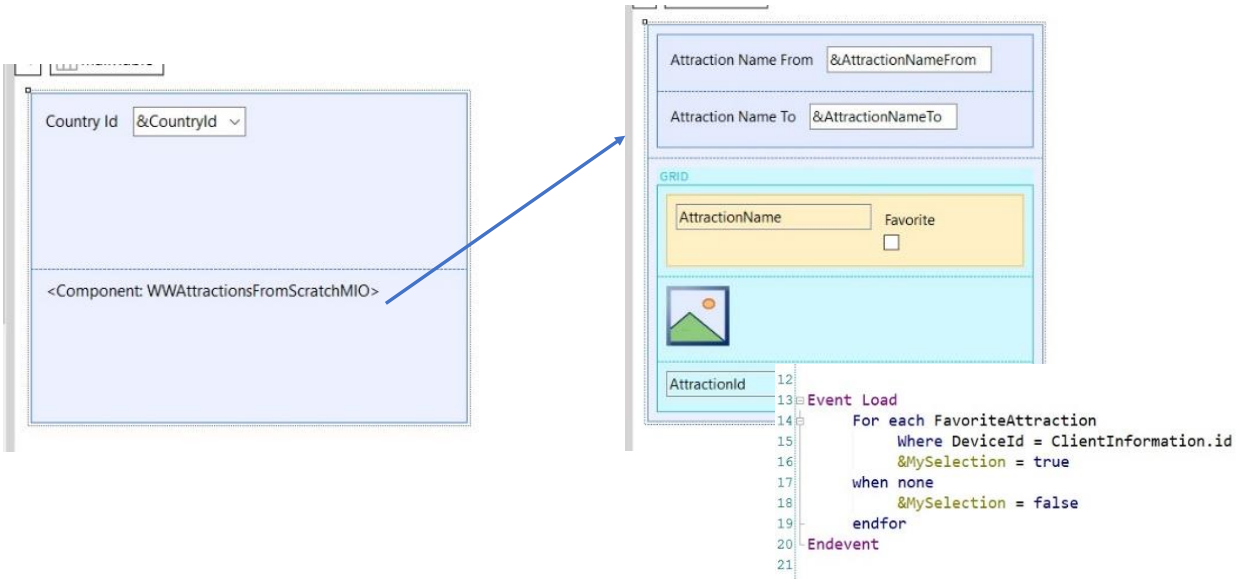
En el caso de aplicaciones Web, la propiedad devuelve un identificador del usuario, que persiste entre todas las sesiones con el mismo navegador y para la misma aplicación. Entonces lo que hemos hecho es agregar una transacción con clave compuesta por el dispositivo y la atracción turística. En la tabla asociada a esta

transacción es que guardaremos los favoritos.

Si observamos el Web component que estamos cargando aquí, vemos que solo contiene un text block, de este nombre, cuyo caption cargamos en ejecución. solo en el evento Start, invocando a esta subrutina que lo primero que hace es calcular la cantidad de atracciones turísticas para este cliente, que se encuentran en la tabla que contiene a estos dos atributos (que es FavoriteAttraction).

Aquí tenemos el valor que estamos buscando. Luego a ese numérico lo transformamos en string, para asignárselo al textblock que se mostrará.

My favorite attractions

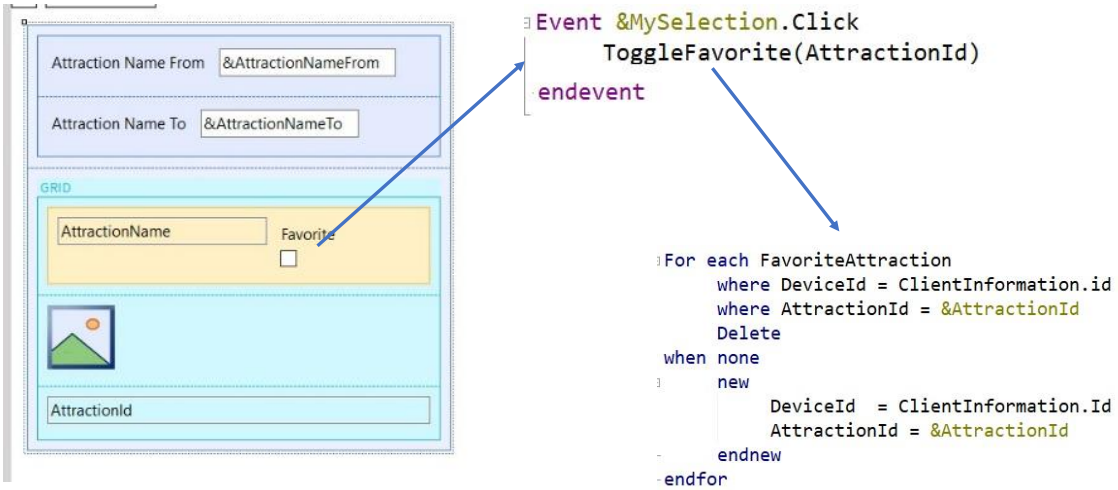


Por otro lado, si observamos lo que se cargará en el ContentPlaceHolder para nuestro Web panel... vemos que es una pantalla con este combo box para elegir un país, y debajo tenemos otro componente, al que se le envía el valor de ese país, toda vez que lo modifiquemos.

Y si observamos quién se está cargando en ese web component, es este otro objeto web, de tipo componente, que tiene un grid que filtra las atracciones turísticas que mostrará, de acuerdo al país recibido por parámetro y a los propios filtros de la pantalla.
Hemos elegido un grid Flex, que contiene una tabla Canvas, para poder superponer la foto de la atracción turística junto con su nombre y una variable booleana para permitir al usuario marcar o desmarcar la atracción turística como favorita.

Sobre el diseño de todo esto no nos detendremos aquí.
Solo observemos que en el evento Load del grid, que tiene como tabla base a Attraction, para cada atracción turística que va a ser cargada se ejecuta este for each, que busca en la tabla subordinada, FavoriteAttraction, si existe un registro para este dispositivo, el que está ejecutando.

My favorite attractions



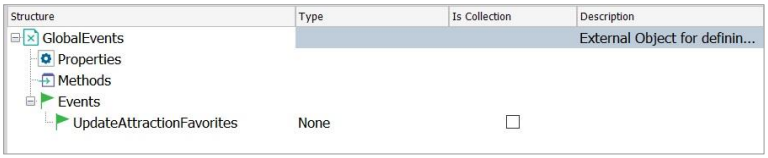
Pero de todo esto, lo que verdaderamente nos importa es la programación del evento click sobre esta variable.

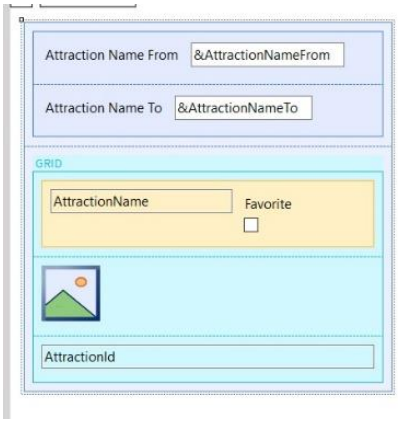
Una vez cargado el grid de atracciones, cuando el usuario hace clic sobre el favorito de una atracción, llamamos a un procedimiento que se fija si la atracción ya estaba marcada como favorita, en cuyo caso la elimina de la tabla. Y si es al revés, hace lo contrario, o sea, la inserta.

Ahora bien, al hacer esto, el total de atracciones favoritas debería actualizarse.

O sea: queremos que un evento de usuario de un componente cargado en este panel permita realizar una acción sobre un componente con el que no guarda otra relación que el encontrarse indirectamente reunidos en la misma pantalla.

External object: GlobalEvents





Event trigger

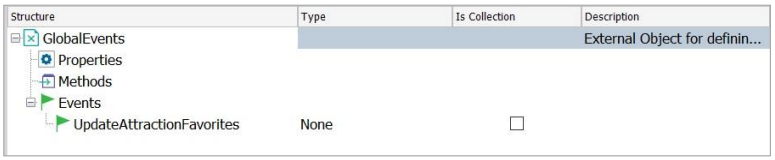
```
Event &MySelection.Click
      ToggleFavorite(AttractionId)
      GlobalEvents.UpdateAttractionFavorites()
endevent
```

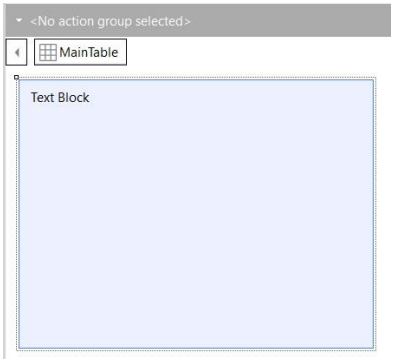
Es entonces que para permitir la interacción toda KB cuenta con este objeto predefinido, Global Events, para que podamos modificarlo.
Podríamos salvarlo con otro nombre y crear tantos objetos particulares como queramos.
En nuestro caso vamos a modificar el predefinido, que viene vacío, agregándole un evento al que llamaremos así.

Los eventos que se vayan agregando podrán manejar parámetros, si se precisan para la comunicación entre componentes.
En nuestro caso solo necesitamos que uno se entere de que otro lo disparó. No necesitamos parámetros.
Tenemos entonces esta forma global de comunicación, así que vamos a utilizarla

Vamos al panel que deberá disparar el evento. El disparo será cada vez que se haga clic sobre la variable. En ese momento invocamos al objeto externo GlobalEvents, evento: el único que hay al momento. Con eso se está reportando el disparo del evento.

External object: GlobalEvents





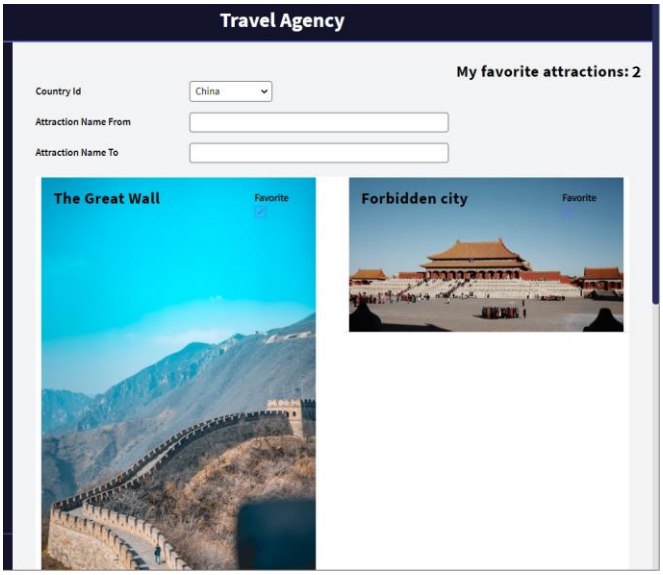
Listen event

```
Event Start
      Do 'GetCount'
Endevent


Sub 'GetCount'
      &FavoriteAttractionss = Count(AttractionId, DeviceId = ClientInformation.id, 0)
      Textblock1.Caption = "My favorite attractions:  " + &FavoriteAttractions.ToString()
endSub


Event GlobalEvents.UpdateAttractionFavorites()
      Do 'GetCount'
endevent
```

Luego, lo que tenemos que hacer es que el componente que cuenta las atracciones para mostrarlas se suscriba a este evento, para poder escucharlo toda vez que se produzca.

Para ello simplemente declaramos escuchar el evento. Y allí programamos lo que queremos que se ejecute cuando el evento ocurre, que en este caso es volver a calcular la cantidad de atracciones.

Run...



Ejecutemos para probar. Desmarcamos y vemos que se está refrescando. Ahora marcamos, marcamos... La comunicación está efectuándose con éxito.

Si bien aquí vimos un ejemplo de comunicación entre paneles web con components, lo mismo vale para comunicar paneles multi-experience con componentes., por ejemplo, para aplicaciones nativas.

Para aprender más sobre este tema, puede dirigirse a nuestro wiki.

GeneXus™

training.genexus.com
wiki.genexus.com