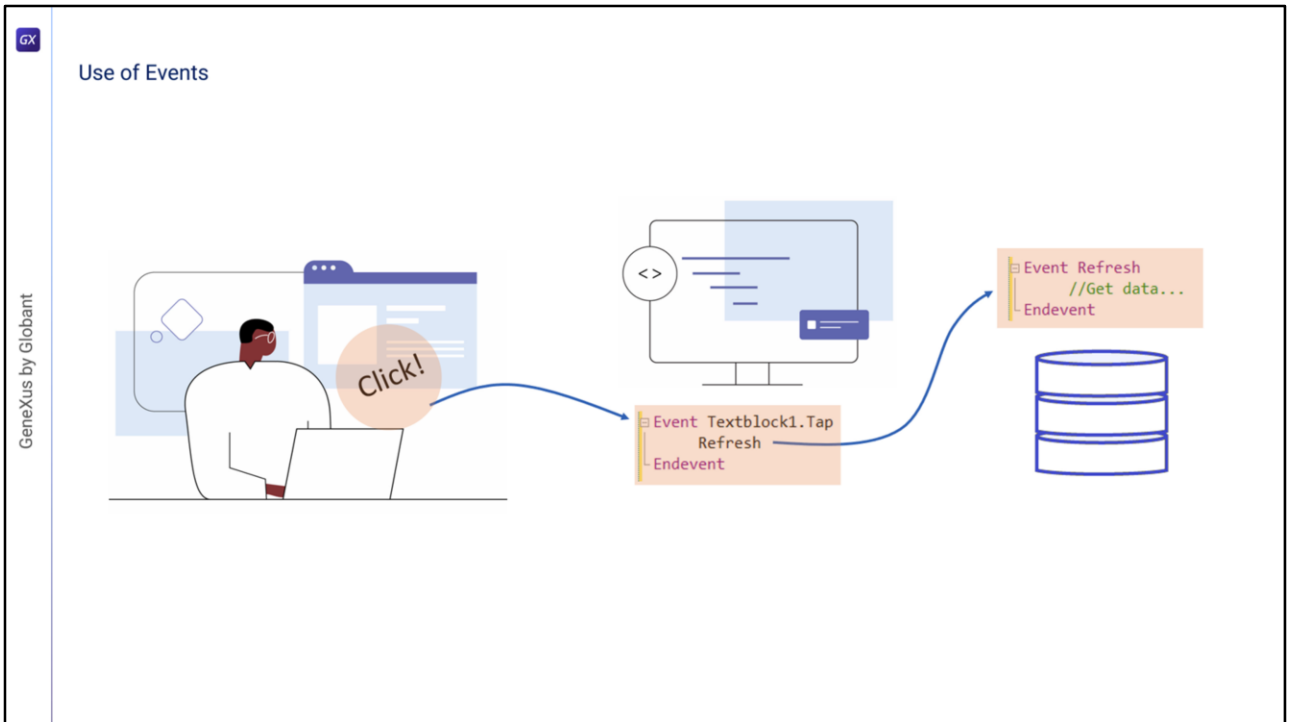


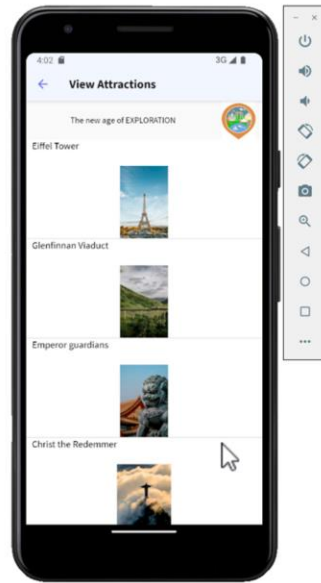
Events in a Panel



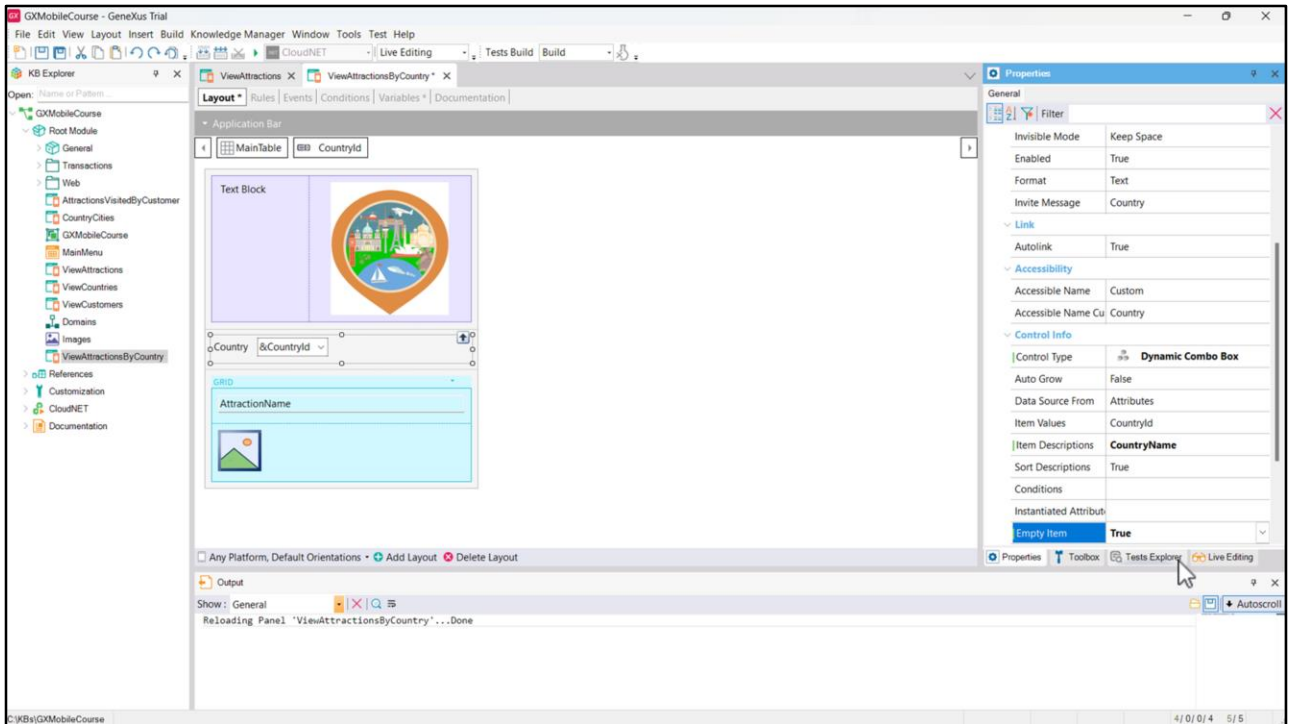
Vanesa Fernández



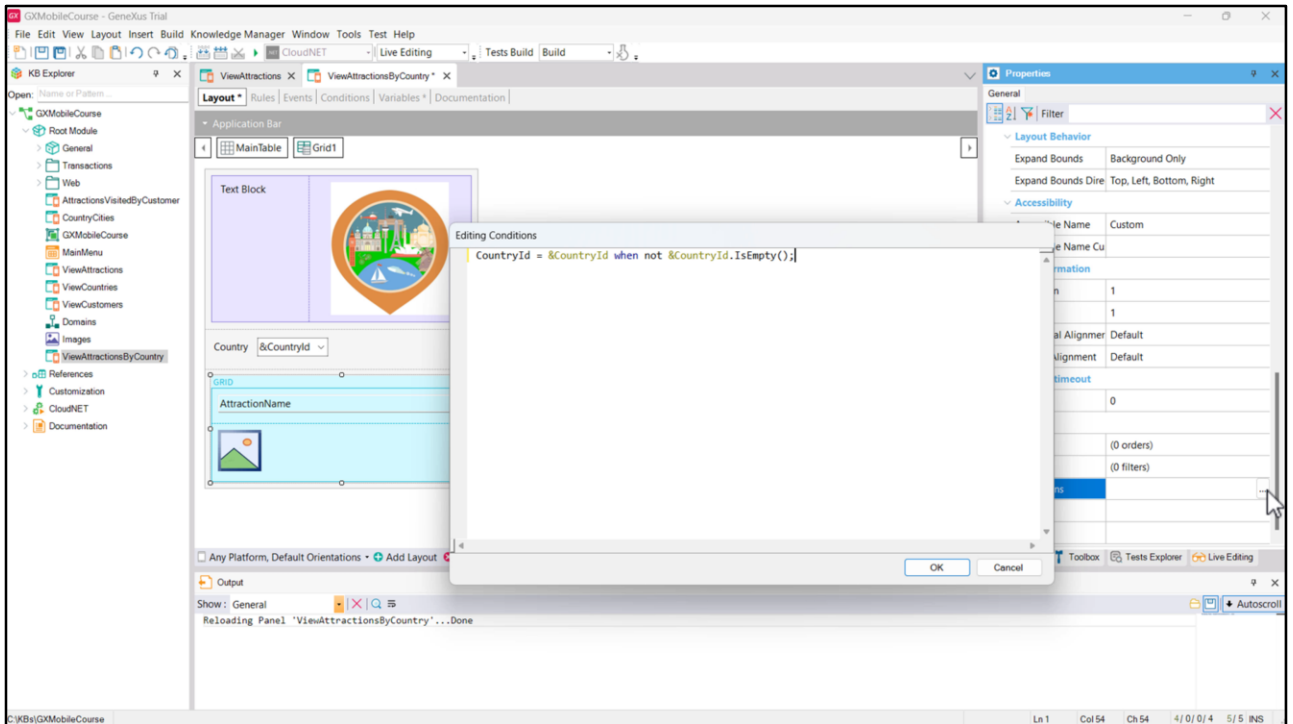
Como ya sabemos, los eventos son mensajes que el software o el sistema nos dan en determinados momentos de la ejecución de nuestra aplicación y eso nos permite programar acciones para que se ejecuten en esos momentos. Por ejemplo, si el usuario hace un clic en un botón, o si escribe algo en un control de texto y se sale del mismo, o si se requiere que el servidor envíe información actualizada, podemos programar una respuesta luego de que se dispara el evento.



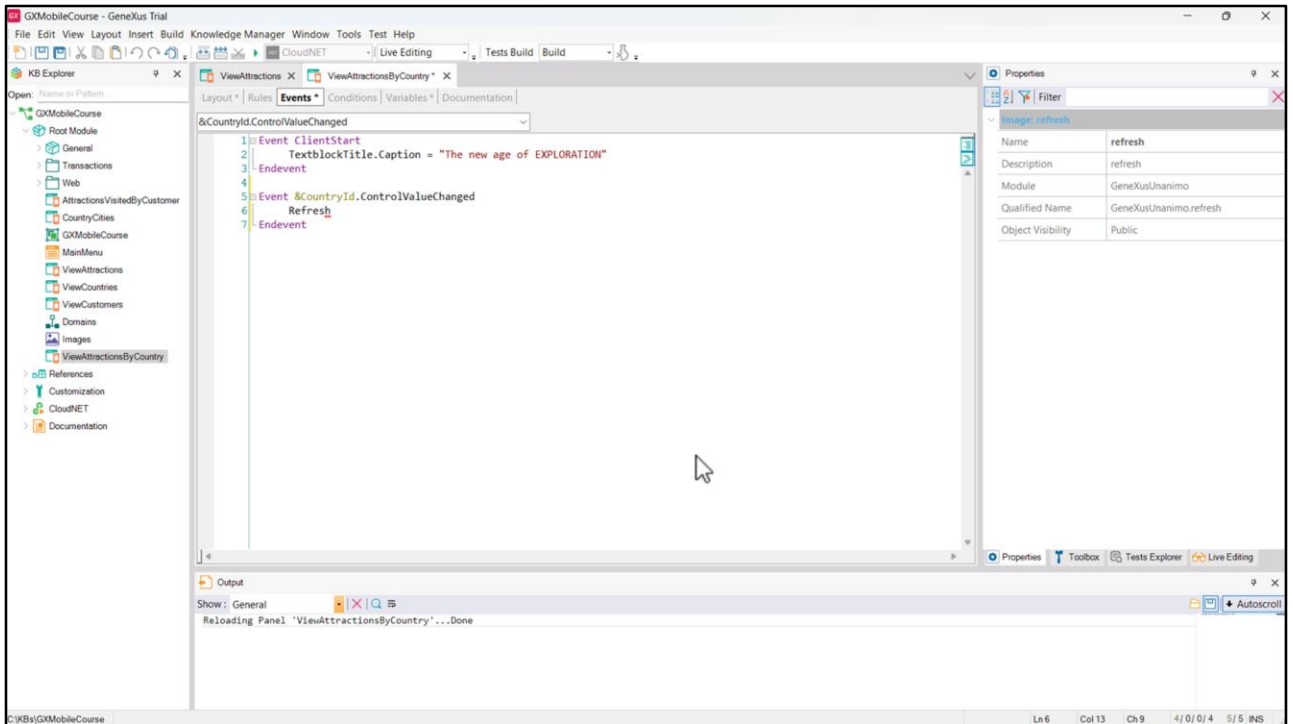
Supongamos que cuando vemos los datos de las atracciones, queremos disponer de filtros para que los datos que se muestren sean acordes a la o las restricciones elegidas.



Para implementar esto, abrimos al Panel ViewAttractions y lo salvamos con el nombre ViewAttractionsByCountry. Ahora vamos a la sección de variables y agregamos una variable de nombre &CountryId que automáticamente queda basada en el atributo CountryId. La arrastramos al form después del textblock del título y antes de la grilla, y en sus propiedades cambiamos la etiqueta de CountryId para Country, cambiamos la propiedad ControlType a Dynamic Combo y ponemos Item Descriptions en CountryName. También asignamos a la propiedad EmptyItem el valor True, para que al inicio aparezca el combo sin ningún valor por defecto.

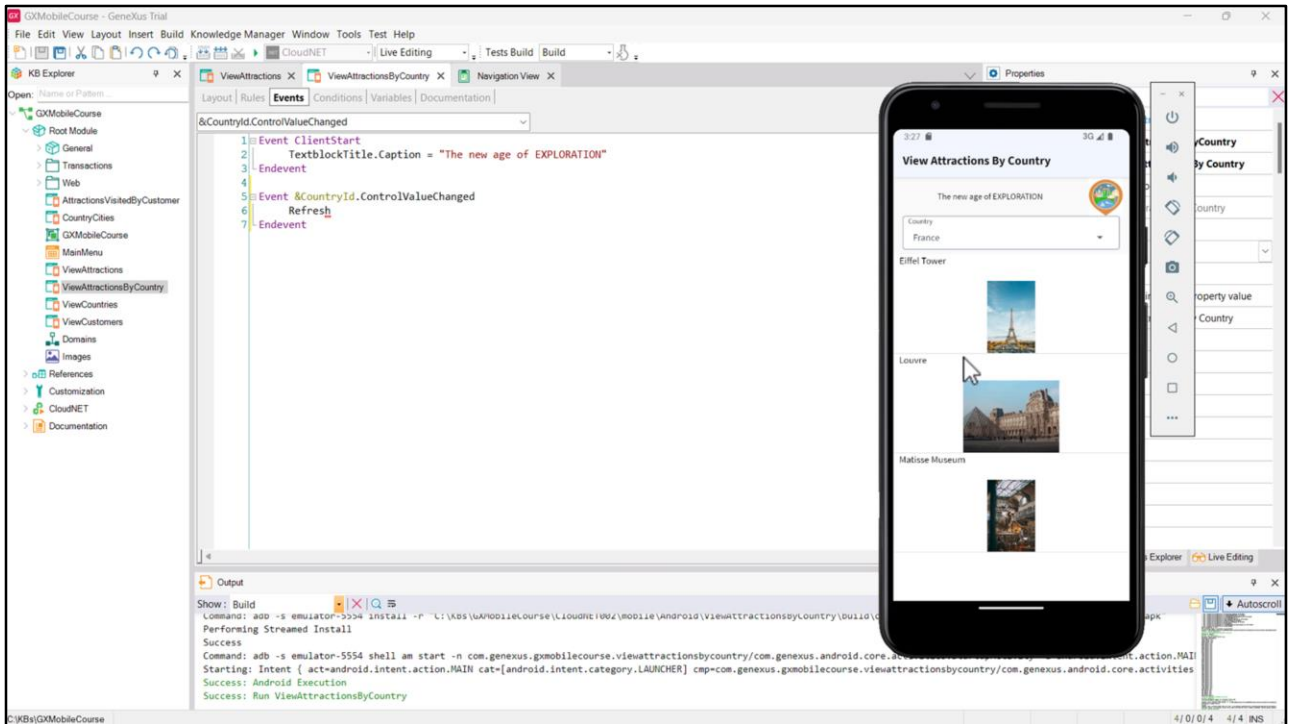


Ahora, para que el filtro tenga efecto en las atracciones que vemos en el grid, editamos las propiedades del mismo y en la propiedad Conditions escribimos: `CountryId = &CountryId when not &CountryId.IsEmpty();` y cerramos con punto y coma. Una vez que elijamos un país, el Panel debe refrescarse para que el grid cargue solamente aquellas atracciones que sean de ese país.



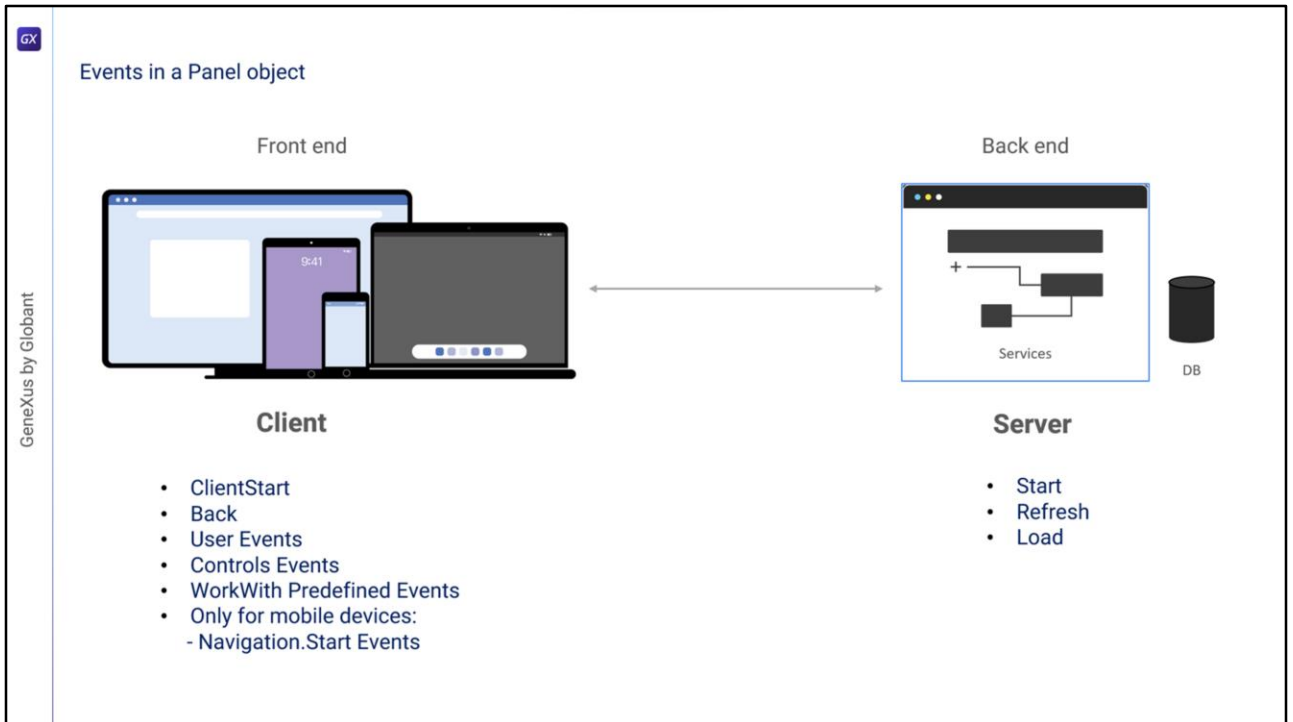
Para eso usamos el evento `ControlValueChanged` del combo dinámico, así que vamos a la solapa `Events` y en el menú elegimos `Insert event`, hacemos clic sobre la variable `&CountryId` y seleccionamos el evento que dijimos.

Vemos que se abre el código del evento para que escribamos lo que queremos que se ejecute cuando se dispare este evento. Escribimos `Refresh`. Este comando provocará que el grid obtenga nuevamente los datos, filtrados esta vez, por el país elegido.



Vamos a setear la propiedad Main program del Panel en True, para dar botón derecho y luego Run. Vemos que, como habíamos puesto la propiedad Empty Item en True, aparece el combo sin un país seleccionado y se muestran todas las atracciones. Si elegimos Francia, vemos que la grilla se recarga y ahora muestra solamente las atracciones turísticas de Francia.

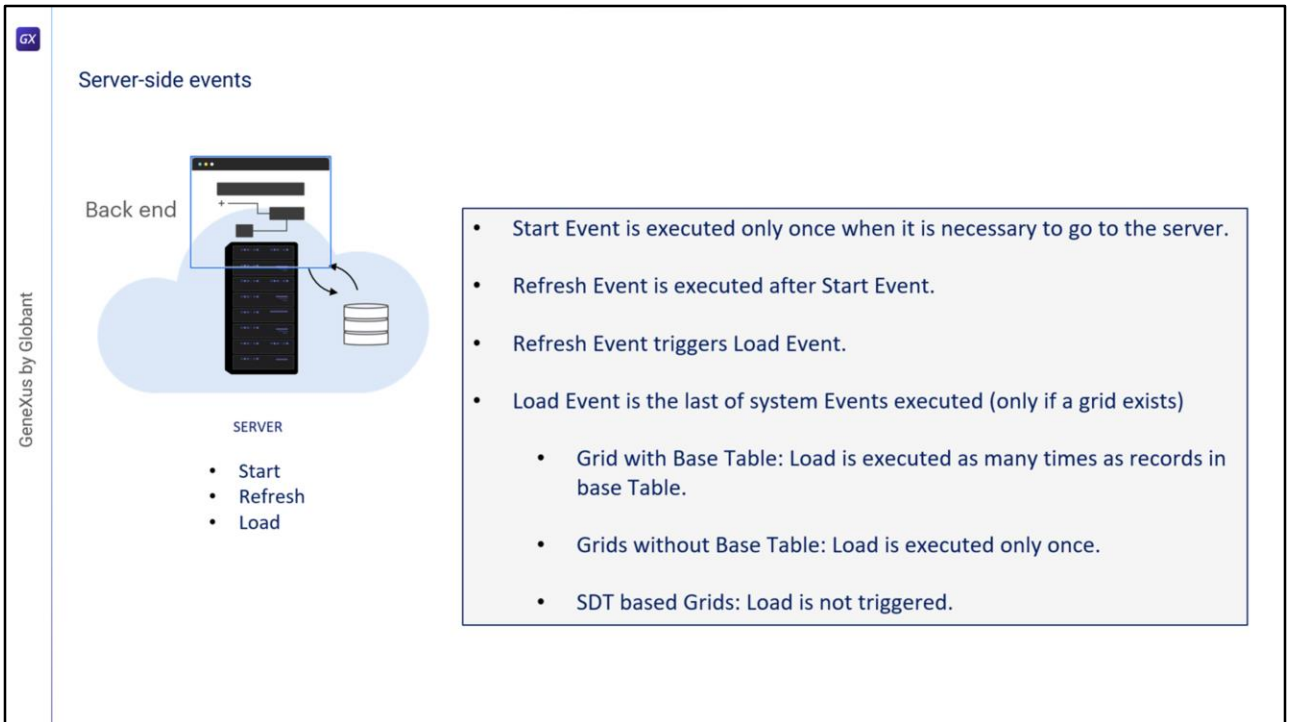
Profundicemos un poco más en el tema de los eventos.



En un objeto Panel tenemos dos tipos de eventos: los que se disparan del lado del cliente y los que se disparan en el servidor.

Los eventos del servidor son los mismos que cuando trabajamos con WebPanels: los eventos Start, Refresh y Load. Estos eventos se dispararán siempre en el servidor para el caso de aplicaciones móviles nativas online. En el caso de que estemos implementando aplicaciones móviles nativas desconectadas, se dispararán internamente en el dispositivo.

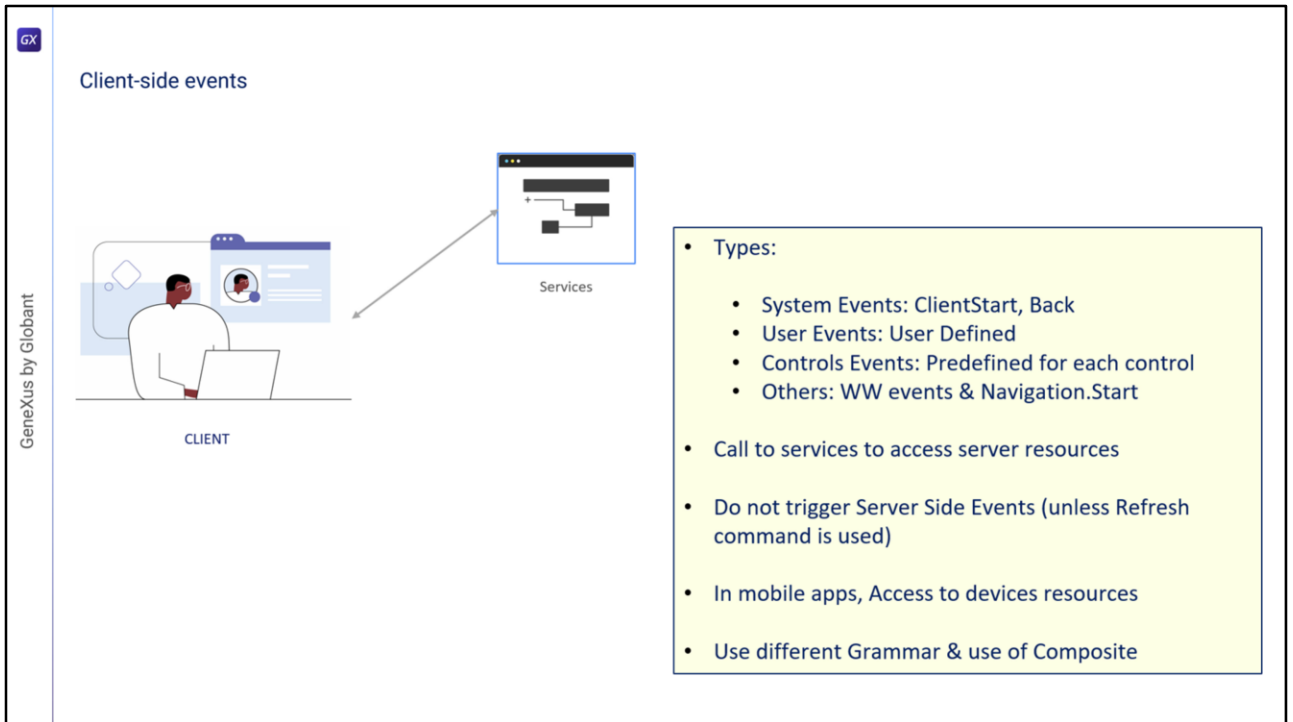
Los eventos del lado del cliente son: el ClientStart, el Back, eventos definidos por el usuario, eventos asociados a los controles de la pantalla y también los eventos predefinidos por el patrón WorkWith. También tendremos los eventos asociados a los estilos de navegación de la información, que dependen, por ejemplo, del tipo de dispositivo y su orientación al iniciarse la aplicación.



Veamos más en detalle los eventos del lado del servidor.

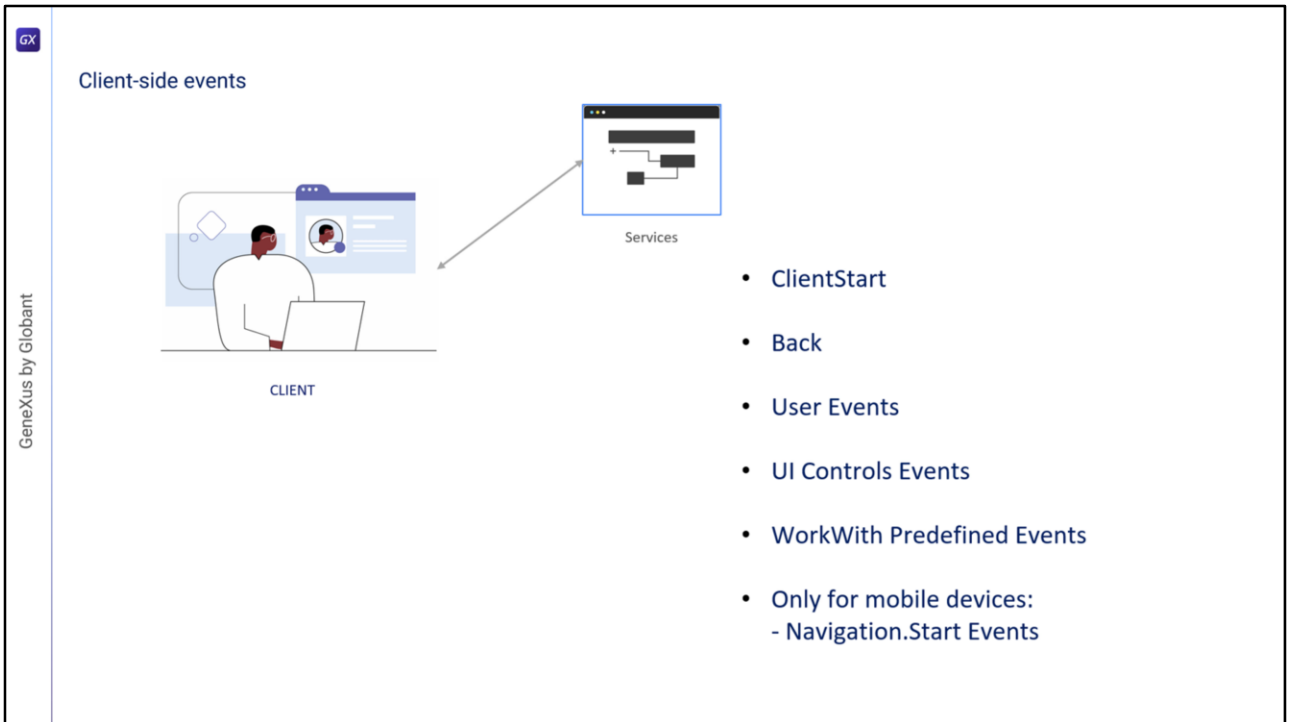
- El primero que se ejecuta es el evento Start. Se ejecuta solo una vez cuando se abre el Panel siempre y cuando sea necesario ir al servidor, y no se ejecutará nuevamente a menos que salgamos del objeto y volvamos a ingresar en él.
- El evento Refresh se ejecuta luego del Evento Start, normalmente una sola vez, pero puede ser invocado nuevamente por medio del comando Refresh. En tal caso, se ejecutará más de una vez y será el primer evento, dado que el Start ya no se volverá a ejecutar. El evento Refresh también se disparará si hacemos un refresh del navegador.
- Cuando se invoque al evento Refresh, al finalizar se disparará el evento Load. Esto es solamente si hay al menos un grid en el Panel y el grid no es una variable SDT colección.
- El evento Load es el último de los eventos del sistema a ser ejecutado, y
- si tiene Tabla Base se ejecutará tantas veces como registros existan en la tabla base;
- si la grilla no tiene tabla base se ejecutará solo una vez
- y si la grilla está basada en un SDT no se ejecutará el evento Load.

Es importante tener en cuenta que cuando trabajamos en una aplicación para dispositivos móviles, en el código de los eventos Start, Refresh y Load no se tiene acceso a los recursos del dispositivo, por ejemplo la cámara, el GPS, etc.



Veamos ahora los eventos del lado del Cliente. Estos eventos son la respuesta de la aplicación a la interacción del usuario.

- Hay varios tipos de eventos en el cliente: los eventos de sistema como el ClientStart y el Back, los de usuario, eventos de controles en pantalla y otros que veremos a continuación.
- El código asociado a estos eventos se ejecuta en el cliente, a menos que se requiera acceder a un recurso del servidor, por ejemplo, si se quiere acceder a la base de datos. En este caso el cliente deberá invocar a un servicio del servidor.
- Durante la ejecución de un evento del lado del cliente, los eventos del lado del servidor no se ejecutan, a menos que se requieran explícitamente a través del comando Refresh, que provoca que se dispare el evento Refresh del servidor, seguido del evento Load (si hay al menos un grid, como vimos antes).
- En aplicaciones nativas para dispositivos móviles, los eventos del lado del cliente tienen acceso a todos los recursos de hardware y software del dispositivo, como la cámara, GPS, micrófono, calendario, contactos, etc.
- Estos eventos del lado del cliente tendrán una gramática particular diferente a los eventos del lado del servidor.

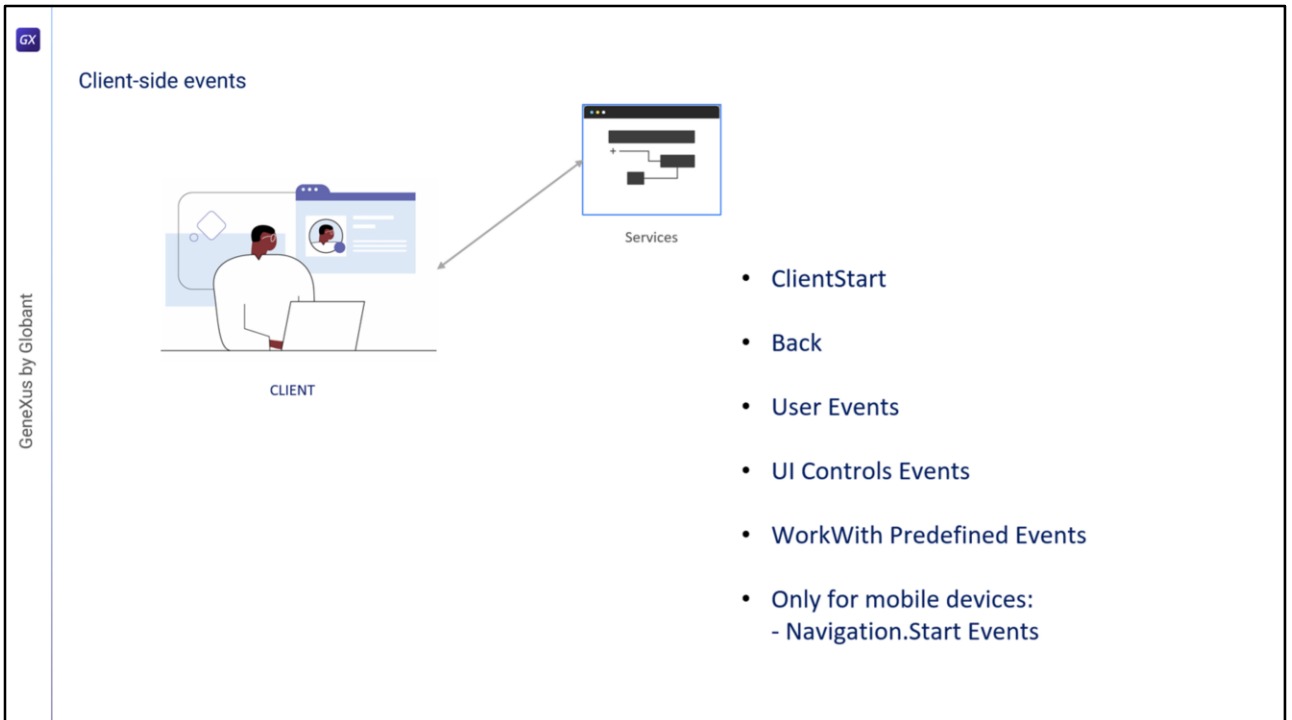


Veamos brevemente cada evento del lado del cliente.

El evento ClientStart es el primer evento que se dispara, incluso antes que el evento Start que se dispara en el servidor y sin necesidad de que haya ninguna interacción del usuario con la aplicación. Se utiliza para inicializar la pantalla de inicio y aspectos relacionados con la UI.

El evento Back se utiliza solo en plataformas móviles y permite capturar que el usuario presionó el botón de volver en dispositivos Android o que se realizó el gesto de volver en dispositivos iOS y programar alguna acción apropiada.

Los eventos de Usuario tienen un nombre dado por el usuario y permite que el desarrollador asocie un cierto código que se ejecuta cuando se activa un determinado control en pantalla, al hacer clic (o tap) sobre él.



Los controles en pantalla tienen eventos propios, dependiendo del control que se trate, como: clic (o tap), doble-clic (o doble tap), drag, swipe, `ControlValueChanged`, etc. Estos eventos se disparan cuando ocurren algunas de las acciones mencionadas y el desarrollador puede programar una respuesta de la aplicación a la interacción del usuario.

El patrón `WorkWith` tiene eventos predefinidos que se disparan dependiendo de la acción que realicemos sobre los datos de la entidad a la que aplicamos el patrón. Entre ellos están el evento `Insert`, `Update`, `Delete`, `Save`, `Cancel`, entre otros. Según la parte del objeto `WorkWith` que estemos ejecutando (lista, detalle, etc.) serán los eventos que estarán disponibles.

En los dispositivos móviles, según el tipo de dispositivo y la orientación, cuando iniciemos la aplicación se disparará un evento `Start` que depende de la navegación. Por ejemplo, si estamos usando una Tablet, por defecto la aplicación inicia en modo `Split`, que significa que se muestra la pantalla dividida en dos secciones, con una lista a la izquierda y el detalle del elemento seleccionado a la derecha. En este caso, al iniciarse la aplicación se dispara el evento `Split.Start`. En el caso de los teléfonos, la pantalla mostrará por ejemplo únicamente la lista y para ver el detalle se abrirá otra pantalla independiente. Este modo se denomina `Flip` y es el comportamiento por defecto en estos dispositivos, por lo cual al iniciar la aplicación se disparará el evento `Flip.Start`. Si bien según el dispositivo hay una navegación por defecto, los objetos `main` de aplicaciones móviles tienen una propiedad que nos permite cambiar la forma en que se inicia la aplicación. Estos eventos `Start` asociados al tipo de navegación, se disparan al inicio de la aplicación mobile e inmediatamente después del evento `ClientStart`.

GeneXus by Globant

Client-side events

```
1 Event ClientStart
2   TextblockTitle.Caption = "The new age of EXPLORATION"
3 Endevent
4
5 Event &CountryId.ControlValueChanged
6   Refresh
7 Endevent
```

En el objeto que vimos antes, programamos solamente un evento asociado a un control, no programamos ningún evento de usuario. A la variable de filtro en pantalla, le programamos su evento `ControlValueChanged`, para disparar al método `Refresh` del Grid, lo que provocará que se disparen los eventos `Refresh` y `Load` del servidor para actualizar el contenido de la grilla con el filtro ingresado.

What can be done in a client-side event?

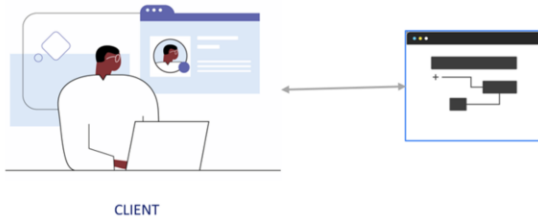


- Call other panels
- Call Rest Services
- Use Business Component
- Call WorkWith objects. In mobile call to Menu
- Call External Objects of GeneXus module
- Call Subroutines
- In addition:
 - Control properties assignments
 - Simple or SDT variable assignment
 - For each Line and Selected Line in grids
 - Use If-Else, Do-Case and Do-While code blocks

Veamos en resumen qué podemos hacer en un evento del lado del cliente.

- Podemos invocar a otro objeto Panel
- También podemos invocar a Servicios Rest y cuando invoquemos a Procedimientos o Data Providers, automáticamente estos serán expuestos como servicios Rest en el servidor. Si esos procedimientos o DPs se hubieran invocado únicamente desde un evento del lado del servidor (dentro de eventos Start, Refresh o Load), no serían expuestos como servicios REST porque no sería necesario.
- Podemos usar Business Components para recuperar o actualizar información, en este caso también esos Business Components serán expuestos como servicios Rest en forma automática.
- Es posible invocar directamente a cualquier nodo del patrón Work With. En el caso de aplicaciones nativas, podemos invocar a un objeto Menu.

What can be done in a client-side event?



- Call other panels
- Call Rest Services
- Use Business Component
- Call WorkWith objects. In mobile call to Menu
- Call External Objects of GeneXus module
- Call Subroutines
- In addition:
 - Control properties assignments
 - Simple or SDT variable assignment
 - For each Line and Selected Line in grids
 - Use If-Else, Do-Case and Do-While code blocks

- Podemos invocar a los objetos externos definidos en el módulo GeneXus. Algunas de estas API's tienen sentido para una plataforma particular, por ejemplo sólo para dispositivos móviles, otras solo en aplicaciones web y otras pueden ser utilizadas en ambas plataformas
- Desde un evento en el cliente podemos llamar a subrutinas
- Y también podemos hacer:
 - Asignación de propiedades a controles
 - Asignación de variables simples o de variables SDT
 - Ejecución de For Each Line y For each Selected Line en grillas
 - Uso de los bloques IF-Else, Do-Case y Do-While

En caso de que intentemos utilizar comandos no permitidos en un evento del lado del cliente, veremos un error en la pantalla de salida, para aquellas líneas que no cumplan con las restricciones de la gramática, por ejemplo, si se intenta usar un comando For Each, un New o cualquier intento de acceder o modificar información que se accede sólo desde el servidor.

Client-side event grammar

COMMANDS

Composite

`<Control>.<Property> = <value>`

If `<Bool_expr>`

Do case... endcase

Do while `<Bool_expr>`

Do-sub (except Menu for Smart Devices)

For each selected line

Simple variable assignation: `&var = <expr>`

SDT or BC elements assignation:

`&SDT.A = <value>`

`&BC.A = <value>`

Return

Refresh

Inside an **expression**:

Variables

Attributes

Constants

Methods

Functions

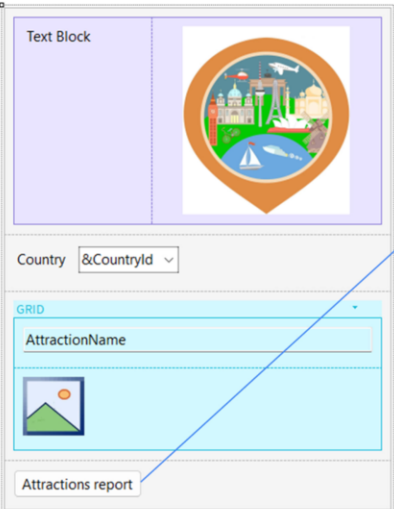
Control properties

Operators (+, -, /, ^)

Aquí vemos un resumen de los comandos que podemos usar en el código de los eventos del lado del cliente. Estas restricciones son solamente para los eventos del lado del cliente, no así para los eventos de lado del servidor (Start, Refresh y Load) donde podemos usar todos los comandos y funciones disponibles en GeneXus. Los comandos aceptados por el momento son los que se muestran.

GeneXus by Globant

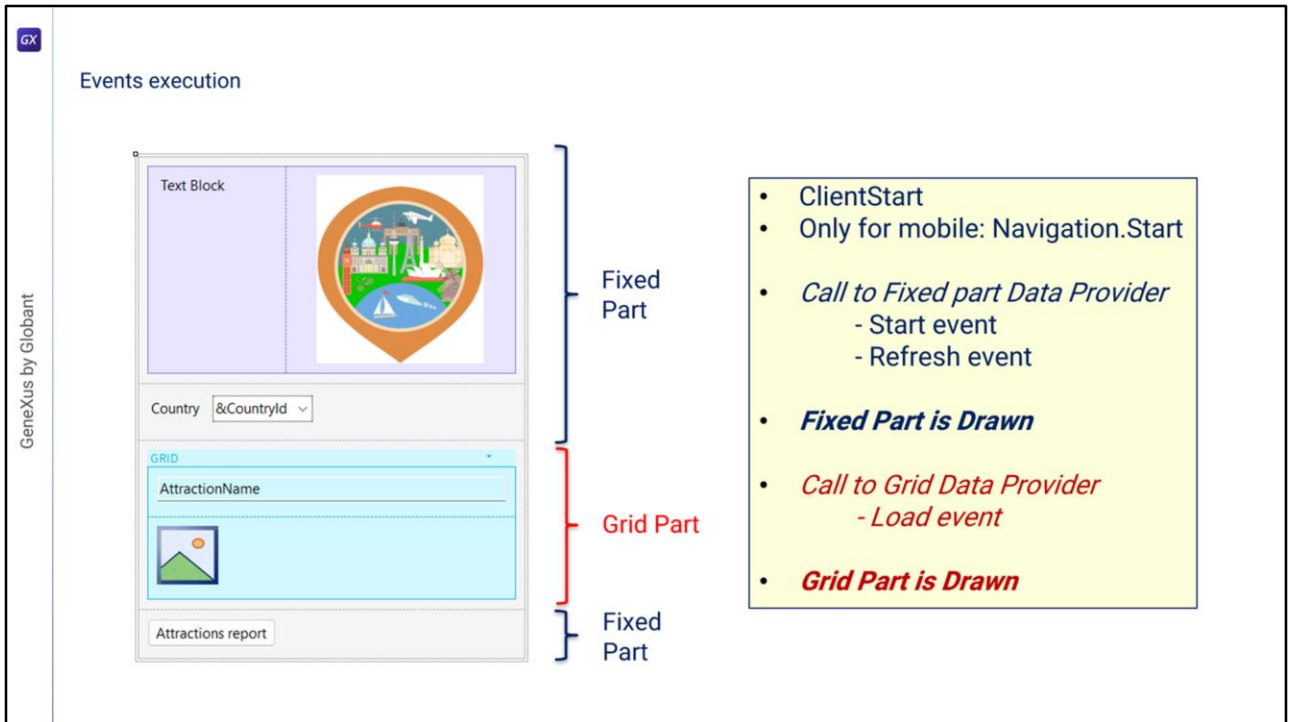
Composite command



```
Event 'Attractions report'  
  Composite  
    AttractionsByName("F", "M")  
  Return  
EndComposite  
Endevent
```

- Stop execution on Error.
- Automatic Error Handling.

Veamos ahora el comando Composite que mencionamos anteriormente. Este comando se utiliza en eventos del lado del cliente en objetos Panel. La importancia de este comando es que, cuando ocurre un error en la secuencia de llamadas, la ejecución se detiene y se manejan los errores automáticamente, desplegándolos en la pantalla sin tener que implementar ninguna programación. Esta es una gran diferencia con los WebPanels, dado que en éstos cuando dentro de un evento un objeto llamado produce un error, no se interrumpe la ejecución, sigue en la sentencia siguiente y es el desarrollador quien debe encargarse de manejar los errores y programar las acciones a tomar. Este comando Composite es opcional: si no lo usamos, el funcionamiento será idéntico al de los WebPanels.



Ahora veamos qué es lo que ocurre con los eventos cuando ejecutamos un objeto Panel.

Primero se ejecuta el evento ClientStart, que se ejecuta solo una vez en el cliente. En caso de aplicaciones móviles luego se dispara el evento Start correspondiente al tipo de navegación establecida en el objeto main, mediante la propiedad Navigation Style. Luego se ejecuta un Data Provider que devolverá los datos necesarios para cargar la parte fija del Panel. Este Data Provider integra la ejecución del código de los eventos Start y Refresh que se ejecutarán en el servidor y devuelve en un único resultado, la información para cargar la parte fija. Luego se dibuja la parte fija del Panel. A continuación, se ejecuta un segundo Data Provider que resolverá la recuperación de los datos requeridos por la grilla. Dentro de la ejecución de este Data Provider se ejecuta el código del evento Load en el servidor. Este evento Load se ejecutará N veces cuando la grilla tenga tabla base, una vez por cada registro, una sola vez si la grilla no tiene tabla base y ninguna vez si la grilla era de una variable SDT colección. Al finalizar la ejecución del Data Provider, el mismo devolverá la información generada por todas estas ejecuciones del evento Load en un único resultado, con el que se cargará el grid y luego, se terminará de dibujar el grid.

GeneXus by Globant

Refresh command

The screenshot displays the GeneXus IDE interface. On the left, a user interface is shown with a 'Text Block' containing an image, a 'Country' dropdown menu with '&CountryId' selected, and a 'GRID' with 'AttractionName' as a column header. Below the grid is an 'Attractions report' button. On the right, the 'Events' tab is active, showing the following code:

```

1 Event ClientStart
2   TextblockTitle.Caption = "The new age of EXPLORATION"
3 -Endevent
4
5 Event &CountryId.ControlValueChanged
6   Refresh
7 -Endevent
8
9 Event 'Attractions report'
10  Composite
11   AttractionsByName("F", "M")
12   Return
13 -EndComposite
14 -Endevent

```

Below the code, a 'Conditions' section shows the condition: `CountryId = &CountryId when not &CountryId.IsEmpty(); ...`. A blue arrow points from the 'Refresh' event in the code to the 'Attractions report' button in the UI.

Una de las cosas que mencionamos fue que la parte fija del Panel se carga en forma independiente de la carga del grid, invocándose Data Providers diferentes, que están publicados en el servidor como servicios y acceden a la base de datos para recuperar la información de cada parte. Cuando cambiamos el valor de un filtro, es necesario que se refresque la información del grid. Para esto, es necesario que agreguemos el método Refresh para que dispare los eventos Refresh y Load del servidor, lo que hará que se cargue nuevamente el grid aplicando las conditions programadas y muestre los resultados filtrados como esperamos. Como el comando Refresh lo debemos invocar luego de que cambiamos el valor de la variable del filtro, usamos el evento ControlValueChanged de la variable, para invocar al método. De esta forma, luego de cambiar el valor del filtro, al salir del campo se disparará el evento correspondiente que esperamos termine refrescando el contenido del grid.

GeneXus by Globant

Refresh command

```

1 Event ClientStart
2   TextblockTitle.Caption = "The new age of EXPLORATION"
3 -Endevent
4
5 Event &CountryId.ControlValueChanged
6   Refresh
7 -Endevent
8
9 Event 'Attractions report'
10  Composite
11    AttractionsByName("F", "M")
12    Return
13  EndComposite
14 -Endevent

```

Conditions: CountryId = &CountryId when not &CountryId.IsEmpty(): ...

Pero ¿qué sucede cuando se invoca el comando Refresh dentro de un evento del lado del cliente? En esta arquitectura, como el objetivo es que la página se cargue la menor cantidad de veces posible, se prioriza el caché de datos, es decir que se trata siempre de recuperar la información previamente almacenada. Esto quiere decir que, dependiendo de la configuración de almacenamiento en caché, se decide si se debe ir o no a recuperar datos del servidor. En caso de que se decida ir al servidor, si no hay cambios en los datos del servidor, no se devuelve nada al cliente. De lo contrario, sí se ejecutan los eventos Refresh y Load (si hay un grid que no sea basado en un SDT, como es nuestro caso) y se actualiza la parte fija y la parte variable del Panel, como esperábamos.



En siguientes videos veremos otros aspectos del diseño e implementación de objetos Panel.