Eventos en un objeto panel

Eventos del lado del cliente y del lado del servidor

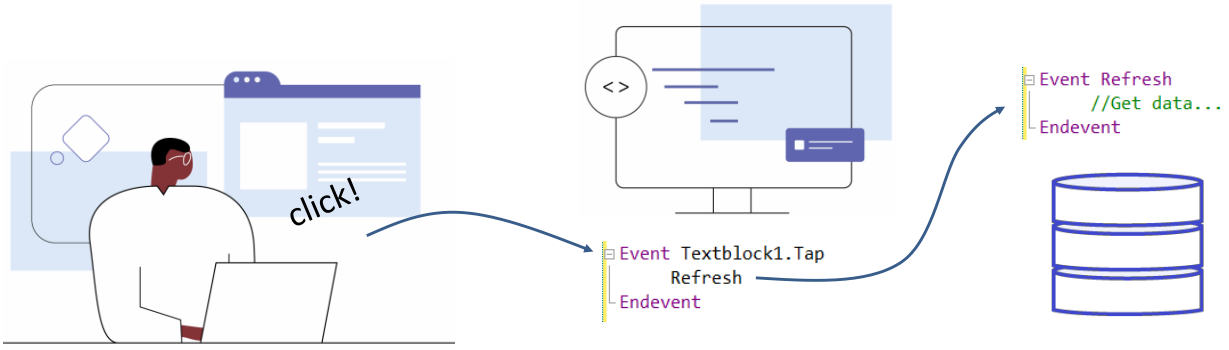
GeneXus™

Tanto sea una aplicación web generada en Angular, como una aplicación generada para dispositivos móviles en Android o iOS, utilizamos objetos panel para implementarla.

Por esa razón, los eventos disponibles en este objeto, son en su mayoría válidos para ambas plataformas.

En este video, todo lo que hablemos de eventos en un objeto panel, será aplicable a ambos tipos de plataformas y haremos una excepción explícita cuando sea el caso.

Uso de los eventos

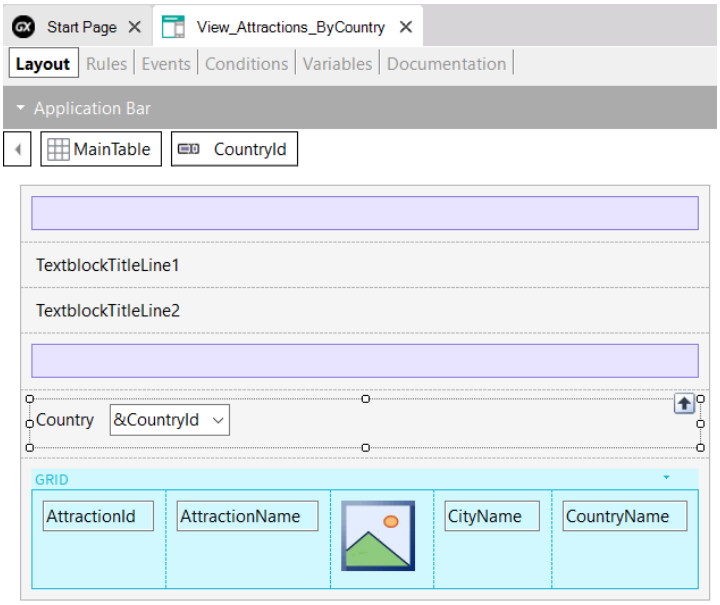


Como ya sabemos, los eventos son mensajes que el software o el sistema nos dan en determinados momentos de la ejecución de nuestra aplicación y eso nos permite programar acciones para que se ejecuten en esos momentos.

Por ejemplo si el usuario hace un click 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 veíamos los datos de las atracciones, quisiéramos disponer de filtros para que los datos que se muestren sean acordes a la restricción elegida.

Un ejemplo sencillo

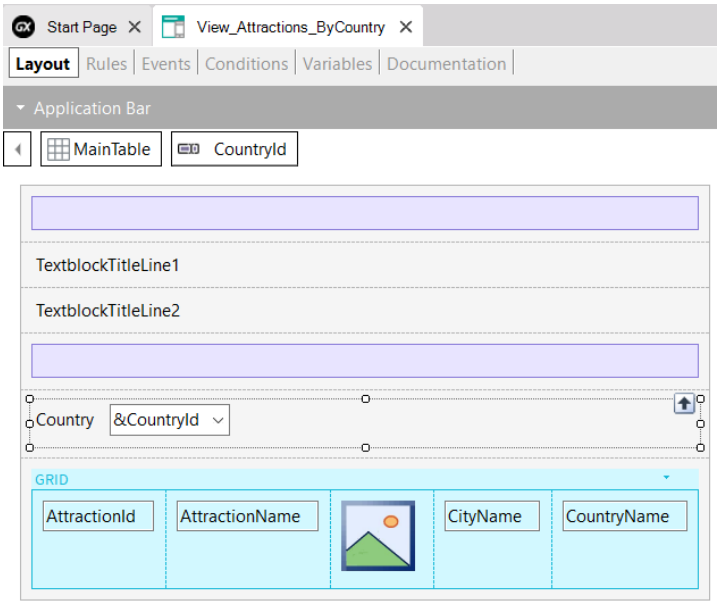


Para implementar esto, abrimos al panel View_Attractions y lo salvamos con el nombre View_Attractions_ByCountry.
Ahora vamos a la sección variables y agregamos una variable &CountryId que automáticamente nos queda del tipo del atributo CountryId.
La arrastramos al form después de los textblocks del título 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 grid y en la propiedad Conditions escribimos: CountryId = &CountryId when not &CountryId.IsEmpty() y cerramos con punto y coma. Aprovechamos que estamos en las propiedades del grid y seteamos la propiedad Auto Grow en el valor True para que la grilla autoajuste su tamaño y pueda mostrar todos los registros.

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 &CountryId y seleccionamos al evento que dijimos.

Un ejemplo sencillo (cont.)



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.

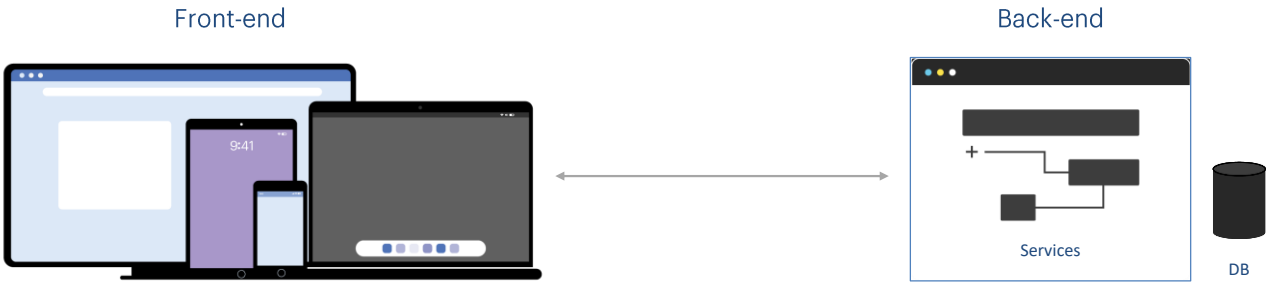
Para que el servidor entienda que queremos traer nuevos datos, hay que hacer que se cambie la URL enviada al servidor, de forma que éste interprete que es una página nueva y obtenga la información correspondiente para enviarla al cliente.
Para eso, agregamos una regla Parm con las variables que usamos en los filtros, para que cuando cambien su valor se actualice la página. En este caso agregamos únicamente a la variable &CountryId.

Damos botón derecho sobre el panel View_Attractions_ByCountry y elegimos Run.

Vemos que, como habíamos puesto la propiedad EmptyItem en True, aparece el combo sin un país seleccionado y se muestran todas las atracciones.
Si elegimos China, vemos que la grilla se recarga y ahora muestra solamente a las atracciones turísticas de China.

Profundicemos un poco más en el tema de los eventos.

Eventos del cliente y del servidor



Client

- ClientStart
- Back
- User Events
- Controls Events
- WorkWith Predefined Events
- Only for mobile devices:
  - Navigation.Start Events

Server

- Start
- Refresh
- Load

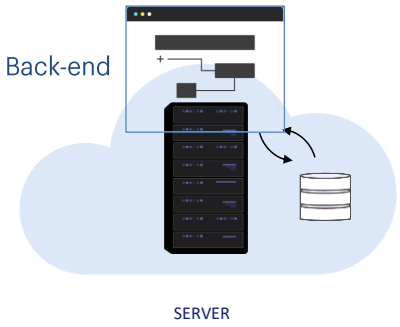
En un objeto panel, tenemos dos tipos de eventos, los eventos que se disparan del lado del cliente y los eventos que se disparan en el servidor.

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 Angular y 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, que es un patrón que al aplicarlo permite generar una serie de panels que cumplen con la funcionalidad de trabajar con una entidad (para realizar inserciones, modificaciones, eliminaciones, navegación de datos, filtros, etc.), en forma análoga al patrón WorkWith que usamos cuando trabajamos con webpanels.

En el caso de que los objetos panel los usemos para generar una aplicación para dispositivos móviles, 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.

Eventos del lado del servidor



- Start
- Refresh
- Load

- Start Event execute only once.
- Refresh Event is executed after Start Event
- Refresh Event trigger Load Event
- Load Event is the last of system Events executed (only if a grid exist)
  - Grid with Base Table: Load is executed as many times as records in base Table.
  - Grids without Base Table: Load is executed only once.
  - SDT based Grids: Load is not triggered.

Veamos ahora 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 y no se ejecutará nuevamente a menos que salgamos del objeto y volvamos a ingresar en él.

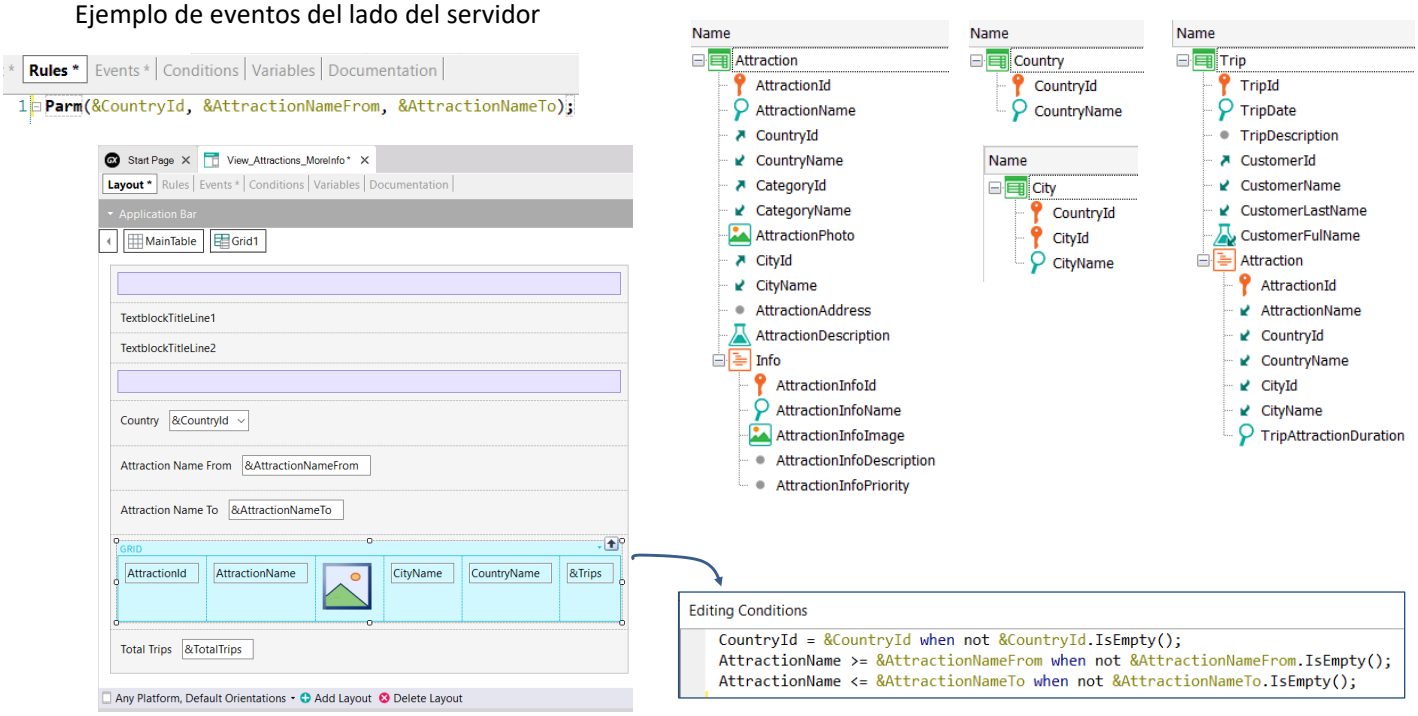
- El evento Refresh se ejecuta luego del Evento Start, normalmente una sola vez, pero puede ser invocado nuevamente por medio del comando Refresh, en ese caso se ejecutará más de una vez y será el primer evento, ya que Start ya no se volverá a ejecutar.

- 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 ejecutados 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 esta basada en un SDT no se ejecutará el evento load.

En el caso de que estemos 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.

Ejemplo de eventos del lado del servidor



Estudiemos el caso del siguiente panel, que nos permite ver una grilla de atracciones y filtrar la grilla por país y nombre de atracción.

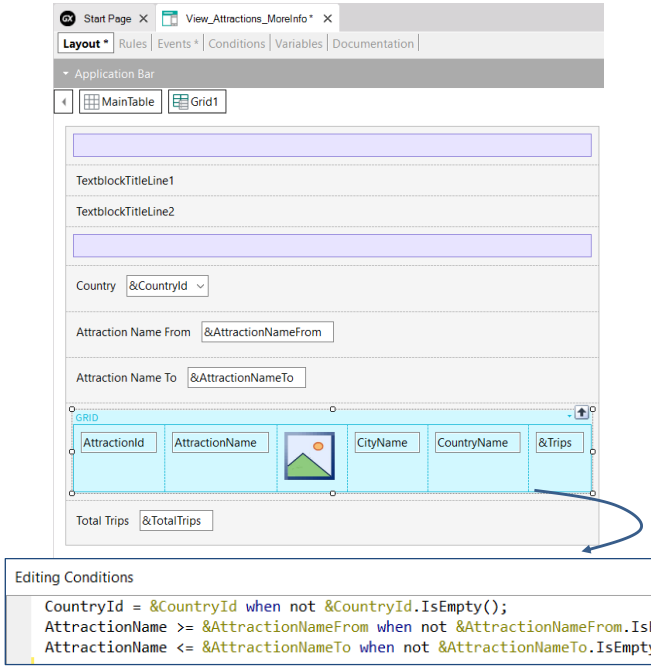
Al panel View_Attractions_ByCountry que vimos recién, lo salvamos con el nombre View_Attractions_MoreInfo, le agregamos variables para filtrar por nombre de atracción y también un par de variables para totalizar los viajes. Aquí vemos al panel con todos los cambios.

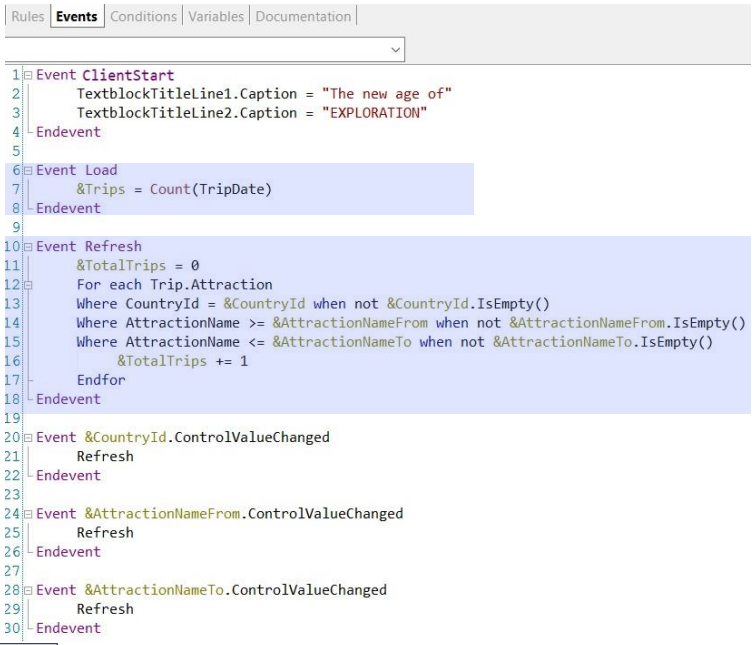
Ahora, en la grilla están presentes los atributos AttractionId, AttractionName, AttractionPhoto, CityName, CountryName y una variable &Trips que nos mostrará los viajes que sea han realizado a cada atracción.

Debajo de la grilla, vemos el total global de viajes realizados a todas las atracciones.

También podemos ver el modelo de las transacciones utilizadas en este panel y las conditions del grid que nos permite realizar los filtros.

Ejemplo de eventos del lado del servidor





En la solapa de eventos, vemos que están programados los eventos Refresh y Load que se disparan del lado del servidor.

En el evento Load calculamos el total de viajes de una atracción mediante una fórmula que accede a la tabla TripAttraction, correspondiente al segundo nivel de la transacción Trip. Observemos que en la fórmula Count estamos haciendo referencia al atributo TripDate, y debido a los atributos presentes en el grid, la tabla base del grid (o sea de la parte variable del panel) será TripAttraction, por lo que se contarán los viajes correspondientes a cada atracción.

En el evento Refresh, programamos un For Each para que acceda a la tabla TripAttraction para contar el total global de viajes de las atracciones, de modo que cuando se dibuje la parte fija, ya se tengan los datos correspondientes al total de viajes.

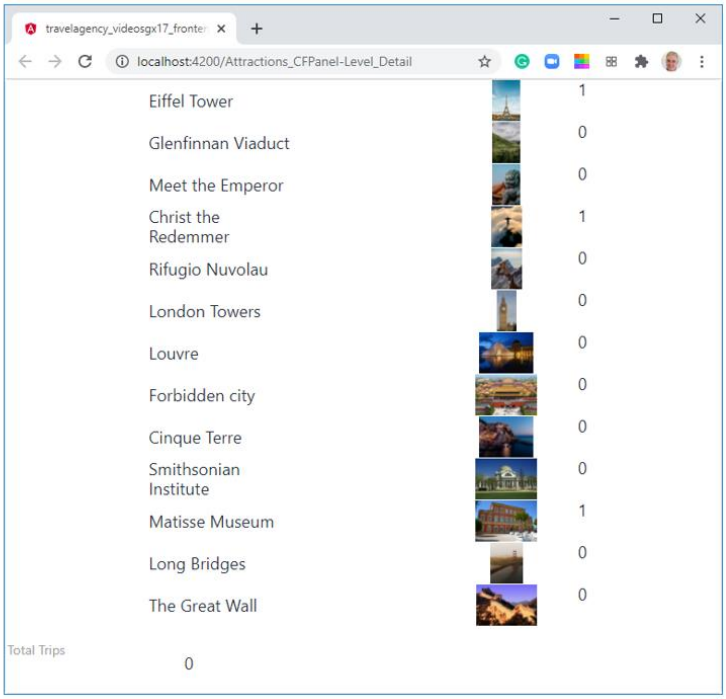
[*Si fuera un Web Panel se haría en el evento Load, pero como es un Panel se hace en el evento Refresh*]

Total global de viajes mal calculado!

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

Events

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

Event Refresh
    &TotalTrips = 0
Endevent
```



[*Veamos un fragmento del video Objeto Panel. Primeros pasos*]

Si vamos a la aplicación ejecutando en el navegador, vemos que aparece el total de viajes de cada atracción en forma correcta, pero que el total acumulado de viajes sale en 0.
¿Qué hicimos mal? La variable &TotalTrips se está incrementando en el evento Load como hicimos en el webpanel y tenemos la seguridad de que este evento se está disparando porque vemos que algunas atracciones tienen viajes…. ¿Entonces?

La razón es que los objetos panels no funcionan igual que los web panels. En los objetos panel, la parte fija se carga en forma independiente de la grilla.
En este caso, la variable &TotalTrips está en la parte fija del panel, que es lo que primero se carga y luego se produce la carga del grid, por lo tanto cuando se muestra la variable aún no hay podido cargarse el grid, no se ha disparado aún el evento Load y no se ha podido acumular el valor de &TotalTrips.

Recordemos que en un objeto panel usado para desarrollar aplicaciones con foco en customer-facing, para cargar la pantalla en el dispositivo cliente, se invoca a servicios ubicados en el servidor que son los que acceden a la base de datos. Estos servicios son data providers, que son independientes para la parte fija y para el grid (o cada grid) de la pantalla.

Cuando se inicia la ejecución del panel, se dispara un evento local en el cliente que invoca al data provider para cargar la parte fija y hace que se disparen los eventos Start y Refresh en el servidor. Luego se ejecuta un segundo data provider que dispara el evento Load en el servidor N veces y se carga la grilla.

En nuestro ejemplo, al dispararse el Refresh primero, se inicializa la variable &TotalTrips y se carga la parte fija, recién después se dispara el evento Load que es donde &TotalTrips se carga con el valor correcto y se actualiza el grid, pero la parte fija ya se cargó antes y no se vuelve a dibujar.

Debido a esta característica no podemos programar el objeto panel como si fuera un webpanel.

En otro video entraremos en detalle en el disparo de los eventos y en la determinación de las tablas base, pero por ahora, para que nos funcione el ejemplo, vamos a cambiar la programación.

Solución correcta

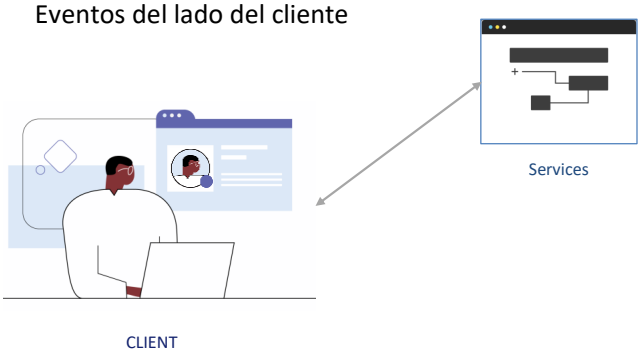
```
Event Load
     &Trips = Count(TripDate)
Endevent

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

La solución es incluir en el evento Refresh un For Each que acceda a la tabla TripAttraction y cuente el total global de viajes. No nos olvidemos de eliminar el cálculo que teníamos en el evento Load.

La razón por la cual incluimos la actualización en el evento Refresh, utilizando un For Each, es porque el evento Refresh se disparará cuando se cargue la parte fija, que va a ser antes de cuando se cargue el grid.
Por lo tanto al cargar la parte fija el valor de &TotalTrips tendrá el valor correcto y recién después se actualizará la parte del grid.

[*Fin del repaso*]

Eventos del lado del cliente

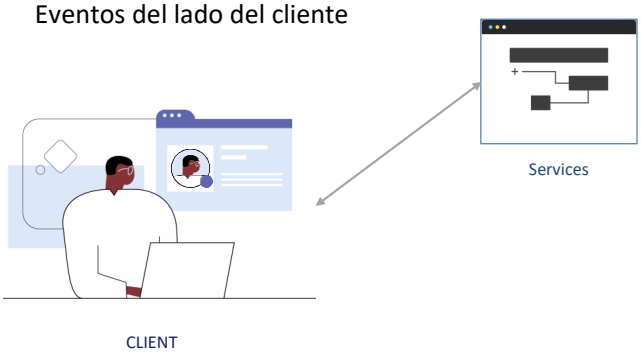


- Types:
  - System Events: ClientStart, Back
  - User Events
  - Screen Controls Events
  - Others: WW events & Navigation.Start
- Call to services to access server resources
- Do not trigger Server Side Events (unless Refresh command is used)
- In mobile apps, Access to devices resources
- Use different Grammar & use of Composite

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).
- Si estamos ejecutando una aplicación nativa para dispositivos móviles, los eventos del lado del cliente tienen acceso a todos los recursos de hardware y sofware 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.

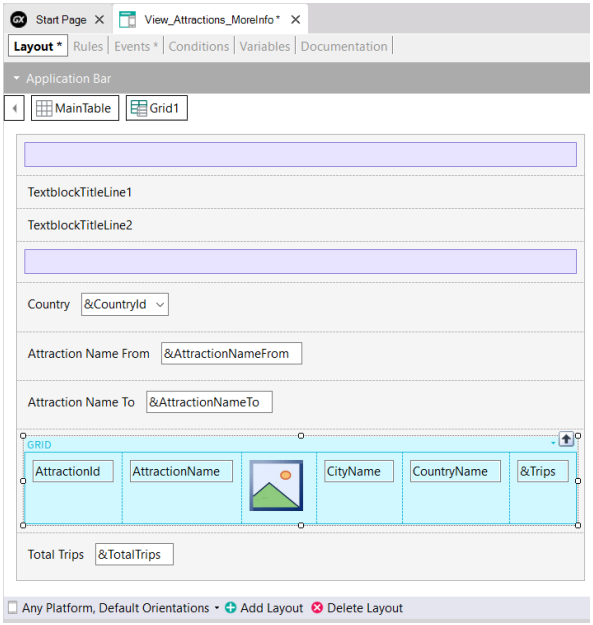
Eventos del lado del cliente

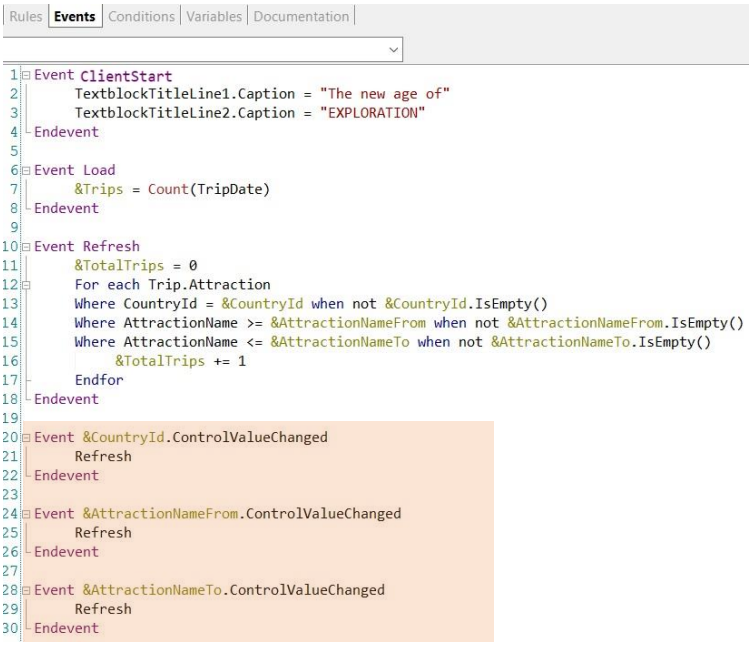


- ClientStart
- Back
- User Events
- UI Controls Events
- WorkWith Predefined Events
- Only for mobile devices:
  - Navigation.Start Events

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 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 ejecute cuando se activa un determinado control en pantalla, al hacer click (o tap) sobre él.
Los **controles en pantalla** tienen eventos propios, dependiendo del control que se trate, como: click (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 pueda 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 el caso de 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.

Ejemplos de eventos del lado del cliente



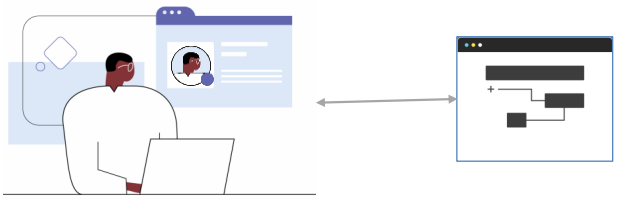



```
Event ClientStart
     TextblockTitleLine1.Caption = "The new age of"
     TextblockTitleLine2.Caption = "EXPLORATION"
Endevent

Event Load
     &Trips = Count(TripDate)
Endevent

Event Refresh
     &TotalTrips = 0
     For each Trip.Attraction
     Where CountryId = &CountryId when not &CountryId.IsEmpty()
     Where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty()
     Where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty()
          &TotalTrips += 1
     Endfor
Endevent

Event &CountryId.ControlValueChanged
     Refresh
Endevent

Event &AttractionNameFrom.ControlValueChanged
     Refresh
Endevent

Event &AttractionNameTo.ControlValueChanged
     Refresh
Endevent
```

En el objeto que vimos antes, programamos solamente eventos asociados a controles, no programamos ningún evento de usuario.

A cada variable de los filtros 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 los filtros ingresados.

Qué podemos hacer en eventos en el cliente



CLIENT

- Call to other panels
- Call to Rest Services
- Use of Business Component
- Call to WorkWith objects. In mobile call to Menu
- Call Externals Objects of GeneXus module For Angular apps see: https://wiki.genexus.com/commwiki/servlet/wiki?46456
- Call Subroutines
- We can do:
  - 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.
- 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. Para conocer las que no pueden ser accedidas por aplicaciones Angular, visite la página del wiki que se muestra en pantalla (https://wiki.genexus.com/commwiki/servlet/wiki?46456)
- 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.

Gramática de eventos del lado del cliente

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

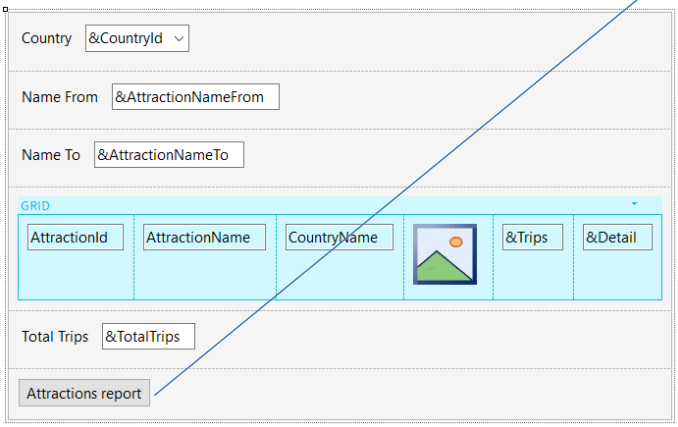
Inside an **expression**:
- Variables
- Attributes
- Constants
- Methods
- Functions
- Control properties
- Operators (+, -,, /, ^)

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.

Comando Composite




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

- Only for Client Side Events with more than 1 line of Code.
- Stop execution on Error.
- Automatic Error Handling.

Veamos ahora el comando composite que mencionamos anteriormente.

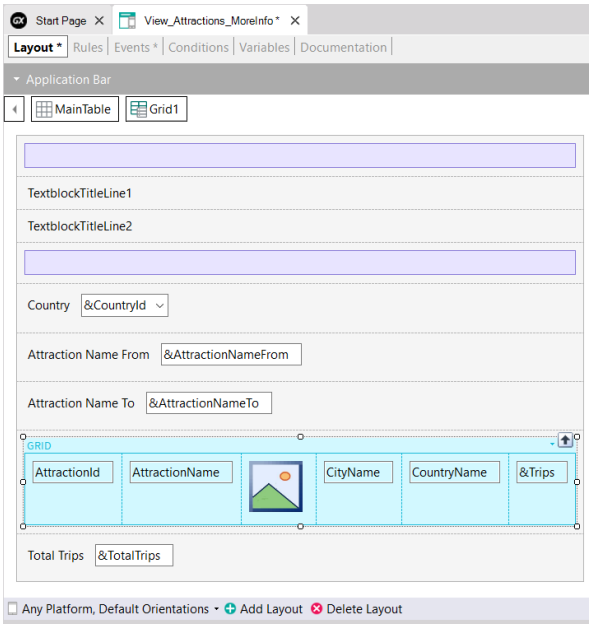
Este comando se utiliza en eventos del lado del cliente en objetos panel.

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.

Orden de ejecución de los eventos



- ClientStart
- Only for mobile: Navigation.Start
- *Call to Fixed part Data Provider*
  - Start event
  - Refresh event
- ***Fixed Part is Drawn***
- *Call to Grid Data Provider*
  - Load event
- ***Grid Part is Drawn***

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

Primero se ejecuta el evento ClientStart, 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.

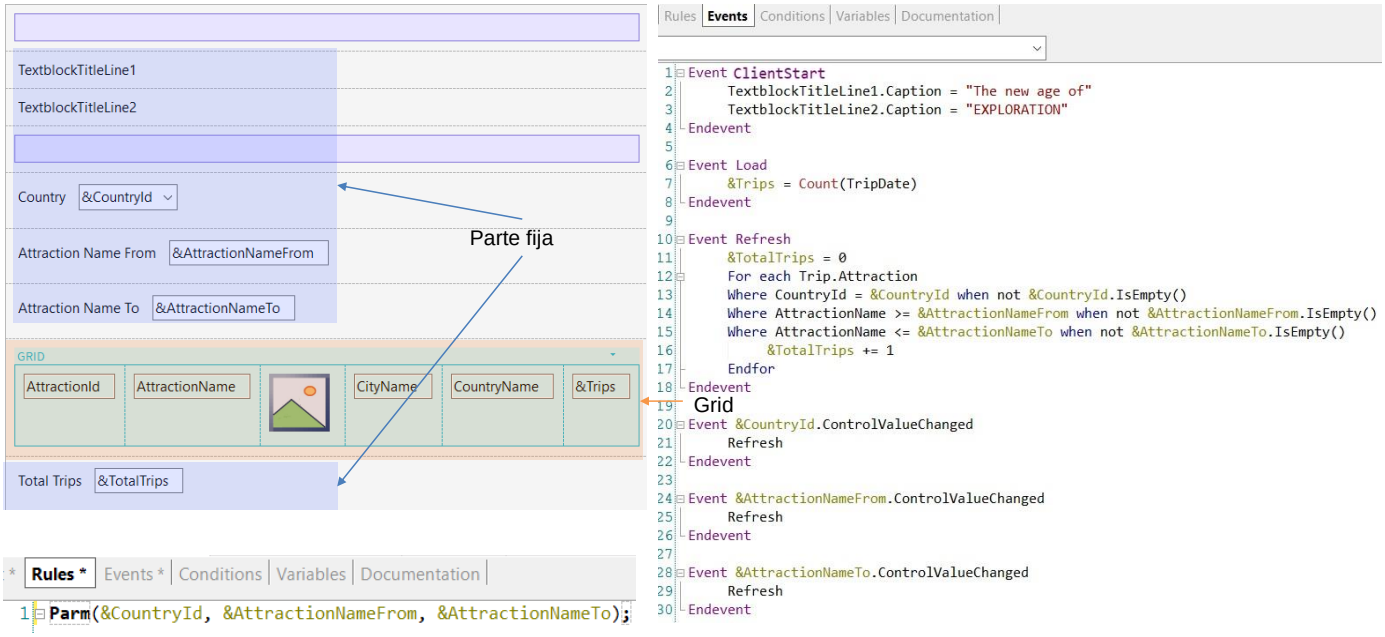
Luego se ejecuta un data provider que devolverá los datos necesarios para cargar la parte fija del panel. Este data provider integra la ejecución del código de los eventos Start y Refresh que se ejecutarán en el servidor y devuelve en un único resultado, la información para cargar la parte fija.

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 ejecutara 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.

Uso del comando Refresh en eventos del cliente



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 suponemos hará que se cargue nuevamente el grid, aplicando las conditions programadas y muestre los resultados filtrados como esperamos.

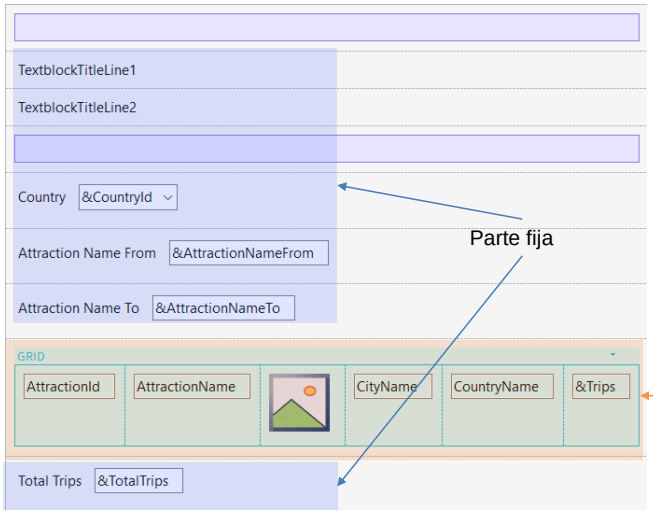
Como el comando Refresh lo debemos invocar luego de que cambiamos el valor de la variable del filtro, usamos el evento ControlValueChanged de cada variable, para invocar al método. De esta forma, luego de cambiar un valor del filtro, al salir del campo se disparará el evento correspondiente que esperamos termine refrescando el contenido del grid.

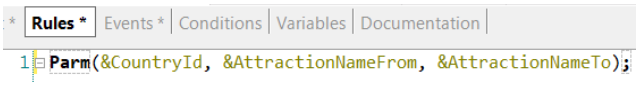
Pero ¿qué sucede cuando se invoca el comando Refresh dentro de un evento del lado del cliente?

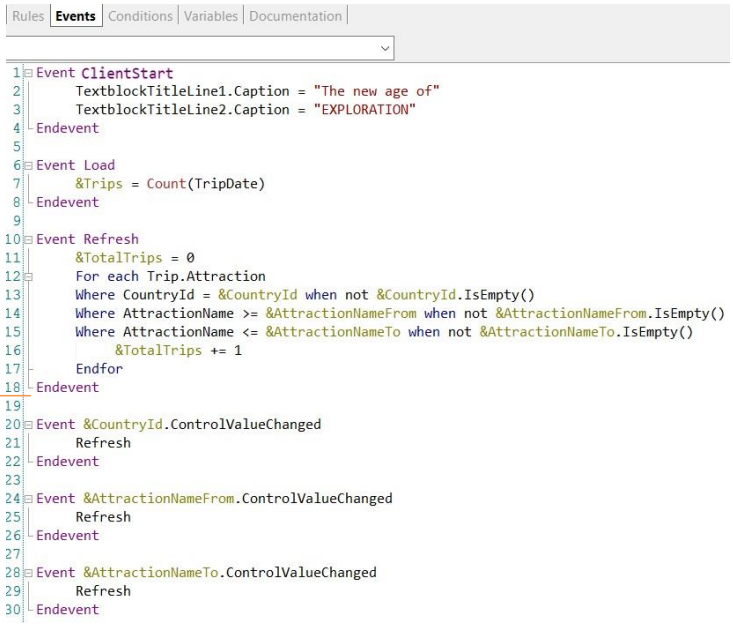
En esta arquitectura, como el objetivo es que la página se cargue la menor cantidad de veces posible, se prioriza el caché de datos, es decir que se trata siempre de recuperar la información previamente almacenada. Es decir que dependiendo de la configuración de almacenamiento en caché, se decide si 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 basada en un SDT, como es nuestro caso) y se actualiza la parte fija y la parte variable del panel, como esperábamos.

Uso del comando Refresh en eventos del cliente (cont.)







Ejecutemos el panel para probar todo esto que vimos.

Vemos que con los filtros vacíos se muestran todas las atracciones y para cada atracción se muestra correctamente el total de viajes, calculado en el evento Load, que actualizó la información del grid.

Vemos también que el total de global de viajes que está en la parte fija se muestra correctamente, dado que se calculó en el evento Refresh y éste se ejecutó antes de dibujarse la parte fija.

Si ahora elegimos a las atracciones del país Francia y elegimos ver las atracciones que comienzan con la letra A hasta la letra N, vemos que el valor de los filtros presentes en las conditions del grid funciona correctamente y que los comando Refresh invocados provocaron que se dispararan los eventos Refresh y Load, por lo que la información se recuperó del servidor correctamente actualizada y solamente vemos a las atracciones de Francia con el nombre en el rango elegido.

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

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications