PANTALLAS INTERACTIVAS

Más sobre Web Panels

¿CON TABLA BASE?
¿SIN TABLA BASE?

(REPASO Y MÁS CONOCIMIENTOS..)

Si bien los ejemplos que hemos visto hasta el momento de web panels "SIN TABLA BASE" son de casos en los que necesitamos solicitar datos al usuario, también existen casos de uso de web panels "sin tabla base" con variables en el grid (pudiendo haber también variables en la parte plana del form) y no necesariamente ser el objetivo del web panel, ofrecer un ingreso de datos.

Dado que el evento Load en los web panels "sin tabla base" se ejecuta solamente 1 vez, es posible incluir dentro de dicho evento una programación explícita con For each e ir asignándole valores a las variables que se encuentran en el grid (por ejemplo con los valores de los atributos navegados, fórmulas, etc.) y una vez inicializadas todas las variables necesarias como para agregar una línea, contamos con el comando Load, el cual solamente aplica para ser usado dentro del evento Load, para cargar una línea en el grid.

Ahondaremos acerca de este otro uso posible de web panels "sin tabla base" más adelante.

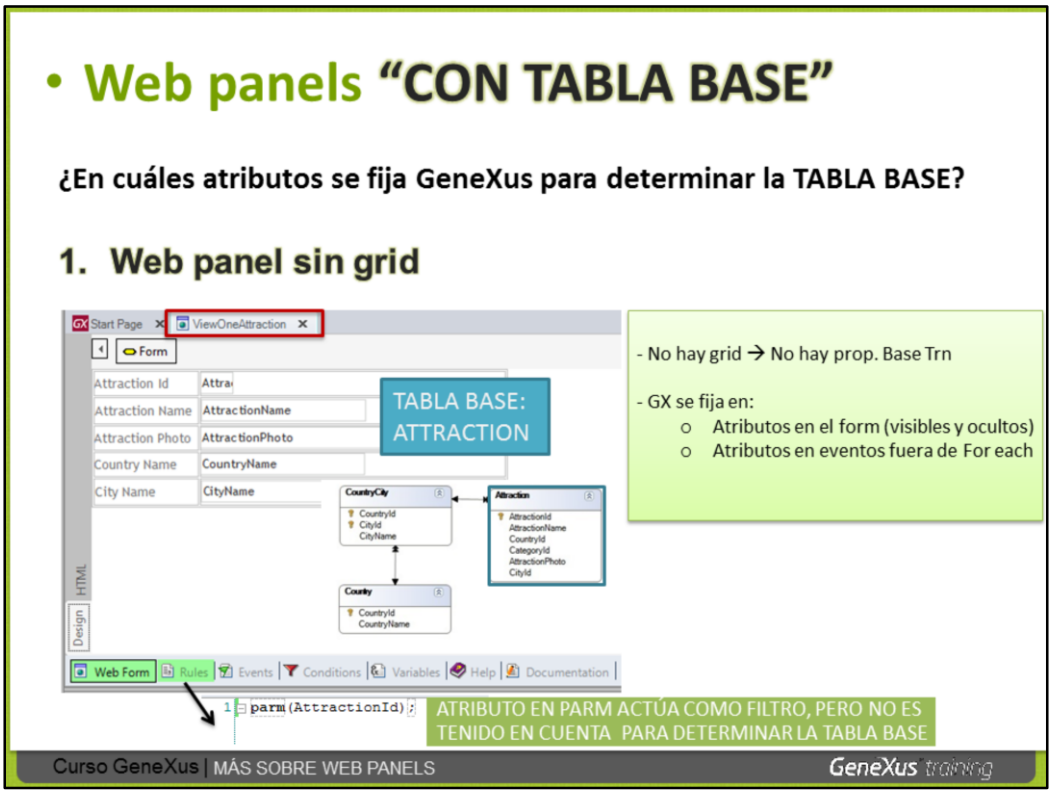
Como en todo momento hemos mencionado, el web panel es un objeto muy flexible, que permite múltiples usos, así que inclusive se presta para mucho más.

• Web panels “CON TABLA BASE”

¿En cuáles atributos se fija GeneXus para determinar la TABLA BASE?

Dividamos en casos:

1. Web panel sin grid
2. Web panel con 1 grid
3. Web panel con más de 1 grid



En este web panel no se ha incluído 1 grid, sino que se han insertado atributos de 3 tablas físicas en el form (quedaron insertados automáticamente en una tabla para que aparezcan alineados en el form).

El web panel tiene solamente los atributos mostrados en el form y la regla parm que se muestra definida.

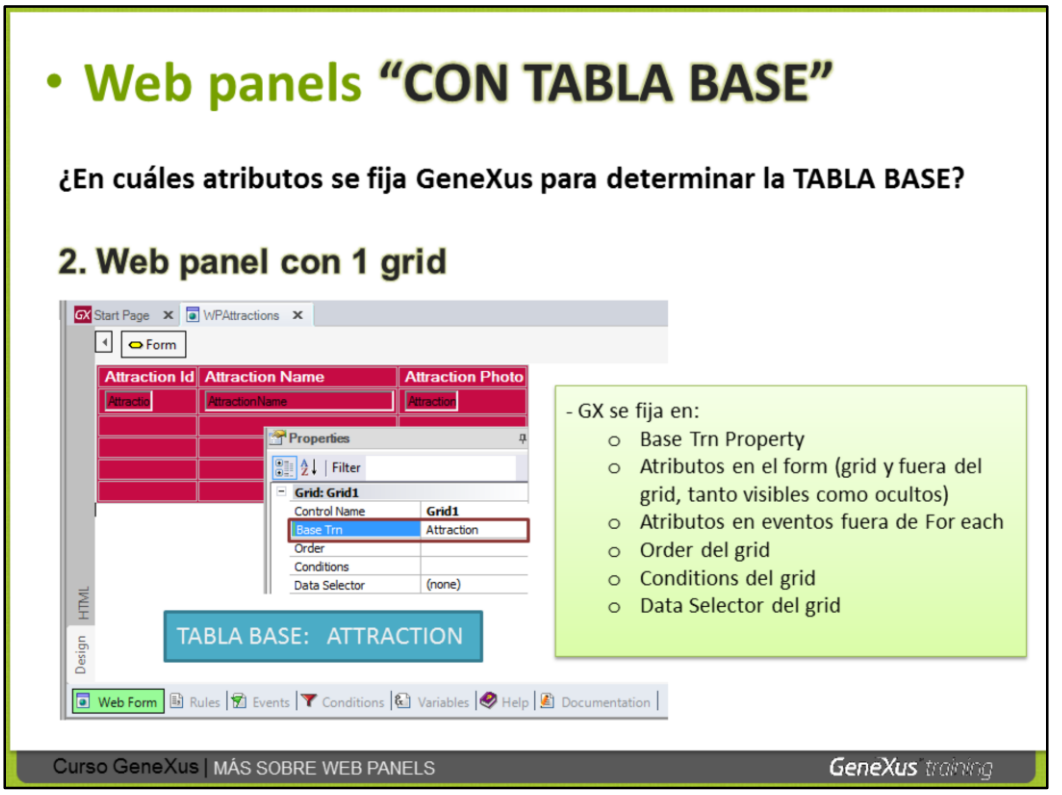
No hay posibilidad de indicar una base transaction, porque no hay grid. Pero el analista GeneXus insertó los atributos, sabiendo que GeneXus analizaría en cuáles tablas se encuentran los mismos y teniendo en cuenta las relaciones, **determinaría una tabla base y accedería a su tabla extendida** para obtener los valores de todos los atributos requeridos. En este ejemplo entonces, es claro que GeneXus determina navegar la tabla física ATTRACTION porque siendo ATTRACTION la tabla base, el resto de las tablas se encuentran en su tabla extendida y hay alcance a todos los datos solicitados.

¿Pero de todos los registros de ATTRACTION, cuál se mostrará?

Si no estuviera la regla Parm definida, se navegaría toda la tabla física y finalmente quedarían a la vista los datos de la última atracción registrada.

Pero la regla parm recibe el **atributo** identificador AttractionId, por lo tanto, entendemos que este web panel será invocado desde algún otro objeto, enviándole por parámetro, cierto identificador de atracción y dicho valor se utilizará como filtro por igualdad, así que se consultará 1 registo específico en la tabla física ATTRACTION y se mostrarán sus datos y los relacionados pertenecientes a su tabla extendida.

O sea que el web panel tiene un **For each implícito, o en otras palabras, una navegación automática determinada por GeneXus**. Por eso, decimos que este web panel **tiene TABLA BASE**.



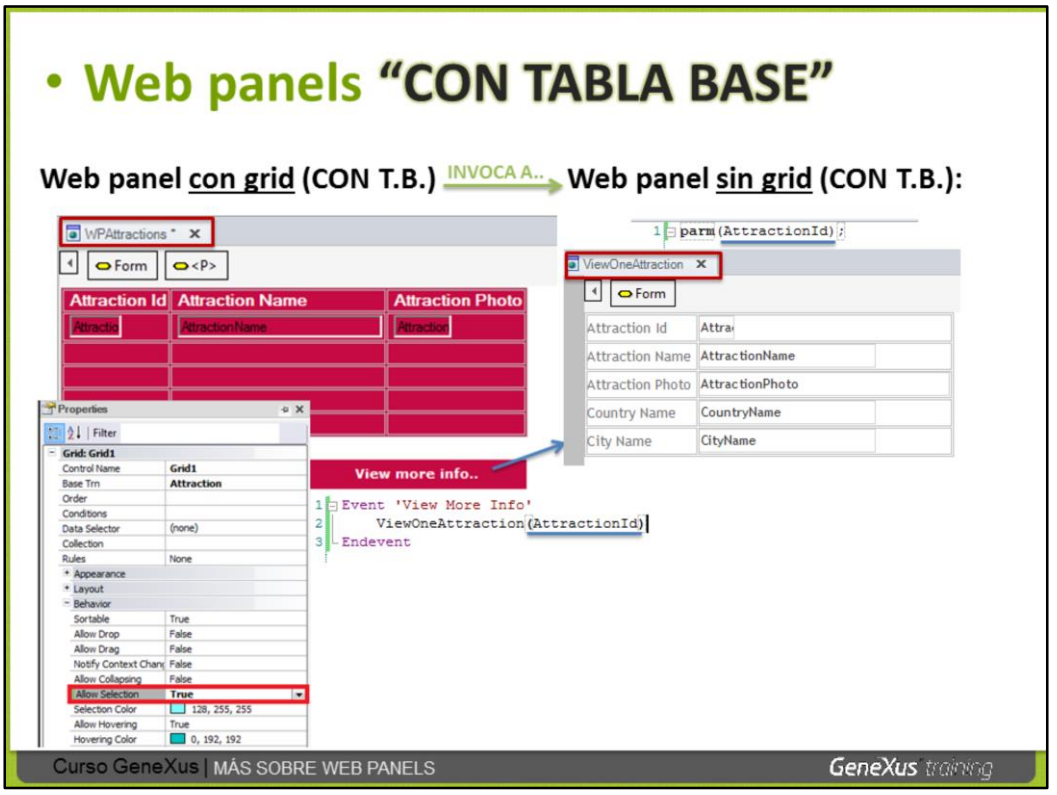
Este web panel contiene solamente en su form, el grid que se muestra, con atributos de la tabla física ATTRACTION y la propiedad Base Trn del grid = Attraction (nada más se ha definido en este objeto).

GeneXus tiene la información clara de cuál será la tabla base a navegar: ATTRACTION.

Así que en tiempo de ejecución se navegarán todas las atracciones y se mostrarán en el grid del web panel, todas las atracciones (ya que no hay filtros definidos en las conditions) y saldrán en el grid las mismas ordenadas por AttractionId (ya que no se especificó ningún order tampoco).

Es lo mismo decir que **la tabla base del web panel es ATTRACTION**, o que **la tabla base del grid es ATTRACTION**, porque hay 1 solo grid, por lo tanto en definitiva ATTRACTION es la tabla base del grid y del web panel. Esto significa que hay un for each implícito que no hemos definido, pero que GeneXus determinó automáticamente para el web panel.

Data Selector es una propiedad del grid, que puede configurarse o no. Veremos oportunamente de qué se trata.

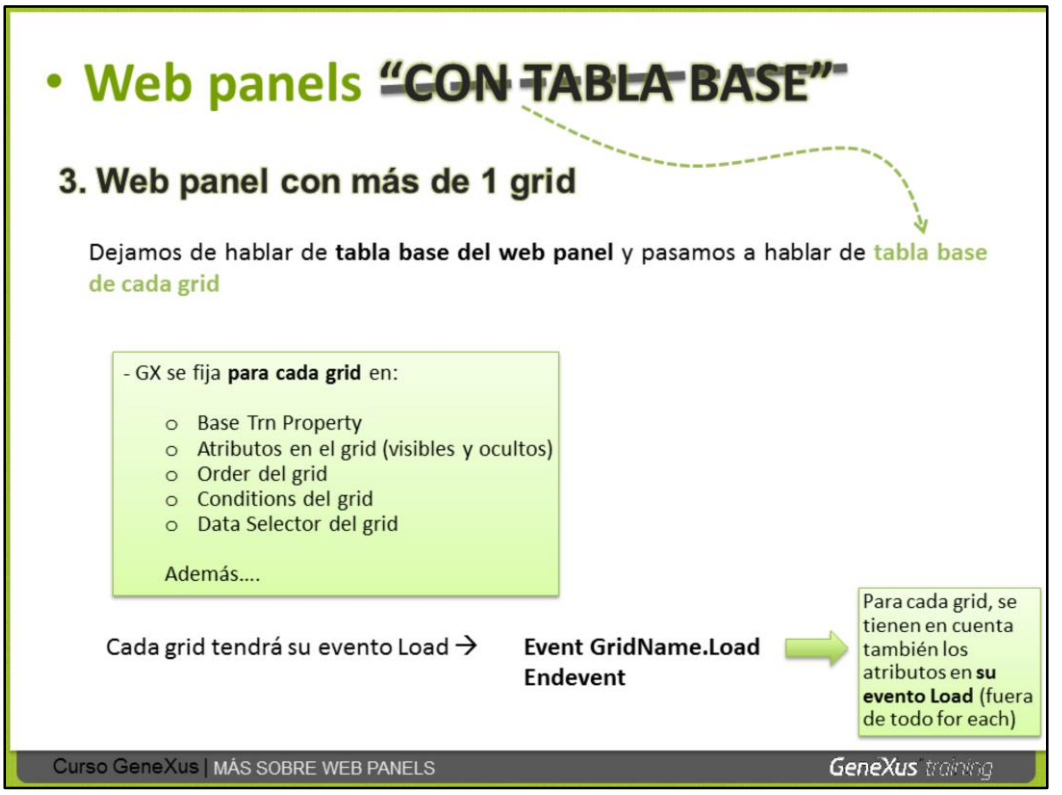


Aquí estamos proponiendo que uno de los web panels que hemos definido invoque al otro, para hacer uso de ambos, integrar lo que hemos definido y aprender más.

Sabemos que el web panel "WPAttractions" (el que contiene el grid), tiene tabla base: ATTRACTIONS y carga en el grid, todas las atracciones registradas.

Ahora le hemos hecho 2 modificaciones a este web panel. En primer lugar hemos configurado la propiedad del grid Allow Selection = True, para que el usuario en tiempo de ejecución, al desplazar el cursor por el grid, vea una línea de color que se desplaza también y pueda hacer click en una de las líneas; al hacerlo, la misma quedará seleccionada y pintada.

Lo segundo que hemos agregado al web panel, es un botón, para que justamente cuando el usuario haya seleccionado una línea, lo pueda presionar y se efectúe la invocación al web panel "ViewOneAttraction" pasándole por parámetro el identificador de la atracción de la línea seleccionada, y dicho valor sea recibido en el objeto invocado, actuando como filtro por igualdad para mostrar los datos de esa atracción.



Cuando hay más de 1 grid en un web panel, **la tabla base de cada grid** se determina **considerando exclusivamente a los atributos mencionados en la diapositiva**.

Cada grid tendrá su evento Load y la sintaxis se muestra en la diapositiva.
No puede usarse el evento Load genérico, dado que no se sabría de cuál grid se trata.

A diferencia del caso de un único grid, los atributos que estén fuera de for eachs en cualquier evento que no sea "su" Load, no participarán de la determinación de las tablas base. Pero deberán cumplir que pertenezcan a la tabla extendida de alguno de los grids. De no ser así, GeneXus lo advertirá en el listado de navegación.

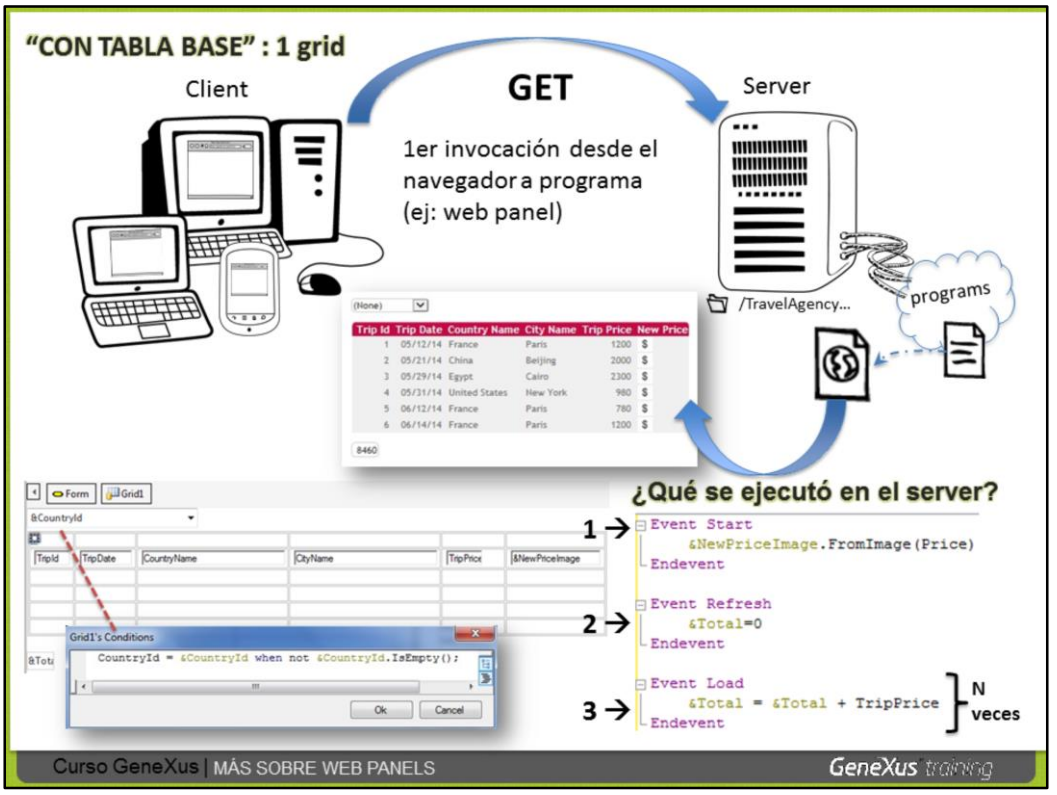
De existir atributos en la parte fija, la tabla base del primer grid en el form se determina tomando en cuenta los atributos sueltos, y los demás grids sin tomarlos en cuenta. Si desea ver más sobre este caso especial acceda a nuestro wiki: http://wiki.gxtechnical.com/commwiki/servlet/hwikibypageid?6105

EVENTOS
cuáles?
cuándo?
dónde?
orden?

Ya hemos visto y programado diferentes eventos en Web panels (por ejemplo el click, asociado a controles incluídos en el form, pero según el control, se pueden programar también el doble click, botón derecho, etc.).

También hemos presentado y usado los eventos Start y Load asociados al objeto web panel. Vimos que el evento Enter que queda asociado por defecto a todo botón que se inserta en el form y hemos definido también para botones, eventos de usuario explícitamente. También los web panels cuentan con el evento Refresh.

Veremos a continuación más detalladamente los eventos de los web panels, dónde se ejecuta cada uno (si en el servidor donde está instalada la aplicación o cliente) y en qué orden.



En toda aplicación web, tendremos una máquina –PC, notebook, o dispositivo inteligente– con conexión a internet, con la cual el usuario accederá a la aplicación a través de un navegador; y por otro lado el servidor, que será donde se encuentran todos los programas de la aplicación, generados por GeneXus. Entre ellos, los correspondientes a los web panels (por ejemplo, el programa generado para el web panel mostrado en la diapositiva).

Este web panel permite seleccionar un país (de todos los almacendos) y el identificador del país elegido quedará en la variable &CountryId presentada con el control Dynamic Combo Box (habiendo configurado las propiedades asociadas a dicho control).

Hay 1 condition definida, como se muestra en la diapositiva, para que en el grid se carguen todas las excursiones que hay registradas para el país elegido en la variable &CountryId.

Dentro del grid en la última columna hay una variable de tipo imagen, a la cual le asignaremos una imagen con el signo de pesos ($), en el evento start.

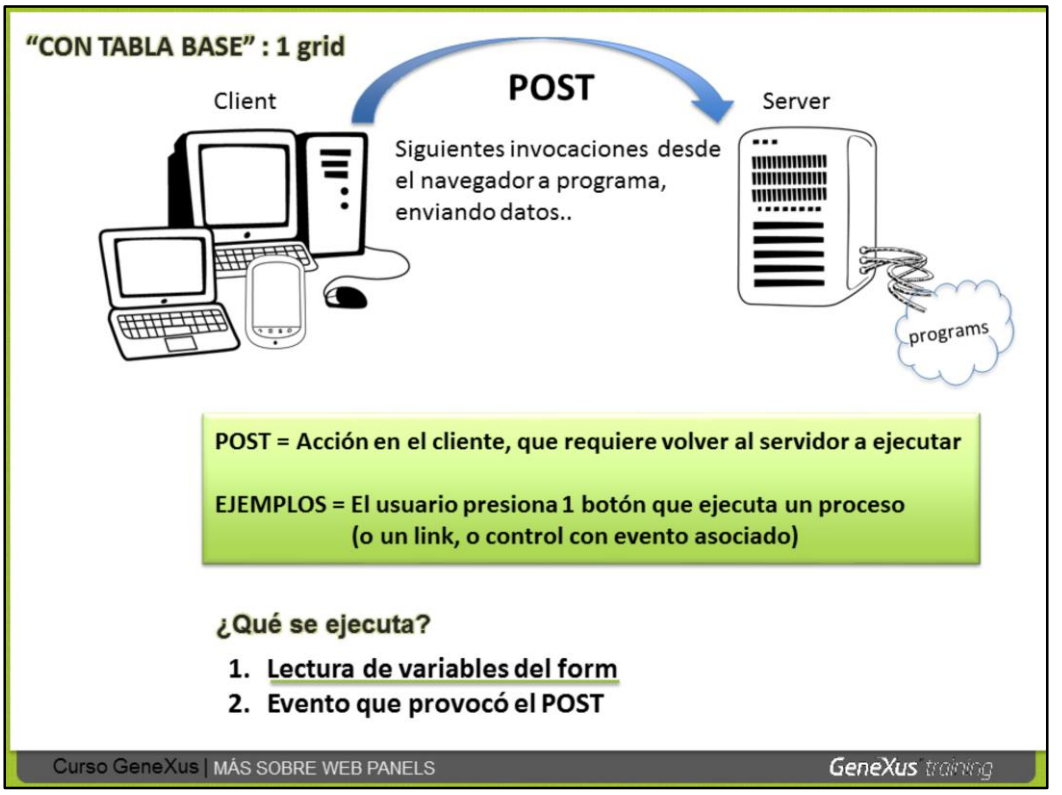
Debajo del grid, se desea ver la suma total de los precios de las excursiones mostradas (en la variable &total).

Bien. ¿Qué sucede cuando invocamos al web panel desde el navegador **por primera vez**? Lo que se conoce como hacer un **GET**.

El cliente le pide al servidor que ejecute el programa asociado al web panel y le devuelva como resultado un archivo html que le indique al navegador cómo dibujar la pantalla (con qué datos, con qué formato, etc.).

¿Pero qué ejecuta el programa en el servidor para armar ese archivo html?

Primero el evento Start. Y por lo que hemos definido dentro del mismo, se carga la imagen del símbolo de dinero en la variable &NewPriceImage.

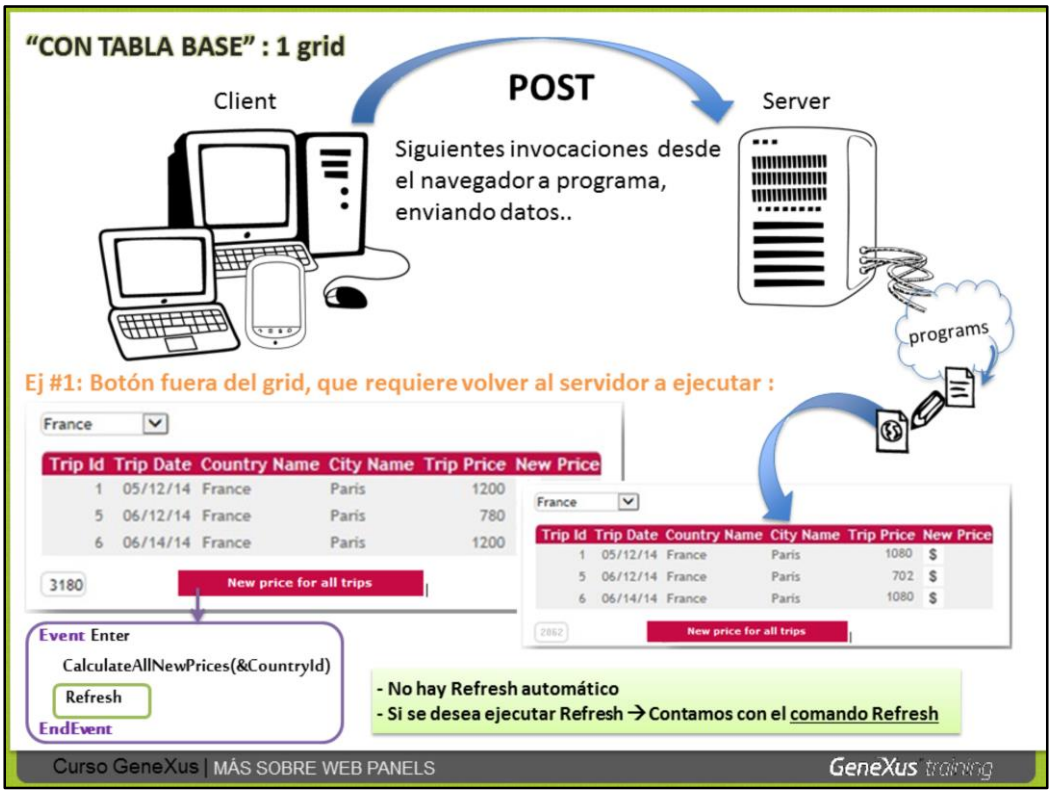


En forma general, un **POST** se produce cada vez que se efectúa alguna acción en el cliente, que requiere volver al servidor a ejecutar.

Acciones de este tipo pueden ser presionar la tecla Enter o algún botón o control asociado a un evento.

Cuando se ejecuta un POST, sucede lo siguiente:
- Se leen la variables en pantalla
- Se ejecuta el evento de usuario que provocó el Post.

Tanto las variables del form como los parámetros del web panel están dentro del alcance del evento.



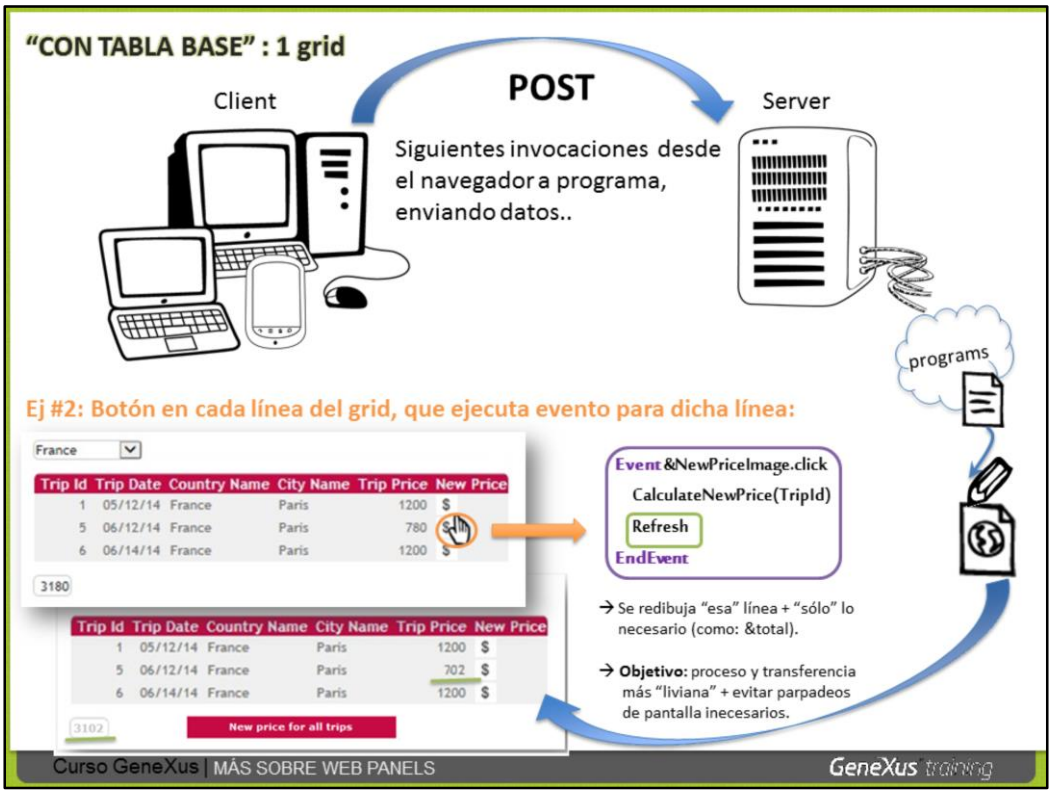
En este ejemplo hay un botón debajo del grid, que ofrece aumentar los precios de todas las excursiones mostradas en el grid (o sea, las del país elegido en el combo). En el evento asociado al botón, se invoca a un procedimiento y se le envía por parámetro el valor de la variable &CountryId. No es importante conocer el criterio de obtención de los nuevos precios (tal vez sea teniendo en cuenta la fecha del día, se verifique en una tabla si hay alguna promoción que se ofrezca puntualmente en el día hoy.. u otras verificaciones). El hecho es que cuando el usuario presione el botón, se invocará al procedimiento, que recorrerá con un For each todas las excursiones del país recibido por parámetro y actualizará el precio de cada una de esas excursiones en la tabla TRIP, en base al criterio que haya pedido la agencia de viajes.

Algo importante de conocer y considerar, es que los eventos (tanto eventos de usuario como eventos asociados a controles) **no ejecutan Refresh automáticamente** (ya que el programador debe determinar si lo requiere efectuar o no; tal vez quiera ejecutar un proceso en el servidor y no refrescar a continuación el form).

De necesitar refrescar el form, GeneXus ofrece el comando: Refresh. En este ejemplo, el procedimiento actualizará la base de datos (los precios de las excursiones) y el desarrollador necesita incluir el comando Refresh inmediatamente a continuación de la invocacación al procedimiento.

El comando Refresh ejecutará el evento Refresh, es decir que el código que dicho evento contenga definido, se ejecutará y a continuación se ejecutará el evento Load tantas veces como registros cumplan con las conditions… y se cargarán en el archivo que se devolverá al cliente para que el navegador dibuje la página "refrescada".

Resumiendo: cuando el usuario presiona el botón, se leen las variables del form (en este caso &CountryId) y posteriormente se ejecuta el evento asociado al botón presionado. El evento tiene en su código 2 líneas: 1) ejecutar el procedimiento y 2) ejecutar el comando refesh.



Ahora estamos presentando otro ejemplo de POST.

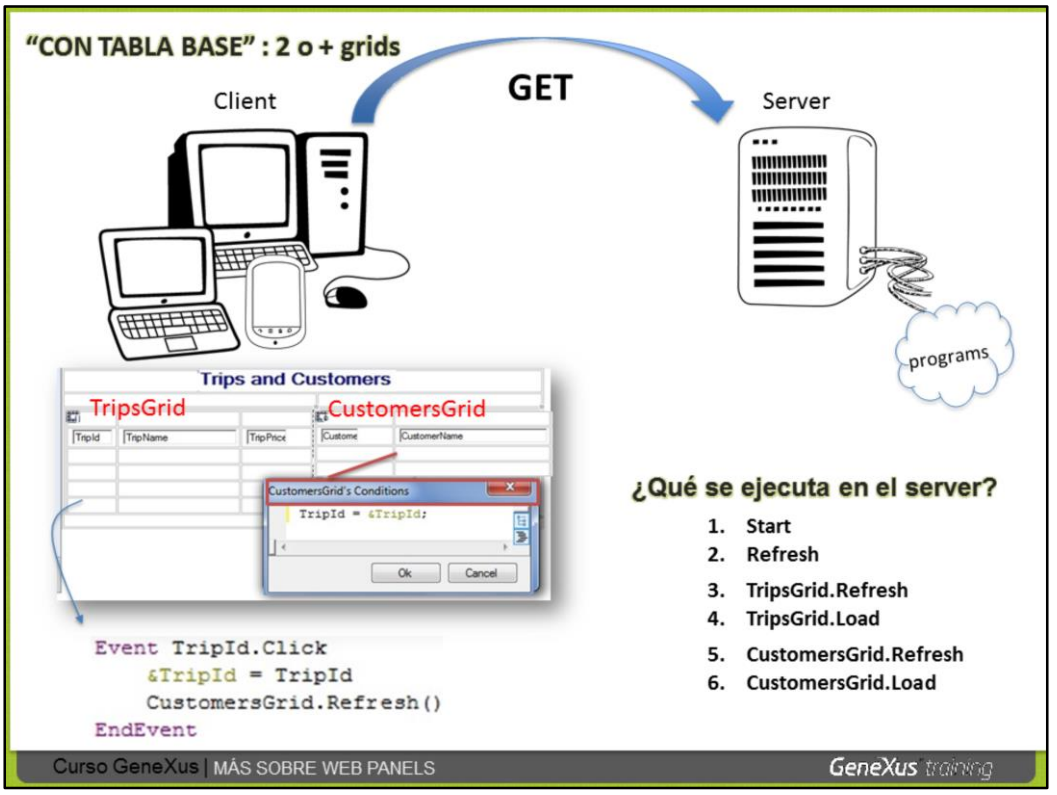
Aquí hay 1 variable dentro del grid, en la última columna (que tiene cargada la imagen del signo de $) y tiene programado el evento click.

Cuando el usuario presione para una línea, la imagen, se ejecutará el evento asociado y como ya hemos mencionado, sucederá lo siguiente:

- Se leerán las variables del form (&CountryId)
- Se ejecutará el evento que provocó el POST

El evento invoca a un procedimiento y le envía el TripId de la línea, para que que procese el cálculo para esa excursión y grabe su nuevo precio. A continuación se solicitó un Refresh.

El comportamiento en este caso, será que la línea implicada del grid, se actualizará, **mientras que las demás líneas permanecerán iguales sin ser recargadas**.



Veamos ahora cómo es el disparo de eventos cuando hay más de 1 grid en el form.

El web panel de arriba tiene 2 grids. Como ya hemos mencionado, cada grid tendrá su tabla base. El grid llamado TripsGrid tiene la tabla base: TRIPS y el grid de nombre CustomersGrid tiene la tabla base CUSTOMERS (por los atributos que incluye cada uno + la property Trn Base de cada uno, etc.).

El objetivo es que cuando el usuario haga click sobre el identificador de una excursión (Trip) vea en el otro grid, los clientes registrados para realizar la misma.

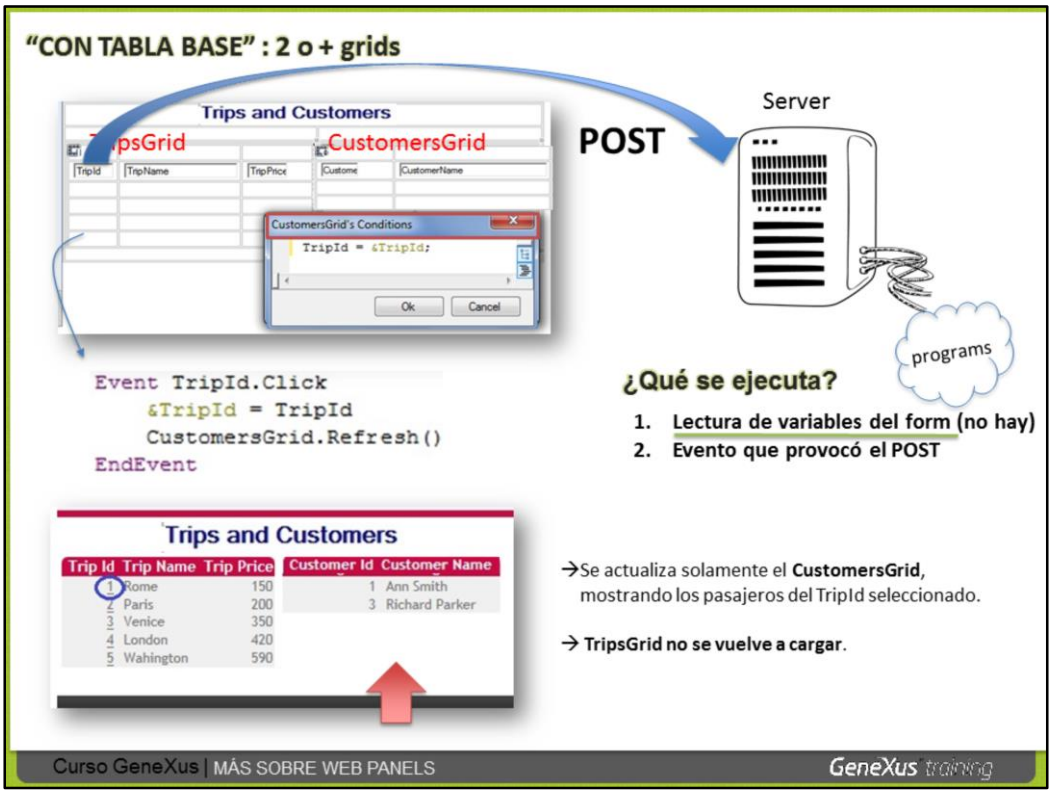
El GET es la 1er ejecución de 1 web panel. ¿Qué sucede en el GET en este caso?

Al ejecutar este web panel desde el cliente se ejecutarán, en orden:

1. El evento Start
2. El evento Refresh, general, que ejecuta su código (de haberse programado) y llama al refresh y load de cada grid. Así, llama a:
3. Evento Refresh del primer grid del form, y éste al
4. Load de ese grid, una vez por cada registro . Y luego…
5. Evento Refresh del segundo grid del form, y éste al
6. Evento Load de ese grid, una vez por cada registro que cumpla las conditions.

Como al ejecutarse el GET, aún nadie hizo click y la variable &TripId está vacía, para el segundo grid no se cargará ninguna línea.

Veamos ahora qué sucede cuando el usuario hace click sobre el identificador de 1 excursión (Trip), y por lo tanto se realiza un POST.



Cuando el usuario presiona click sobre 1 TripId del 1er grid, se realiza la lectura de variables en pantalla (en este ejemplo no hay ninguna), y luego se ejecuta el evento click que causa el POST.

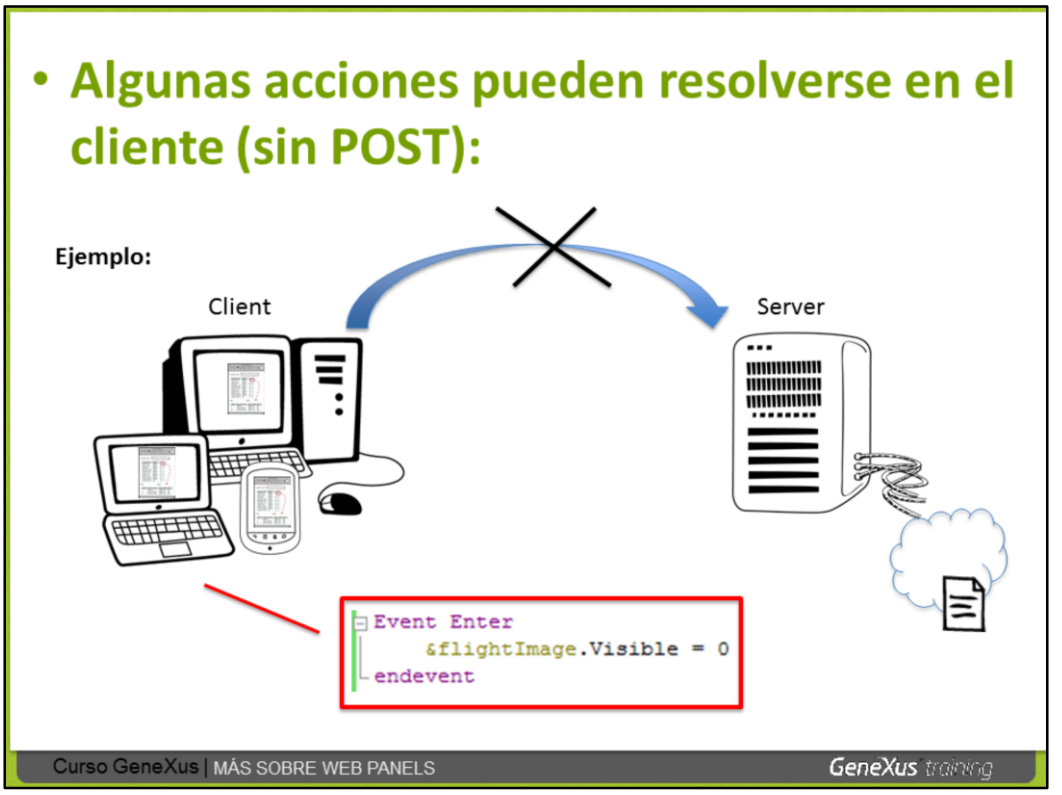
En el evento TripId.click, se guarda en la variable &TripId el identificador de la excursión seleccionada, y se provoca luego explícitamente el Refresh del 2do grid (ejecutando el comando: CustomersGrid.Refresh()).

Observemos que el CustomersGrid tiene definida una condición que indica que se filtre por la excursión asignada a la variable &TripId.

Así que cuando se ejecute lel comando: CustomersGrid.Refresh(), se aplicará el filtro definido en las conditions de CustomersGrid y se disparará a continuación el evento Load del CustomersGrid N veces filtrando los clientes registrados para la excursión clickeada en TripsGrid.

CustomersGrid se actualiza, mientras que el TripsGrid no se vuelve a cargar.

En cuanto al evento Start, solamente se ejecutó una vez, al efectuarse el GET del web panel.

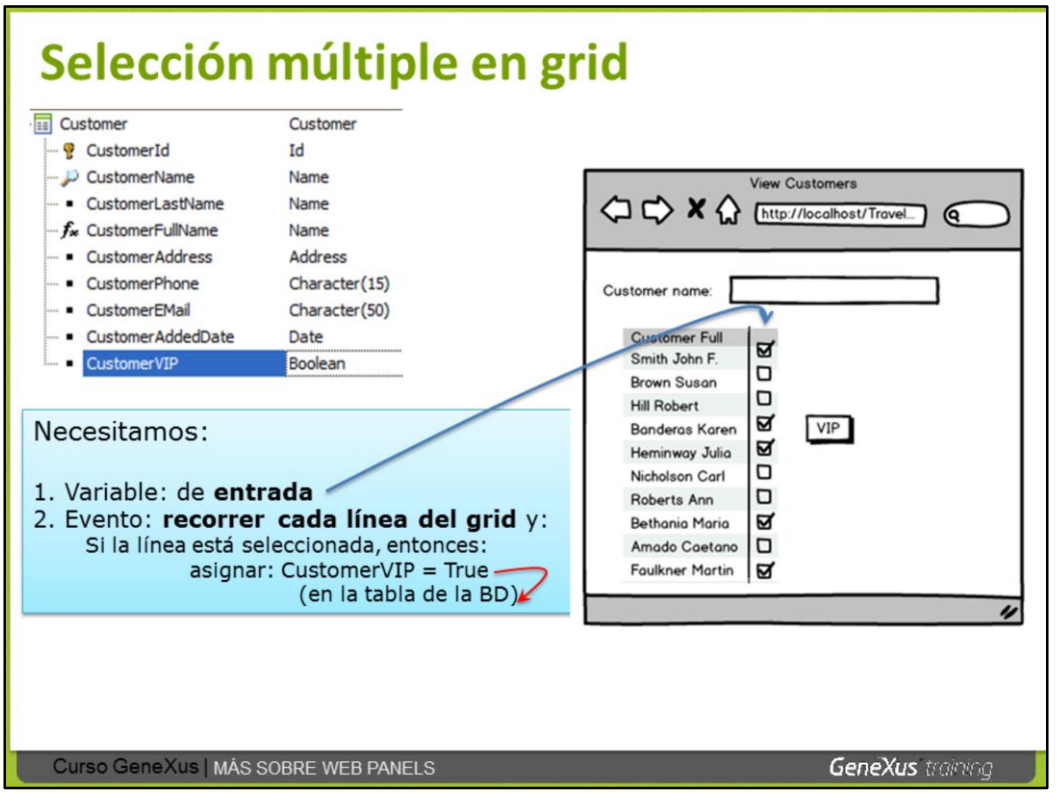


Para terminar, mencionemos que no toda acción efectuada por el usuario y asociada a un evento, producirá un POST al servidor. Algunas se pueden resolver en el propio cliente.

Por ejemplo, si en un evento enter, de usuario o asociado a un control, se pone invisible un control, eso se puede resuelve en el propio cliente.

Con esto hemos visto lo más importante acerca de los eventos que se ejecutan en los web panels.

GRID
selección múltiple
cómo recorrerlo



Veamos un ejemplo que propone incluir una variable en el grid, presentada como checkbox para que el usuario pueda marcar líneas con el objetivo de poder hacer una selección múltiple de líneas y luego ejecutar una acción. El atributo CustomerVIP que hemos agregado en la transacción Customer, permite distinguir a los clientes preferenciales.

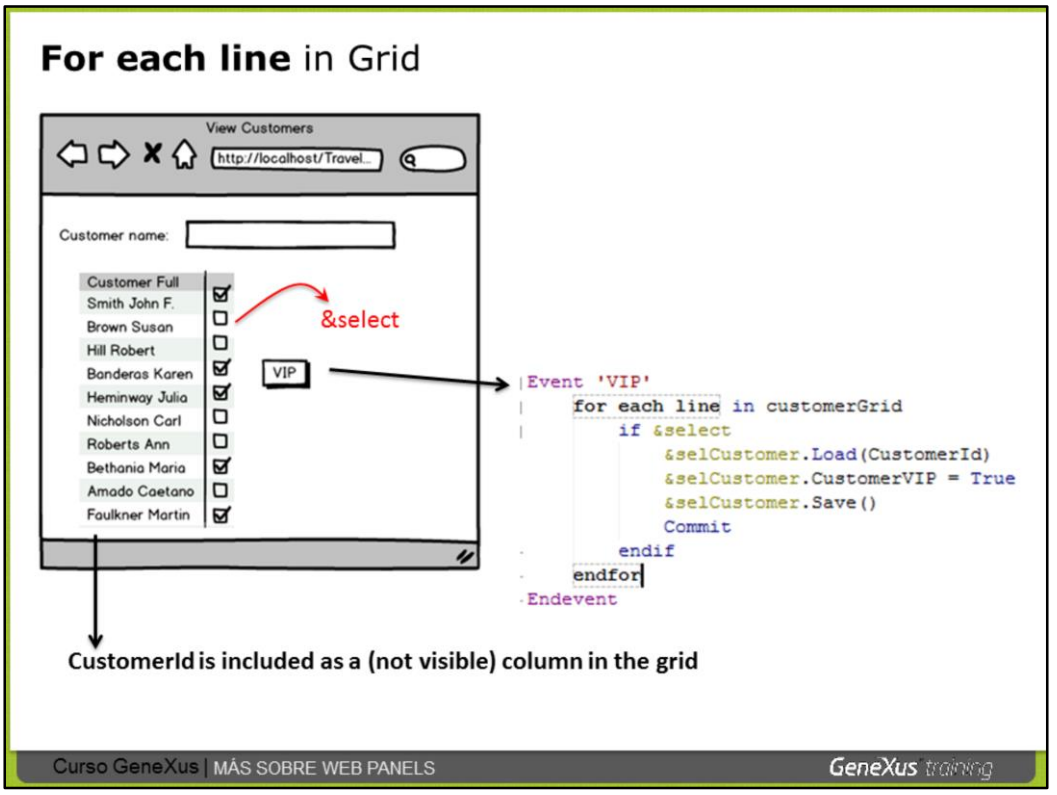
Necesitamos que el web panel que muestra los clientes, permita seleccionar varios clientes y presionando el botón VIP, asignarles el status VIP.

Para ello, agregamos una variable booleana al grid de vuelos, &select, que permitirá que el usuario seleccione la línea (por defecto toda variable booleana será mostrada como un checkbox) y agregamos también el botón en el form con un evento de usuario asociado que creamos especialmente, con el nombre VIP.

Necesitamos dos cosas:

1. que la variable que agreguemos al grid como columna no sea read only (comportamiento por defecto de las variables en grids de web panels), para que el usuario pueda decidir si marcarla o no, para cada línea.
2. que al presionar el botón VIP, podamos programar una recorrida de las líneas del grid y para cada línea que tenga la variable marcada (esto es, con valor True), podamos asignar True al atributo CustomerVIP correspondiente a ese cliente y grabar ese cambio físicamente.

El punto 2 sabemos resolverlo, usando una variable del tipo business component Customer (&selCustomer). Pero vamos a ver cómo recorrer las líneas del grid…



El comando **for each line** permite recorrer las líneas de un grid. La iteración se finaliza con endfor.

Es importante tener clara la diferencia de que mientras el **for each** recorre registros de una tabla (**base**) de la base de datos, el **for each line** recorre las líneas de un grid.

Si el web panel tiene solamente 1 grid, se puede escribir solamente: **For each line**. En cambio si contiene más de 1 grid, hay que especifar cuál grid se quiere recorrer, por lo tanto hay que especificar: **"for each line in" + el nombre del grid.**

Por defecto las variables dentro de grids de web panels, son read-only. Pasan de ser Read only, a ser de entrada, en los siguientes casos:

1) Cuando en un evento del web panel se utiliza el comando específico para recorrer las líneas de un grid, es decir: **for each line in GridName**.
2) Cuando se programa el evento click asociado a 1 imagen que se encuentra dentro del grid.

En el ejemplo presentado en la diapositiva, el código asociado al evento 'VIP' recorre las líneas del grid y para cada línea, primero evalúa si la línea está seleccionada; en caso afirmativo, codificamos la asignación del cliente como VIP, utilizando el concepto de Business Component. Es decir:

- La transacción Customer está configurada como Business Component y el web panel tiene definida la variable &selCutomer basada en el tipo de datos Business Component: Customer.
- Se aplica el método Load a la variable &selCutomer, pasándole por parámetro el valor del atributo CustomerId disponible (oculto) en el grid para cargar en memoria el registro de cliente

- Seguidamente se asigna que el cliente es VIP, se graba físicamente y ejecuta commit.

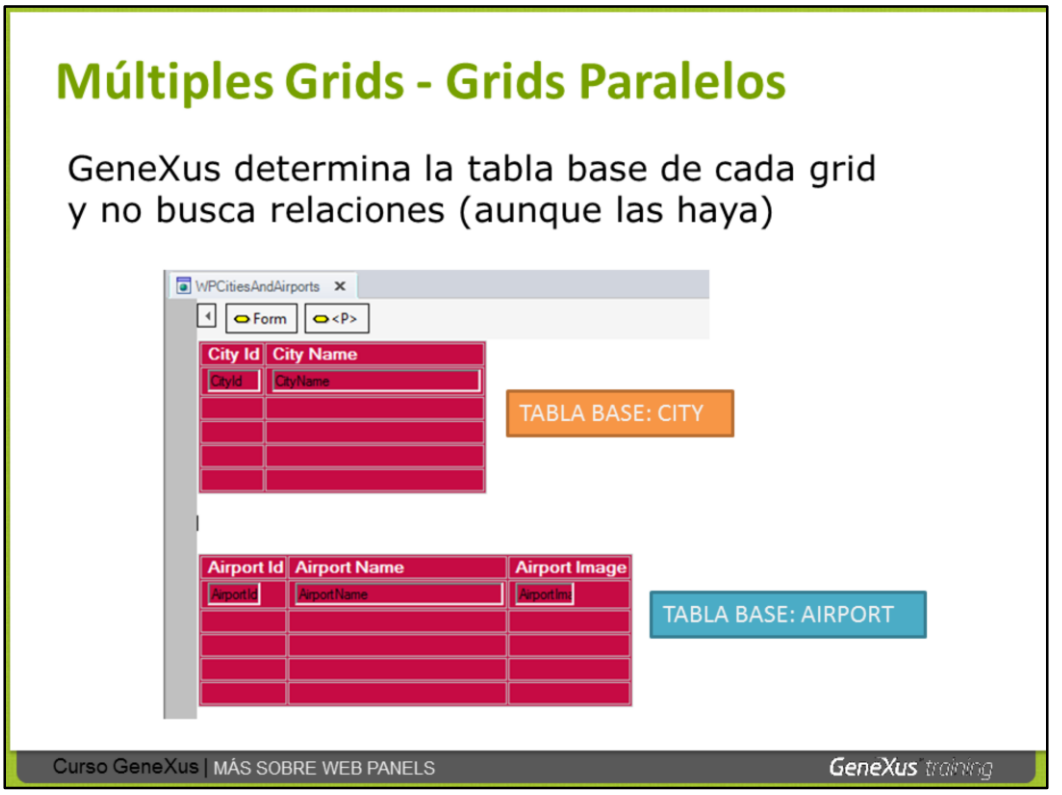
MÚLTIPLES GRIDS

Existe un tipo de grid diferente del standard, llamado **grid free style**, que permite diseñar 1 línea con un diseño más libre y vistoso que no contiene columnas estructuradas y esa línea se repetiría mostrando ese diseño para cada registro navegado

1 grid free-style puede contener dentro otro grid free-style dento o standard.

En el ejemplo mostrado en la diapositiva hemos incluído en el form del web panel 1 **grid free style**, dentro colocamos el atributo CustomerName, y debajo 1 grid standard con atributos de atracciones.

Para el grid anidado no se establecemos conditions a nivel del grid. ¿Por qué? Porque debido a que el grid se encuentra dentro de otro, GeneXus anida las navegaciones, como si se tratase de un par de for eachs anidados. El filtro automático es por: CountryId.



En el web panel de la diapositiva de arriba, hemos incluído 2 grids standard.

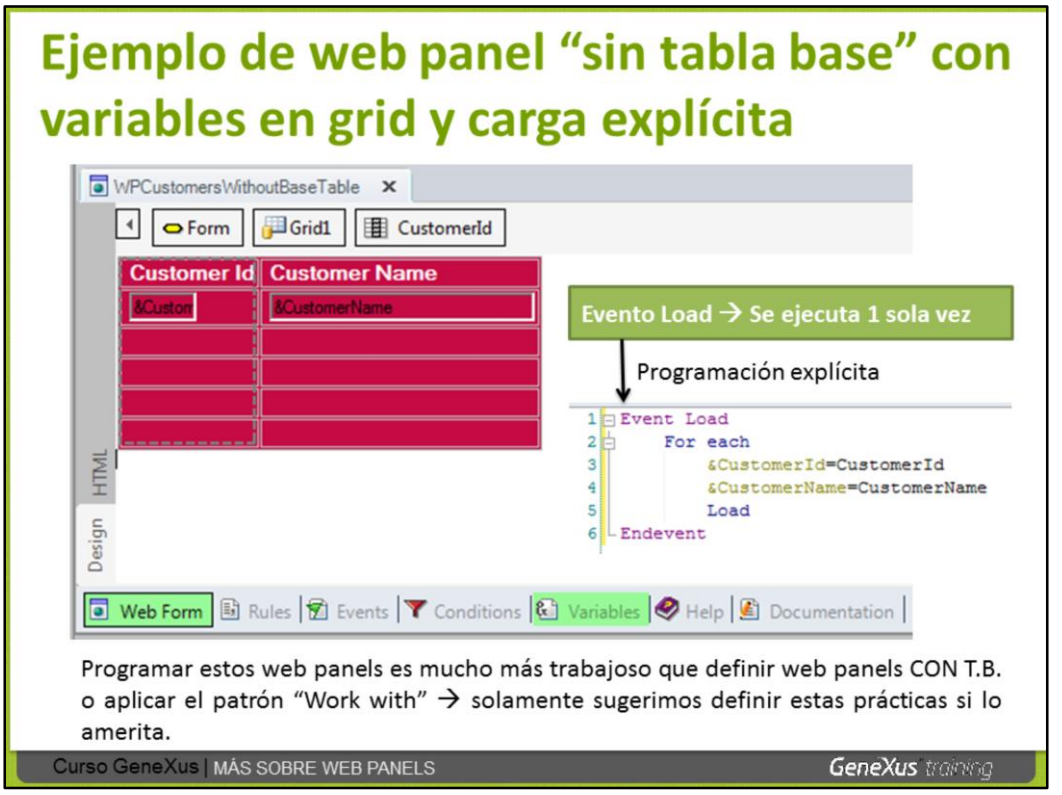
Para el 1er grid GeneXus determinará que su tabla base es: CITY y para el grid de abajo determinará que su tabla base es AIRPORT.

Así, el web panel mostrará en el 1er grid, todas las ciudades registradas y en el 2do grid, todos los aeropuertos registrados. Pero no buscará ni establecerá relaciones.

Es posible programar la posibilidad de que el usuario pueda seleccionar 1 ciudad, presionar 1 botón y se carguen los aeropuertos de dicha ciudad en el 2do grid (como ya hemos visto un ejemplo análogo).

Y sino, se podría implementar una solución como la anterior con grids anidados y las navegaciones son anidadas y con filtros implícitos.

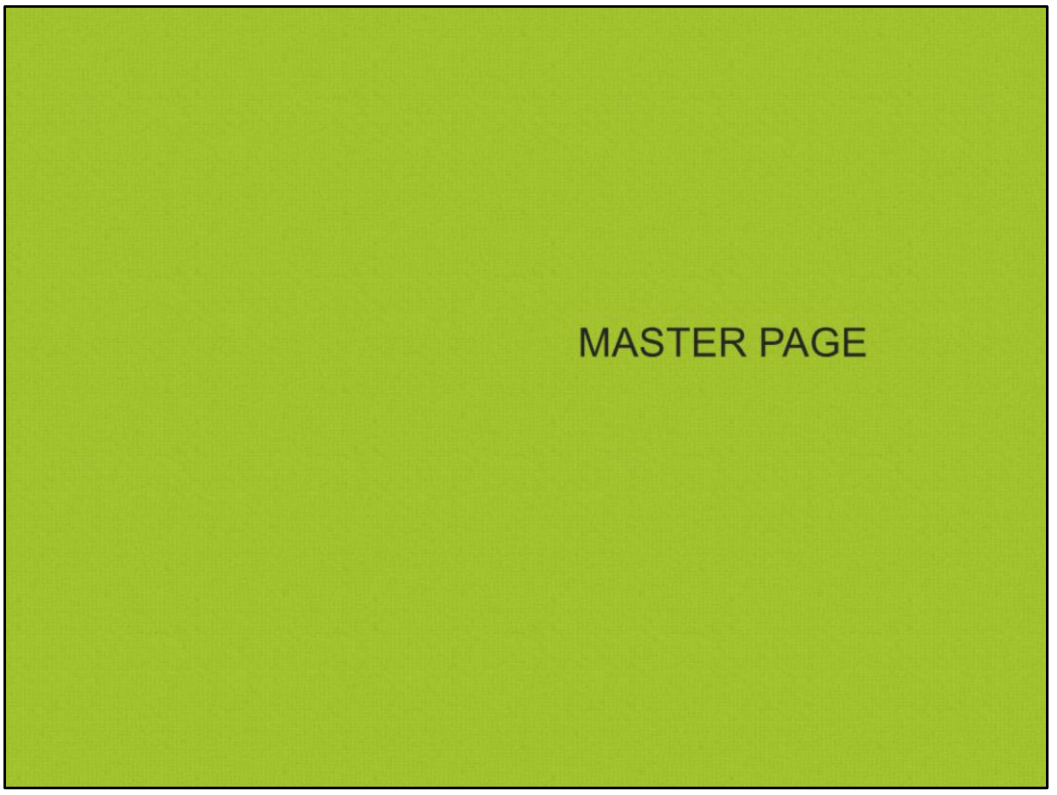
MAS SOBRE WEB PANELS
“SIN TABLA BASE”

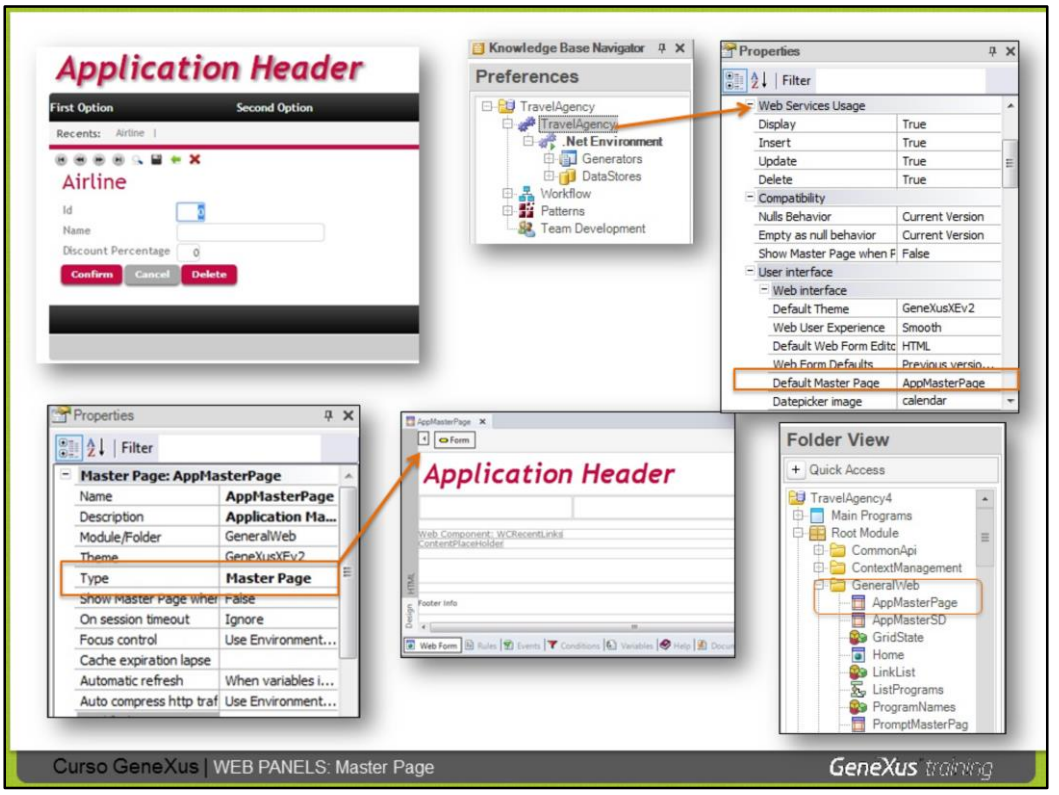


Ya hemos explicado que además de definir web panels “SIN TABLA BASE” con variables para solicitar datos al usuario, también existen casos de uso de web panels “sin tabla base” con variables en el grid (pudiendo haber también en variables en la parte plana del form).

Dado que el evento Load en los web panels “sin tabla base” se ejecuta solamente 1 vez, es posible incluir dentro de dicho evento una programación explícita con For each e ir asignándole valores a las variables que se encuentran en el grid (por ejemplo con los valores de los atributos navegados, fórmulas, etc.) y una vez inicializadas todas las variables necesarias como para agregar una línea, contamos con el comando Load, el cual solamente aplica para ser usado dentro del evento Load, para cargar una línea en el grid.

Programar este tipo de web panel es mucho más trabajoso que definir web panels “con tabla base” o aplicar el patrón “Work with” así que solamente sugerimos definir estas prácticas si lo amerita.


MASTER PAGE



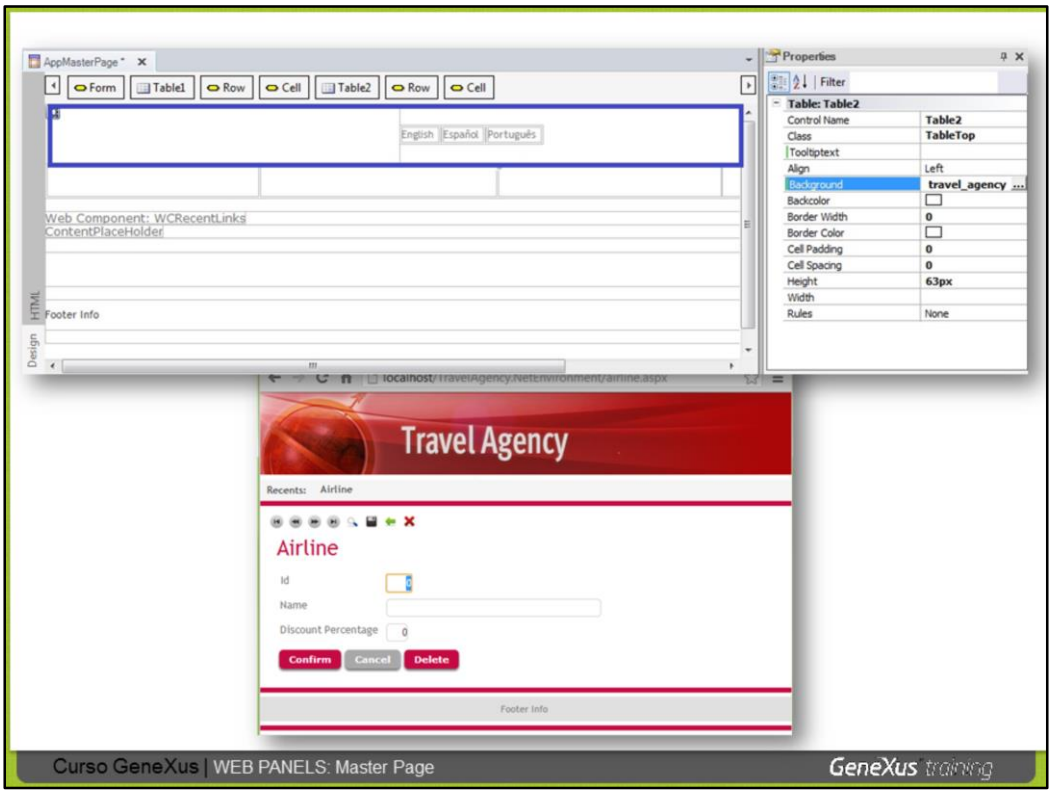
Los web panels pueden ser de tres tipos: Web Page, Component o Master Page, siendo el primero el predeterminado.

Al crear una KB con GeneXus, automáticamente podemos ver que se crea, dentro de la carpeta GeneralWeb, el objeto AppMasterPage, que es el que diseña el marco general y el comportamiento que en forma predeterminada tendrán todas las páginas web de nuestra aplicación. Así, por ejemplo, si ejecutamos cualquier transacción o web panel, veremos que su form está saliendo en un espacio central de una página que tiene, además, un encabezado, Application Header, y una franja final, Footer.

Este objeto, AppMaterPage, es un web panel cuyo tipo es Master Page. Por ese motivo, en su form se puede colocar un ContentPlaceholder, que es donde se cargarán todos los objetos web (transacciones y web panels) que tengan especificada a esa, como su página maestra.

Si bien se puede especificar la Master Page predeterminada (default) a nivel de la versión, también se puede especificar a nivel de objeto (por ejemplo, decirle a la transacción Airline, que su master page sea otra). Para ello, busque entre sus propiedades, la de nombre Master Page.

También podrá crear su propio web panel de tipo Master Page, y modificar el Default por el suyo. O simplemente personalizar el que ya viene con la kb, como mostramos en la página siguiente.



Aquí hemos modificado el objeto AppMasterPage, quitándole la imagen que traía, y agregando una nueva imagen a la KB, especificada como fondo de la tabla correspondiente.

GeneXus training

training.genexus.com