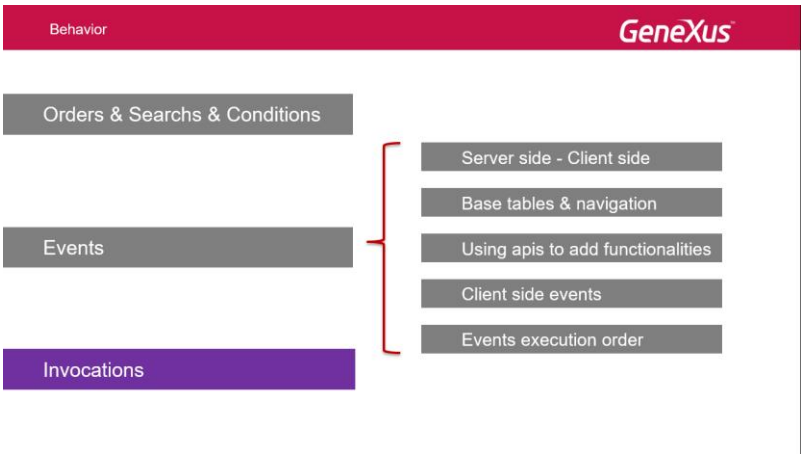
Invocations Between SD Objects



Ahora haremos un compendio de los objetos que se pueden invocar y cómo; y qué opciones de invocación tenemos, como por ejemplo lograr que el objeto llamado se abra con determinado efecto y se cierre con determinado efecto de transición. Vamos a ver los tipos de invocación que podemos conseguir, el tamaño de la pantalla del objeto invocado, etc.

Todo esto se conoce como **CallOptions**,y va a ser lo que vamos a trabajar a continuación.

Primero vamos a hacer un repaso de la sintaxis de las invocaciones a los Work With y a los paneles, que son los principales objetos con interfaz en las aplicaciones para Smart Devices.

El List del Work With habíamos visto que se invocaba de esta manera: con el nombre del Work With Devices del objeto que correspondiera, el nombre del nivel, y luego el método List sin parámetros.

Para el Detail en modo View, era la sintaxis que estamos viendo en la pantalla, donde necesitamos pasar la primary key, para identificar de quién queremos mostrar ese detalle.

Para el caso del Detail en modo Edit, debemos indicarle el modo: si se dese insertar, por ejemplo un nuevo orador, si fuera de la transacción Speaker que estamos hablando; si queremos actualizar los datos de un orador, entonces le tenemos que pasar la clave primaria de ese orador; o si queremos eliminar un orador, y también debemos pasar la clave primaria.

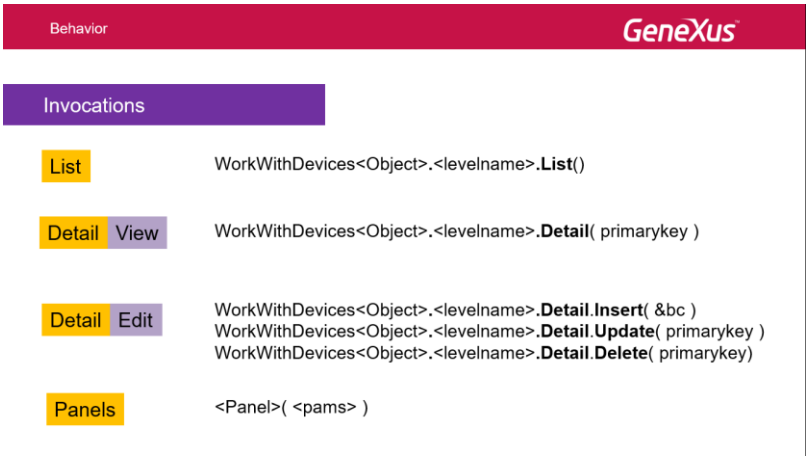
El caso del Insert era especial: recordemos que podíamos no pasar parámetros, ningún parámetro, y en ese caso se va a llamar a la pantalla de Edit y el usuario va a ingresar todos los datos, y luego los va a grabar. O la

otra opción que teníamos, si necesitábamos pasarle algún parámetro a esa pantalla del Detail, a aquella pantalla donde el usuario va a ingresar los datos, debíamos inicializar esos valores que le queremos pasar en un Business Component correspondiente a la transacción de la que estamos queriendo llamar el Work With.

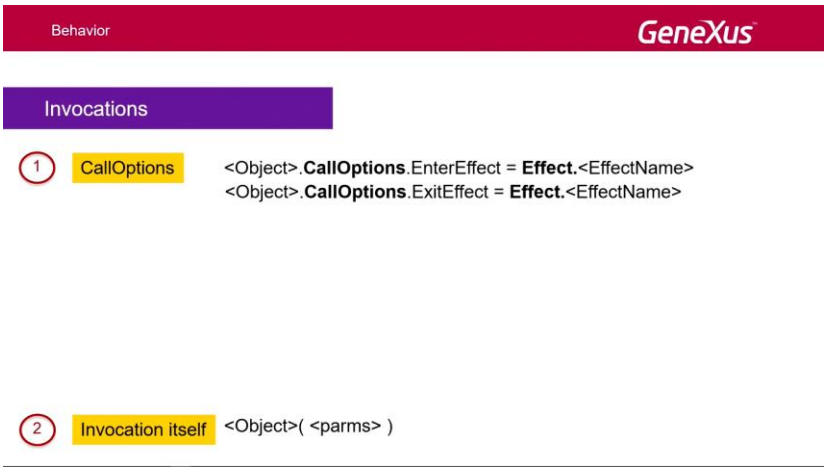
Entonces, en ese caso inicializamos en el Business Component los valores que nos interese pasarle al objeto, y luego esos valores van a aparecer inicializados en la pantalla abierta, desplegada, el usuario va a modificar los valores que le interesen, y va a salvar. Y el Business Component va a volver, va a ser un parámetro de in-out, de entrada y de salida; va a volver cargado con todo lo que el usuario ingresó en esa pantalla.

Por ejemplo, supongamos que en el caso de que el Business Component tenga la clave primaria con autonumerado, y necesitamos saber el valor que se le dio a esa clave primaria, entonces vamos a ahí en ese caso utilizar el Business Component para que vuelva con ese valor cargado. Y como vimos en un ejemplo antes también, de pronto nos interesa saber todos los valores que cargó el usuario.

Bien. Y por otro lado tenemos la invocación a los paneles, que es exactamente igual a la invocación a cualquier otro objeto GeneXus.

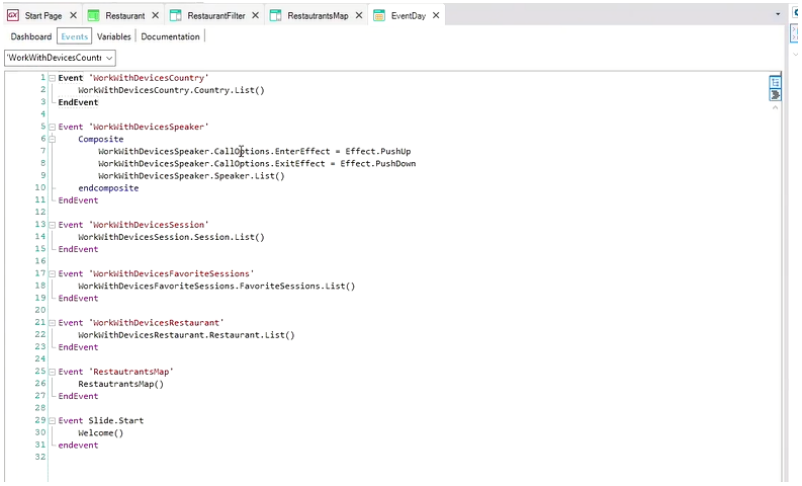


Pero además de invocar al objeto, podemos hacerlo especificando inmediatamente antes, en runtime, la transición de entrada y salida para la interfaz de usuario de la pantalla llamada… veamos que Effect es un dominio predefinido, que tiene como valores los efectos de transición que soporta GeneXus.

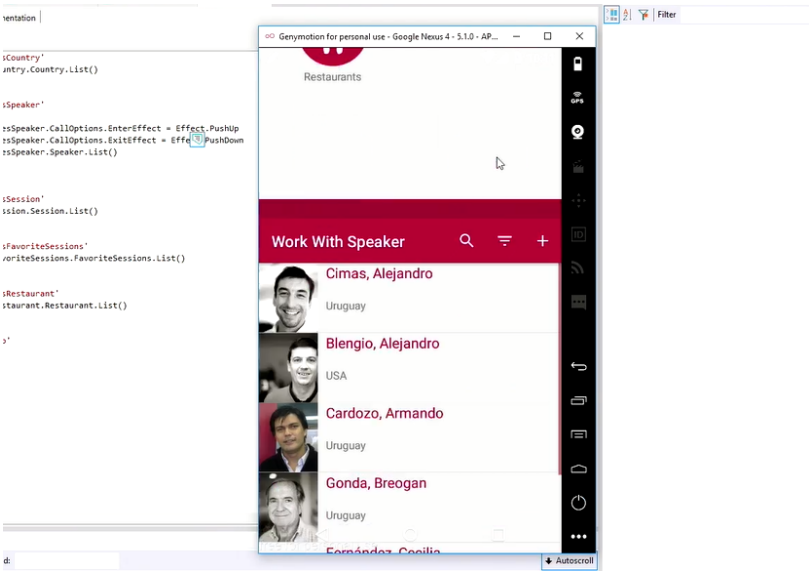


Por ejemplo, voy a ir a GeneXus… a mostrarles cómo, en el Dashboard, entre los eventos, programamos las Call Options, el efecto de entrada y de salida, para la invocación al Work With de Speaker, al List.

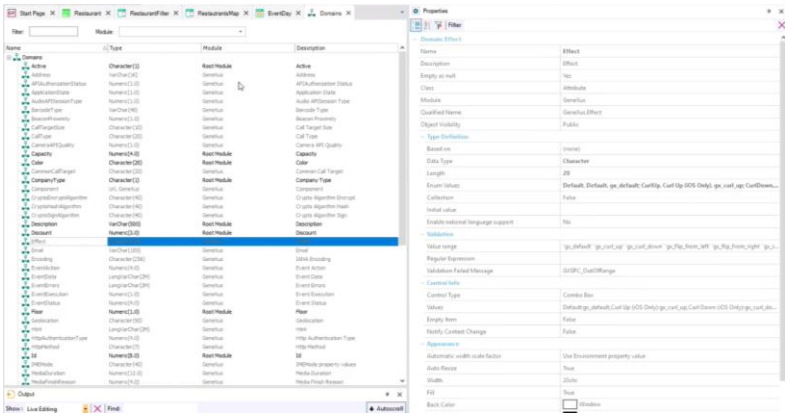
Y recordemos que esto lo habíamos hecho antes, en forma estática, a nivel del form del objeto que se abría… lo hacíamos en la clase Form, configurando entonces los efectos, pero también lo podemos hacer de esta manera en forma dinámica.



Y si vamos al emulador, vemos cómo el efecto de transición se observa al abrir, cómo está apareciendo de abajo. Y al cerrar se cierra en el sentido contrario (PushUp, PushDown).



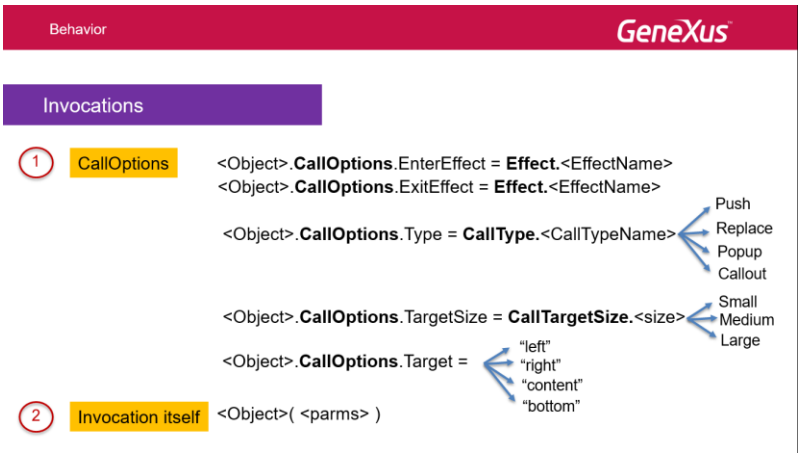
Acá tenemos el dominio predefinido Effect, lo podemos ver si abrimos los dominios… View Domain…Vamos acá a Effect, que es un dominio predefinido que está Read-only, y que tiene entonces los valores que acepta, que soporta la herramienta.



Bien. Podemos también especificar el comportamiento respecto al tipo de call (por ejemplo, si queremos que el objeto llamado se abra como ventana Popup). Observemos que también aquí contamos con el dominio predefinido CallType, que asumirá uno de los cuatro valores que estamos mostrando: Push, Replace, Popup, o Callout. Ahora en un ratito vamos a ver un ejemplo de estos casos; vamos a estudiarlos con un poco más de detenimiento.

Podemos especificar el tamaño de la pantalla en la que se va a abrir el objeto llamado, y la región (target) que ocupará el objeto llamado.

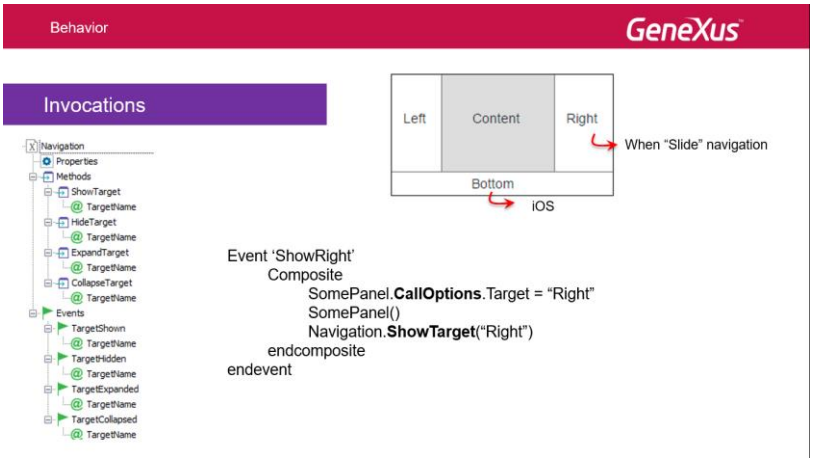
Es por eso que antes de hacer la invocación al objeto debemos definir las calloptions deseadas; pueden ser una o varias.



Recordemos que contamos con las regiones “right” para estilo de navegación Slide, y “bottom” en iOS, como habíamos visto cuando estudiamos el estilo de navegación de la aplicación.

El external object Navigation, que habíamos mencionado en aquella oportunidad, permite a los desarrolladores mostrar u ocultar contenido en forma dinámica en la región que se desee.

Por ejemplo, si el Navigation Style de la app es Slide y el panel central contiene un evento definido como se muestra, no sólo hay que cargar el panel en la región derecha, con la CallOption target que estamos viendo, sino que hay que desplegarla también. Y para ello se utiliza entonces el método ShowTarget del external object Navigation.



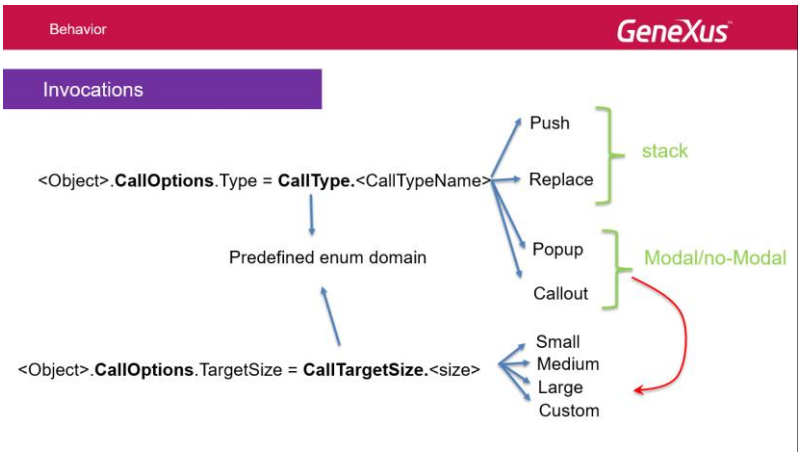
El tipo de invocación, el CallType, define qué va a suceder con el stack de invocaciones y con la forma y modo de la ventana en la que se abrirá el objeto llamado.

Los primeros dos tipos presentados (Push y Replace) definen qué va a suceder con el stack de invocaciones cuando se realice la llamada, que tendrá que ver con a qué objeto se vuelve al finalizar la ejecución del llamado o al hacer back.

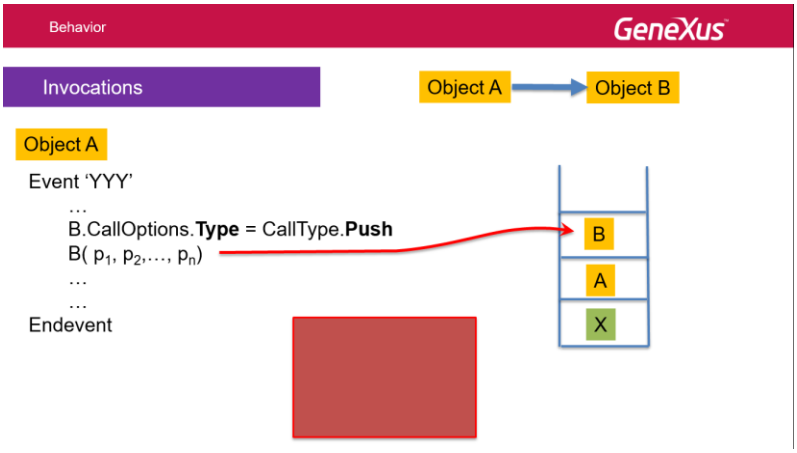
Los últimos dos tipos (Popup y Callout) definirán si el el objeto invocado funcionará como una ventana Popup o como Callout. Para Popup, además, la pantalla será modal o no, dependiendo de si hay parámetros devueltos, o no los hay. En el caso del tipo Callout, será no modal: haciendo tap fuera del área del Callout, de la pantallita, se habilitará al llamador.

Es justamente para el caso de Popup o Callout, que aparece la otra calloption: TargetSize, para indicar el tamaño de la pantalla Popup o Callout.

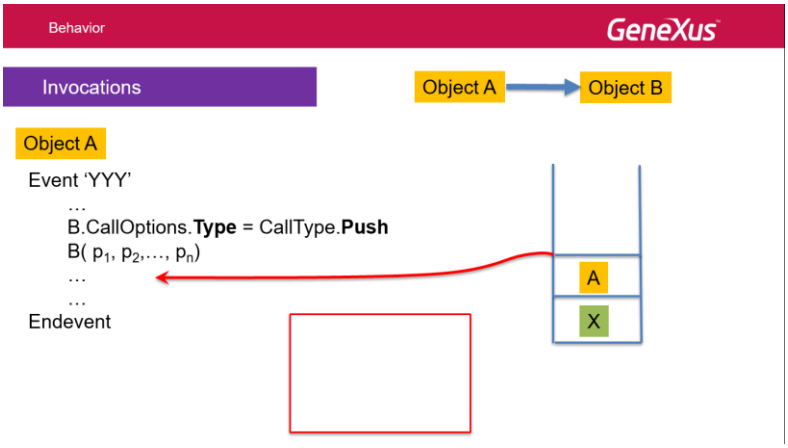
Respecto a los tipos Push y Replace, Push es el default de toda invocación, es decir que si no lo especificamos de esa manera es que se va a abrir: como Push.



Supongamos que un objeto X llamó a un objeto A. Si ahora desde A, en un evento llamamos a un objeto B, con el tipo de call **Push**, el objeto llamado es colocado arriba en el stack, su pantalla se abre sobre la pantalla del llamador, ocupando exactamente el mismo lugar…

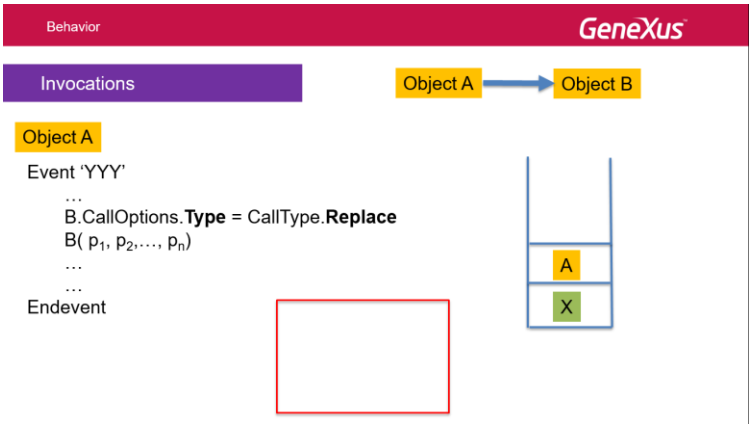


…y el llamador espera para continuar su ejecución a que termine la ejecución del objeto llamado, B, que es así eliminado del stack.

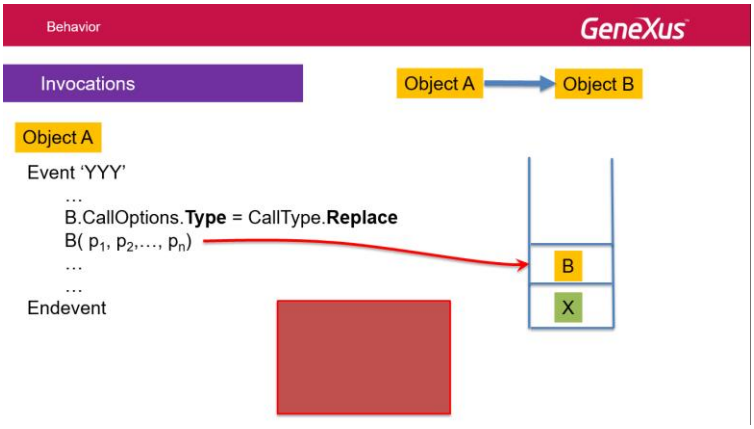


La ejecución continúa con el siguiente comando, que le seguía a la invocación al objeto B.

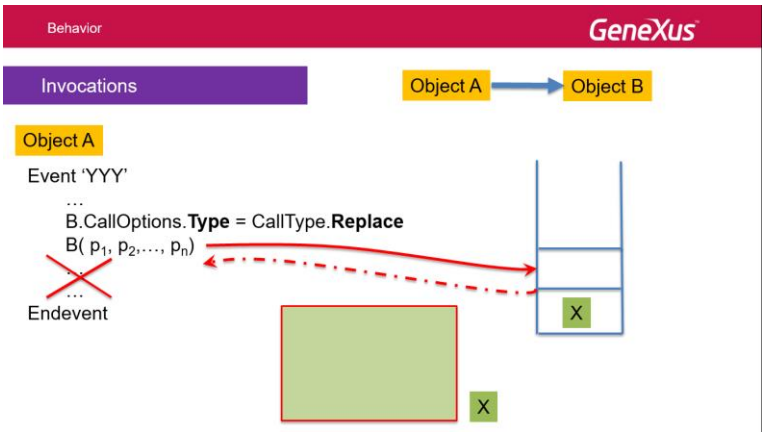
Si, en cambio, desde el objeto A, llamamos a B con el tipo de call Replace…



el objeto llamado también se abrirá ocupando exactamente la misma área de pantalla que el llamador, pero va a sustituir en el stack al objeto llamador…



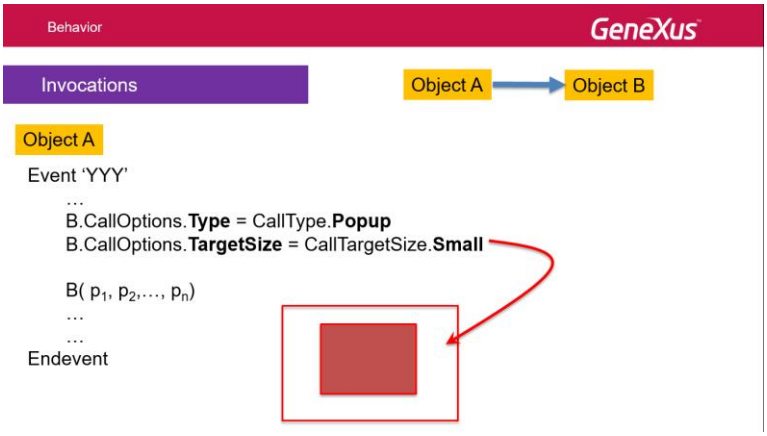
…por lo que, cuando termine su ejecución, no volverá a continuar la ejecución del evento de A, sino que volverá al objeto que estuviera antes en el stack, en este caso el objeto X. Por tanto lo que seguía en el evento evidentemente no se ejecutará.



Respecto a Popup y Callout: Veamos el tipo Popup.

Si no se modifica el TargetSize, ocupará la misma área de pantalla que el llamador. En caso contrario, ocupará el área que hayamos especificado con TargetSize.

Si entre los parámetros de invocación, alguno es output, entonces el diálogo será modal, es decir, el llamador va a esperar el retorno de la ejecución de B para continuar. Si ninguno de los parámetros es de output, entonces el diálogo será no modal.



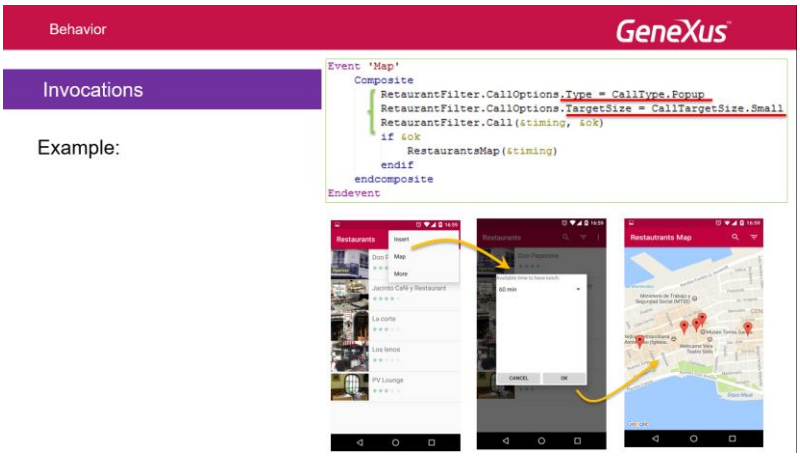
Veamos un ejemplo que ya habíamos presentado anteriormente.

Cuando el usuario elige ver la lista de restaurantes queremos darle la opción, a través del botón “Map”, de que los pueda ver en un mapa. Pero antes de mostrárselos, queremos darle la posibilidad de mostrar sólo aquellos restaurantes que se comprometen a que el cliente almuerce en no más del tiempo del que dispone.

Observe que la primera pantalla, es el nodo List del Work With de Restaurants; la segunda es un panel, que ya habíamos visto, de nombre “RestaurantFilter”; y la tercera es un panel de nombre “RestaurantsMap”, cuyo layout es casi idéntico al del List de Restaurant, con la diferencia de que el grid correspondiente va a tener la propiedad control type en “SD Maps” y que además ese grid filtra en sus condiciones por el tiempo del que dispone el cliente para almorzar, tiempo que es recibido por parámetro.

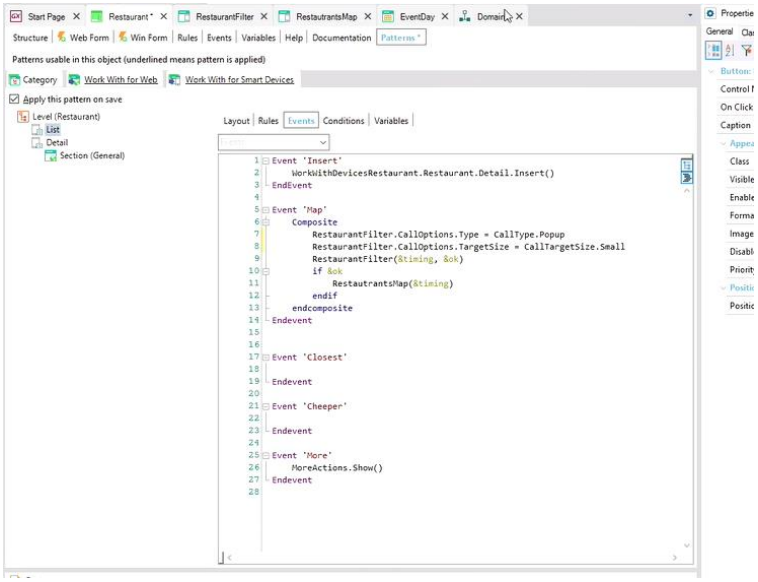
Lo que deseamos es que la pantalla “RestaurantFilter” no se abra como una pantalla independiente, ocupando toda el área de la pantalla, sino que se abra como un Popup, así como lo vemos aquí (modal porque vamos a llamar a este panel para que el usuario elija un valor, y cuando presione OK devuelva ese valor a quien lo llamó), y queremos que ocupe un área menor de pantalla, un área Small.

Entonces lo que hacemos, antes de realizar la invocación a ese panel, “RestaurantFilter”, al que le pasamos el timing y la variable &ok (que son dos variables que van a volver devueltas por ese panel), es configurar las CallOptions: Type como Popup, y TargetSize como Small.

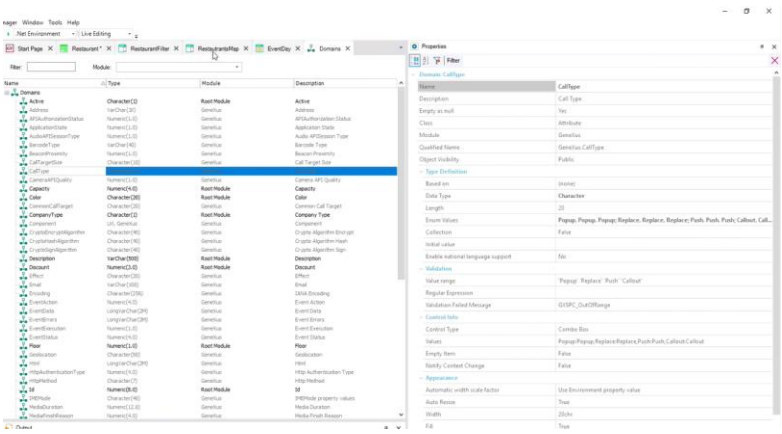


Veámoslo en GeneXus.

Teníamos en el Work With de Restaurant, a nivel del List, el botón Map, y si vamos a ver como estaba configurado su evento, ya habíamos dejado comentado esto. Vamos a des-comentarlo…



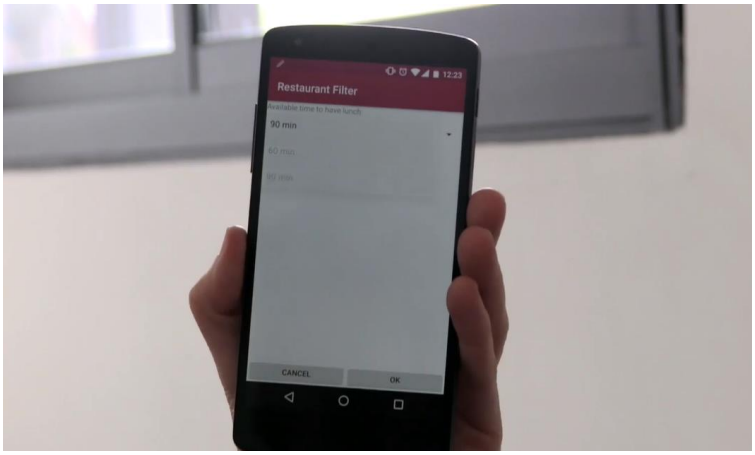
…y vemos que el dominio CallType también es un dominio enumerado, predefinido, que asume los valores Popup, Replace, Push, y Callout, como habíamos mencionado.



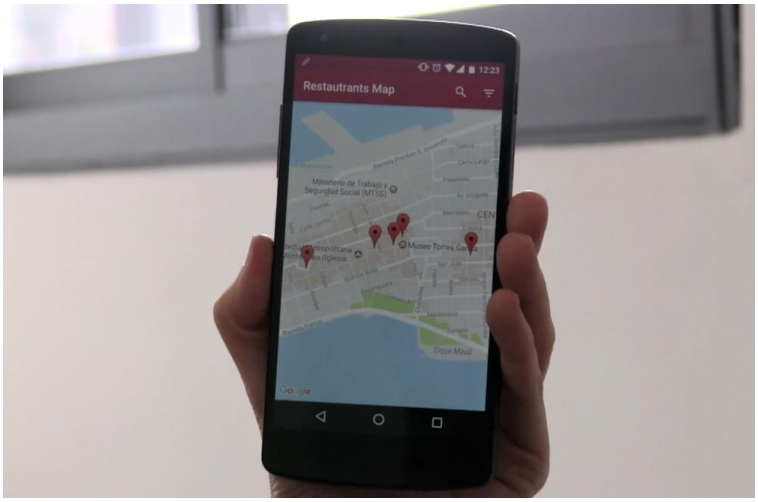
Bien. Y lo que está haciendo entonces es configurar esos dos tipos de invocación, y luego llamando al objeto “RestaurantFilter”, pasándole las dos variables.

Entonces, si lo vemos en el dispositivo antes de volver a generar la aplicación… veamos cómo funcionaba cuando estaban comentadas esas dos CallOptions.

Llamábamos al List de Restaurants, y vamos a ver la opción Map… y vemos que nos está desplegando ese panel a pantalla completa, como una invocación común y corriente. Acá elegíamos el valor que nos interesaba, por ejemplo 90 minutos...



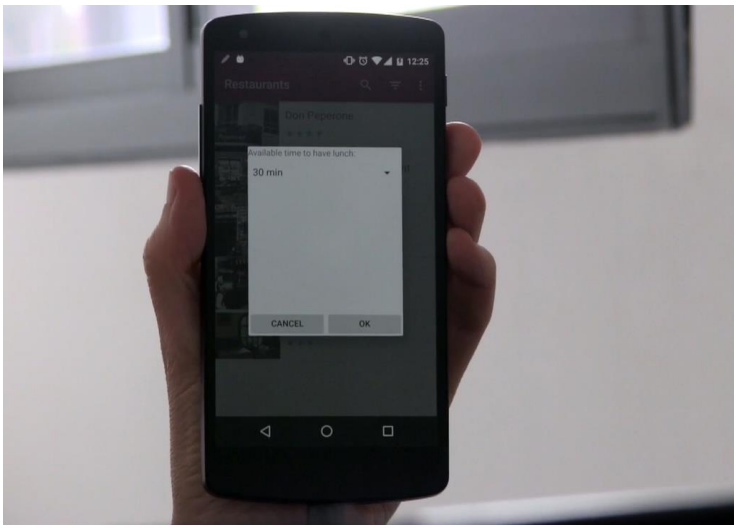
Y al dar OK nos abría entonces este otro panel que nos mostraba esos restaurantes en el mapa.



Vamos entonces ahora a hacer un Build de la aplicación; yo justo no tengo el teléfono conectado todavía, sino con Live Editing ya nos mostraría automáticamente el resultado del cambio, de haber des-comentado los comandos de CallOptions. Entonces ahora lo que voy a hacer es enchufar y les voy a mostrar ya esto ejecutándose.

Vamos a hacer un Run.

Bueno, levantó la aplicación… vamos otra vez a hacer lo mismo… Map… y ahora vemos cómo está saliendo como un Popup y ocupando esa área de pantalla; todo lo demás es igual.



Bien. También podemos definir el tamaño de la ventana Callout o Popup en dips o porcentajes relativos al padre.

Con esto cerramos el tema comportamiento, por lo que terminamos con lo que nos proponíamos. Hemos visto entonces cómo diseñar la aplicación y cómo programar el comportamiento. Pasemos al siguiente tema.