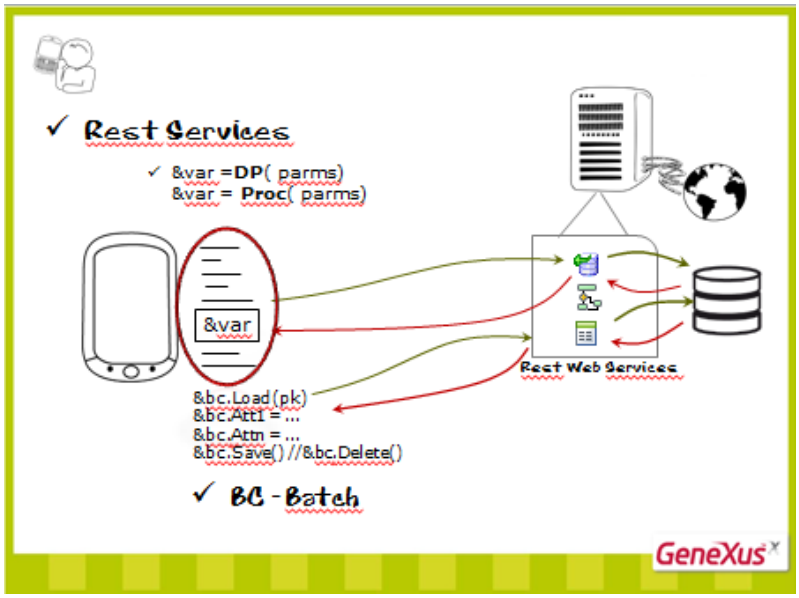
Eventos en Smart Devices

Tenemos eventos cuyo código se ejecuta en el servidor y eventos cuyo código se ejecuta en el cliente (es decir, en el dispositivo).

Queremos abordar ahora el tipo de acciones que pueden escribirse dentro de un evento de usuario.



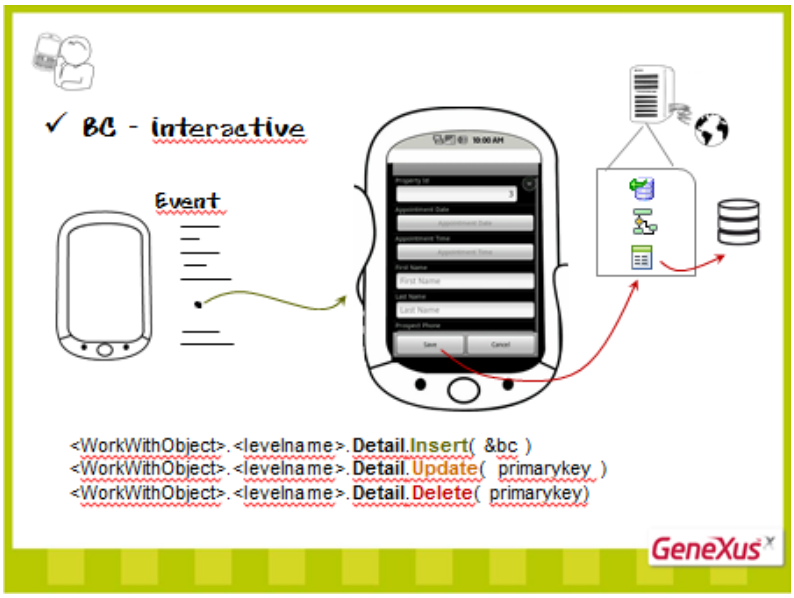
Imaginemos que este es el código del evento en el cliente. Por el otro lado tenemos el servidor Web y la base de datos. ¿Qué cosas podemos hacer aquí?

Llamar a servicios rest del servidor, como data providers o procedimientos que busquen en la base de datos y nos devuelvan la información que cargaremos en una variable.
Necesariamente deben estar expuestos como servicios rest. No podemos llamar a un

procedimiento interno desde el dispositivo. Por ejemplo, corresponde a haber escrito en el Evento:

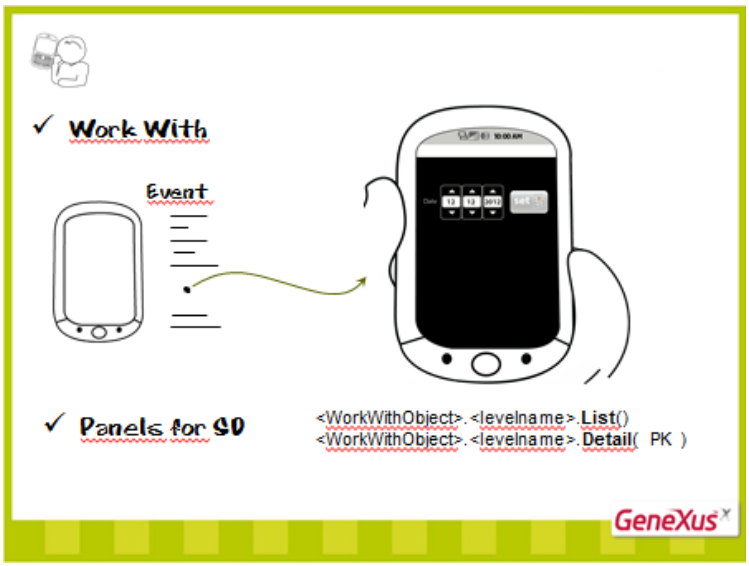
También podemos querer dentro de un evento ingresar un nuevo registro a la base de datos sin tener que pedir información al usuario. Esto se hace como en cualquier otro objeto GeneXus, con los métodos y propiedades del BC, a excepción de que aquí también deberá estar expuesto como servicio rest, dado que estamos invocando desde el dispositivo. Es el caso de actualización batch.

Resumiendo: podemos invocar a servicios Rest: Data Providers, Procedimientos y Business Components.



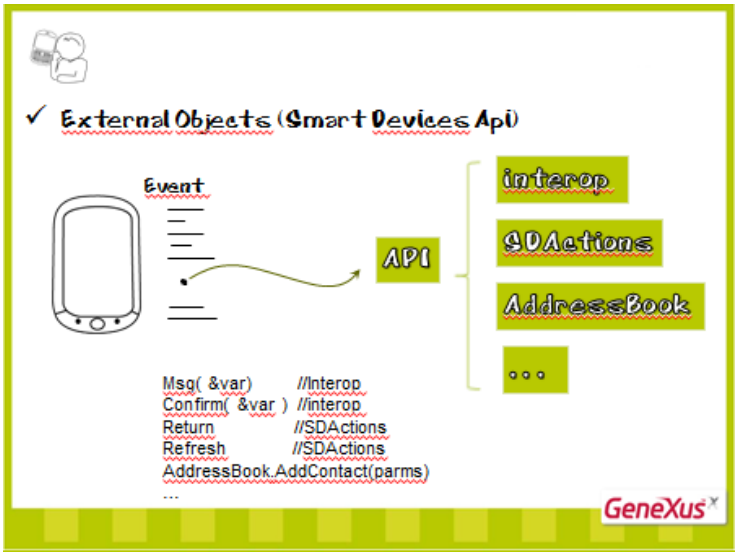
Como vimos, también podemos llamar a la pantalla de Detail del WorkWith, para insertar, actualizar o eliminar. A través de la pantalla invocada, se le piden los datos al usuario y luego se realiza la operación correspondiente (lo que internamente se traducirá en una invoación al BC rest).

Por tanto, estamos invocando al Detail de un Work with, que utilizará internamente el BC rest cuyos datos son pedidos al usuario a través de la pantalla Edit.

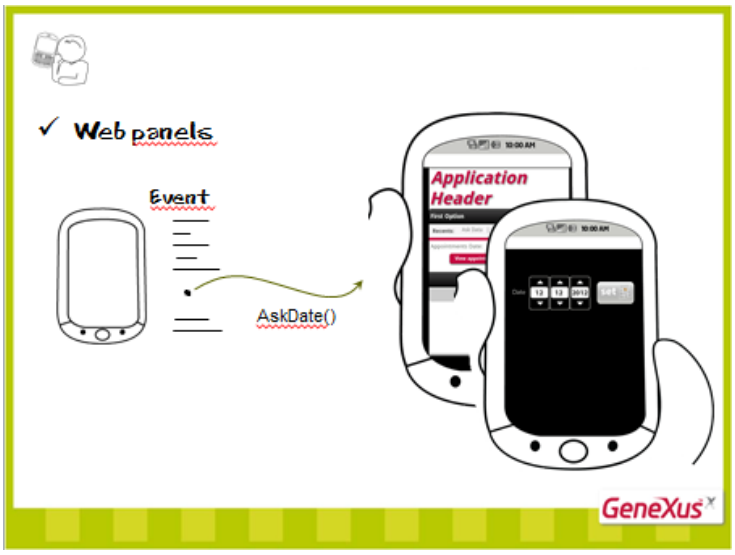


También podríamos simplemente querer llamar al List o al Detail en modo view.

Así como a objetos Panels for Smart Devices, que son pantallas un poco más libres que las de los work with vistos.

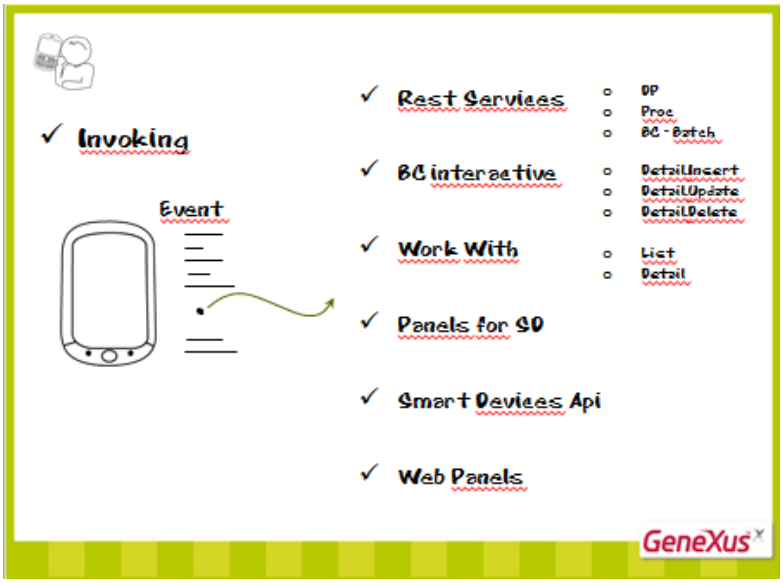


…O utilizar algunas de las funcionalidades provistas por las apis, como desplegar un mensaje en la pantalla, pedir confirmación al usuario para continuar, volver al llamador, refrescar la pantalla, agregar un contacto a la libreta de direcciones, etcétera. Observemos que de estos ejemplos, salvo el refresh, todas las demás acciones se resuelven sin ir al servidor a ejecutar nada.



También se puede invocar a un web panel, es decir, a un panel de la aplicación web, que despliega y permite ingresar datos pero para ese tipo de aplicaciones (como el Panel for Smart Devices lo hará para aplicaciones Smart Devices).

Por ejemplo, aquí estamos invocando al web panel de nombre AskDate(). Éste se abrirá en el navegador del dispositivo (al que se le oculta el marco, para que luzca más parecido al resto de la aplicación). ¿Qué otra forma tendríamos de ejecutar un objeto web, que a través de un navegador web?



Hasta aquí vimos todas las invocaciones posibles:

A servicios rest (Data providers, procedimientos y Business Components)

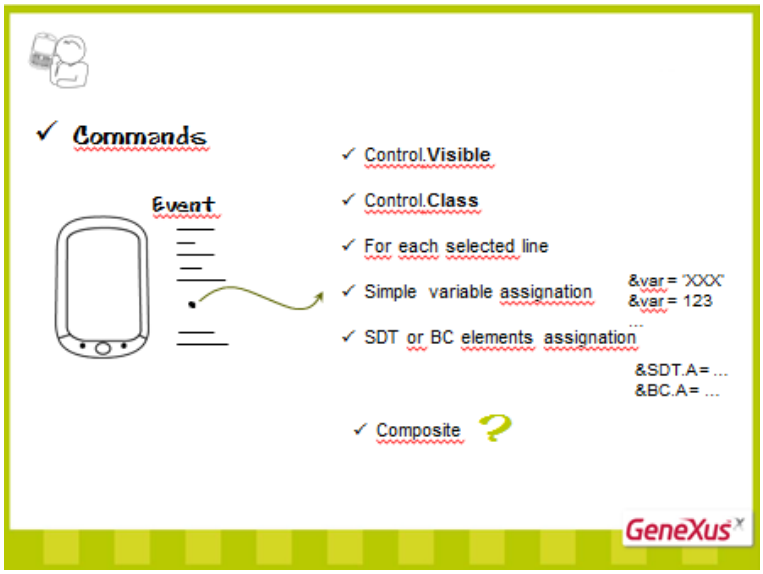
A pantallas de Detail del Work With en modo Edit, para hacer Insert, Update o Delete interactivamente.

A pantallas del Work With para visualizar información (tanto List como Detail en modo View)

A paneles para Smart Devices, que estudiaremos en otro video.

A cualquiera de las apis para Smart Devices provistas por GeneXus, a través de sus métodos.

Y a Web Panels (este es el único caso de objeto del servidor, que no es invocado como servicio rest. La razón es simple: en verdad es un navegador el que se está ejecutando de manera transparente)



Pero aparte de los comandos de invocación que acabamos de ver, tenemos estos otros, que nos brindan:

La posibilidad de hacer visible o invisible un control de pantalla.

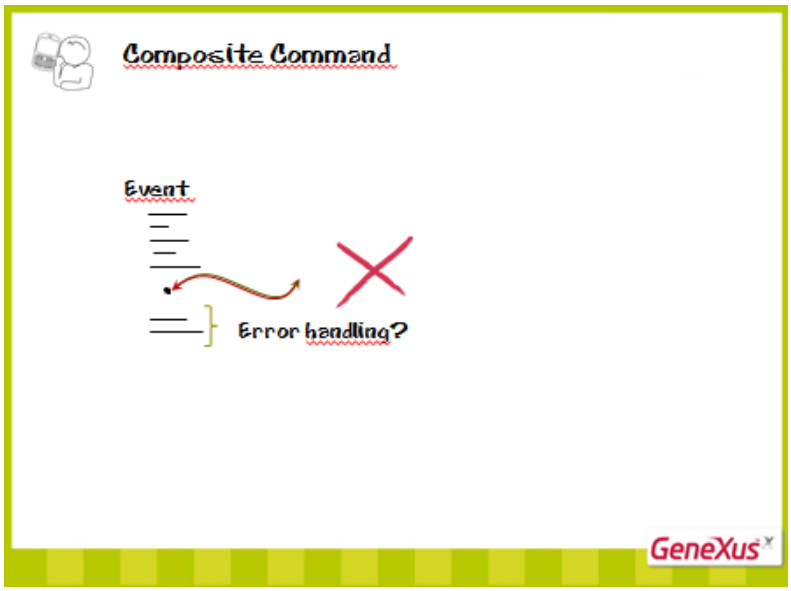
La posibilidad de cambiar la clase de un control de pantalla.

La posibilidad de recorrer de todas las líneas de un grid sólo aquellas que fueron seleccionadas por el usuario antes de ejecutar el Evento.

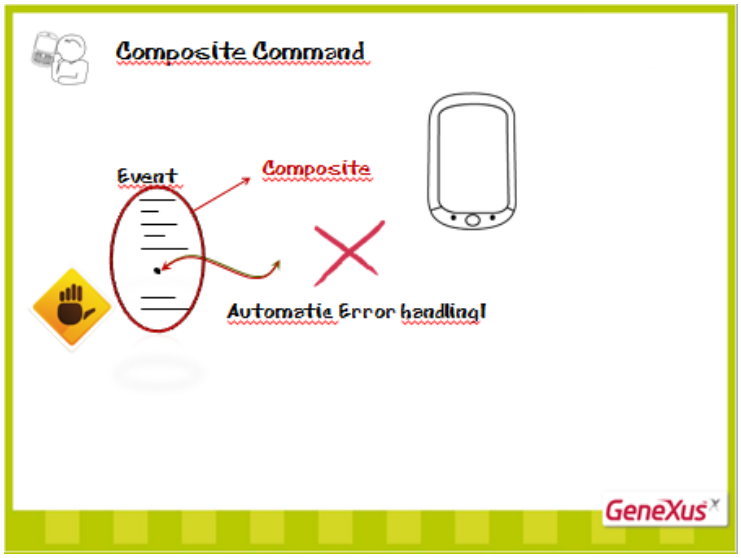
La posibilidad de asignar valores a variables de tipos de datos simples.

Y a variables de tipos de datos compuestos (estructurados o Business Components).

Y por último, tenemos el comando Composite. ¿Para qué lo necesitamos?

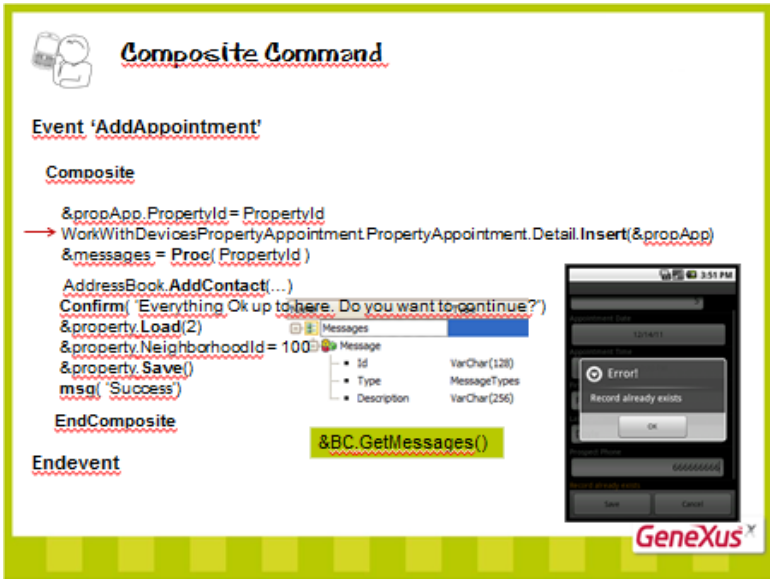


En las aplicaciones Web GeneXus, cuando dentro de un evento que se viene ejecutando, un objeto llamado produce un error, no se interrumpe la ejecución, sigue en la sentencia siguiente y es el desarrollador quien, dentro del propio código del evento, debe encargarse de manejar los errores y programar las acciones a tomar.



Si queremos otro comportamiento, donde cuando ocurra un error en una secuencia de llamadas se detenga la ejecución y se manejen los errores **automáticamente**, desplegándolos en la pantalla sin tener que escribir ninguna programación, necesitamos un comando especial que lo especifique. Este comando es el composite.

Está implementado sólo en Smart Devices y es obligatorio en éstos, toda vez que se realice más de una invocación dentro del mismo evento.



En este ejemplo, vemos una sucesión de comandos, algunos de los cuales son invocaciones.

Si falla la primera invocación, por ejemplo por clave duplicada, entonces se despliega el mensaje de error como vemos, y se **detiene** la ejecución (no se ejecuta la invocación siguiente, al procedimiento, ni nada de lo que sigue).

Si no es el caso, entonces esta invocación sí se realiza. Si en el procedimiento especificamos como parámetro de salida una variable del tipo de datos el SDT predefinido Messages (que, recordemos, es lo que devuelve automáticamente un BC cuando ejecutamos su método GetMessages), y la cargamos dentro del procedimiento con los mensajes de error o advertencias que nos convengan, a la vuelta de la ejecución del procedimiento, esta variable es inspeccionada automáticamente y en caso de error se detiene la ejecución y se despliegan los mensajes en pantalla.

En caso contrario, se ejecuta la siguiente invocación y así sucesivamente.

Acabamos de estudiar los eventos cuyo código se ejecuta en el dispositivo. Ahora le toca el turno a los eventos que se ejecutan en el Servidor.

Son los eventos del sistema: Start, Refresh y Load.

Para introducirlos pensemos el siguiente ejemplo:

Queremos mostrar en el list de propiedades inmobiliarias, un check box que esté marcado para las propiedades que han sido muy visitadas, y una imagen que indicará que la propiedad fue ingresada recientemente al sistema.

{demo}

Para ello, en el List del Work With de Property, hemos agregado una variable booleana…

Y en el Layout la hemos insertado…

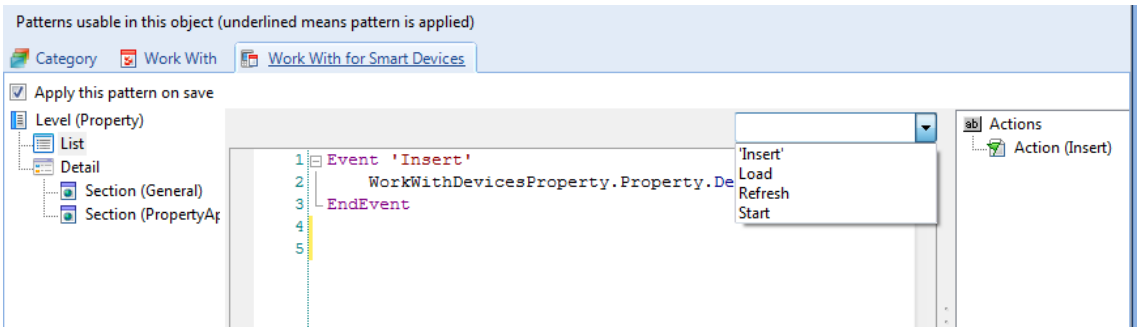
Junto con un control imagen, al que llamamos Image1, para poder referenciarlo luego.

Hemos creado un procedimiento que recibe como parámetro el identificador de propiedad, y devuelve un valor booleano, que es el resultado de evaluar si la cantidad de visitas que tiene agendadas esa propiedad, es mayor de 2. Aquí pondríamos el valor que dispongamos que marcaría la diferencia entre “muy visitada” y lo contrario. Para facilitarnos el testing, pusimos este valor bajo. Así, si tiene más de 2 visitas, carga True en la variable y en caso contrario, False.

Con esto, todo lo que nos resta es programar que al momento de consultarse la tabla de propiedades inmobiliarias para recuperar cada línea que va a cargarse en el grid, además, se cargue el valor de la variable {señalarla}, con el resultado del procedimiento y también se marque si la imagen que indica que la propiedad es nueva, debe mostrarse o no.

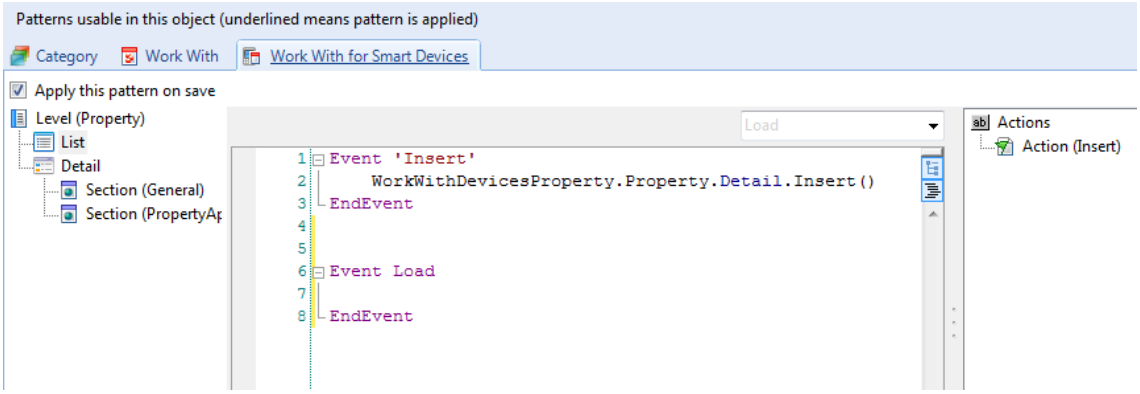
Pero, ¿dónde colocamos este código? En el evento del sistema Load, que es el que consulta la base de datos, recuperando los registros a cargar como líneas del grid.

Aquí podemos ver todos los eventos hasta ahora definidos para el List.



Entre ellos vemos los tres eventos del sistema mencionados.

Elegimos Load. {hacerlo}



Y escribimos:

Hacemos F5.

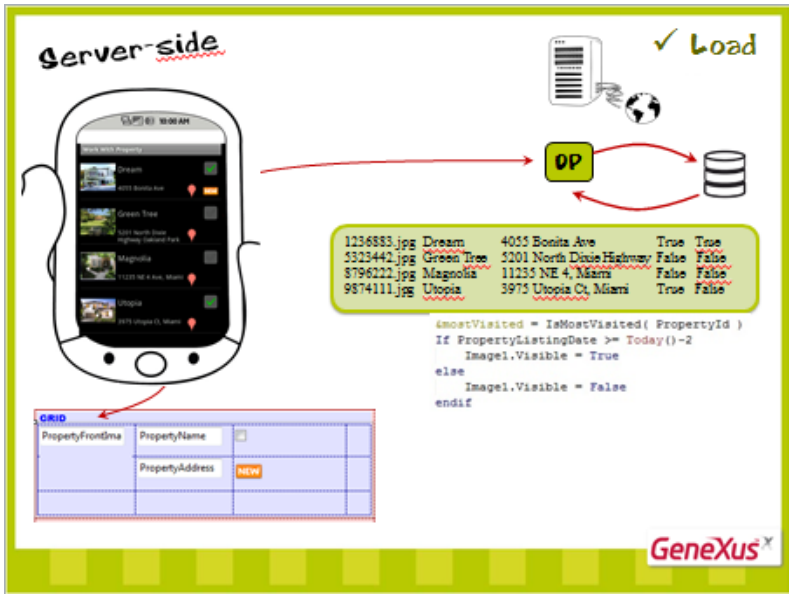
Vemos que Dream está marcada, indicando que tiene más de 2 visitas agendadas. Si vamos a verlas… tiene 3. Además, su fecha de ingreso al sistema es 28 de marzo de 2012. Sabiendo que hoy es 29 y que indicamos que es nueva si se ingresó en los últimos dos días, entonces…está apareciendo la imagen.

Green Tree fue ingresada el 12 de setiembre del año pasado, y tiene 1 visita, por lo que… ni fue muy visitada ni es nueva.

Magnolia es del 26 de junio del año pasado y no tiene visitas agendadas, por lo que…

Por último, Utopía está marcada como muy visitada. Así que tenemos que encontrar más de 2 visitas, y su fecha de ingreso al sistema debe ser anterior al 27 de marzo de 2012…

Lo que, como vemos, se está cumpliendo…

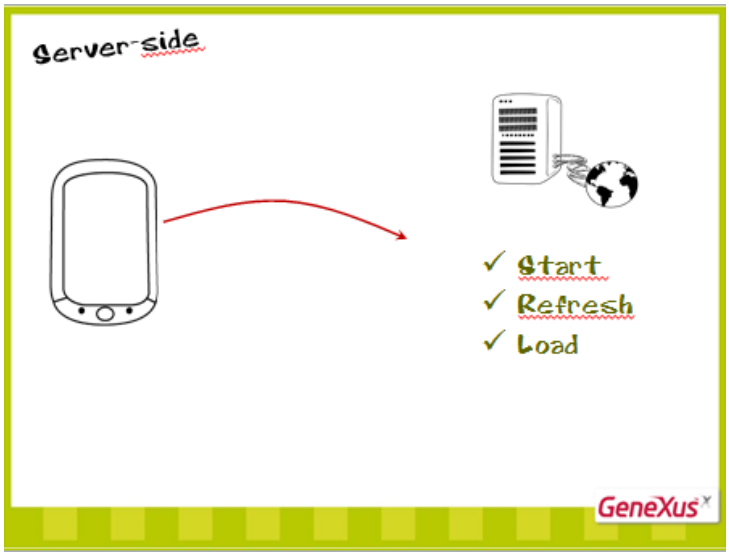


En definitiva, al tener un grid con atributos en una pantalla, estamos implícitamente diciendo que se debe acceder a la tabla de la base de datos correspondiente a esos atributos (y eventualmente a su extendida), y de todos los registros que cumplan las condiciones, devolver al dispositivo una colección con los valores pedidos.

Quien hace este trabajo de recuperar los datos y enviarlos al dispositivo, no es ni más ni menos que un Data Provider Rest, implícitamente creado y generado por GeneXus, de manera transparente para nosotros.

Este Data Provider, por cada registro de la tabla recuperado para ser incluido en la colección devuelta, ejecuta el código del evento Load. De esta manera es que carga para cada propiedad inmobiliaria los valores True o False tanto de la variable mostVisited como de la propiedad Visible del control Imagen.

Este código se ejecuta, por tanto, en el servidor, por lo que podemos utilizar los comandos que usamos habitualmente en aplicaciones web. Por ejemplo, observemos que aquí estamos invocando a un procedimiento, IsMostVisited que es interno.



El Start se ejectuará también en el Server, pero únicamente cuando se ejecuta la pantalla por primera vez. Es decir, la primera ver que se abre esa pantalla del work with.

El Refresh, en cambio, se ejecutará cada vez que deba refrescarse la pantalla. Inmediatamente después del Refresh, se ejecutará el Load (la carga del grid).

Con esto, completamos la exposición que queríamos realizar, acerca de los eventos de las distintas pantallas de un Work With for Smart Devices. Todo lo dicho valdrá también para los eventos de Panels for Smart Devices.

¿Desea implementar paneles que permitan pedir datos al usuario, mostrar datos no necesariamente provenientes de una o varias tablas de la base de datos, dar más flexibilidad a sus pantallas?

Continuará.