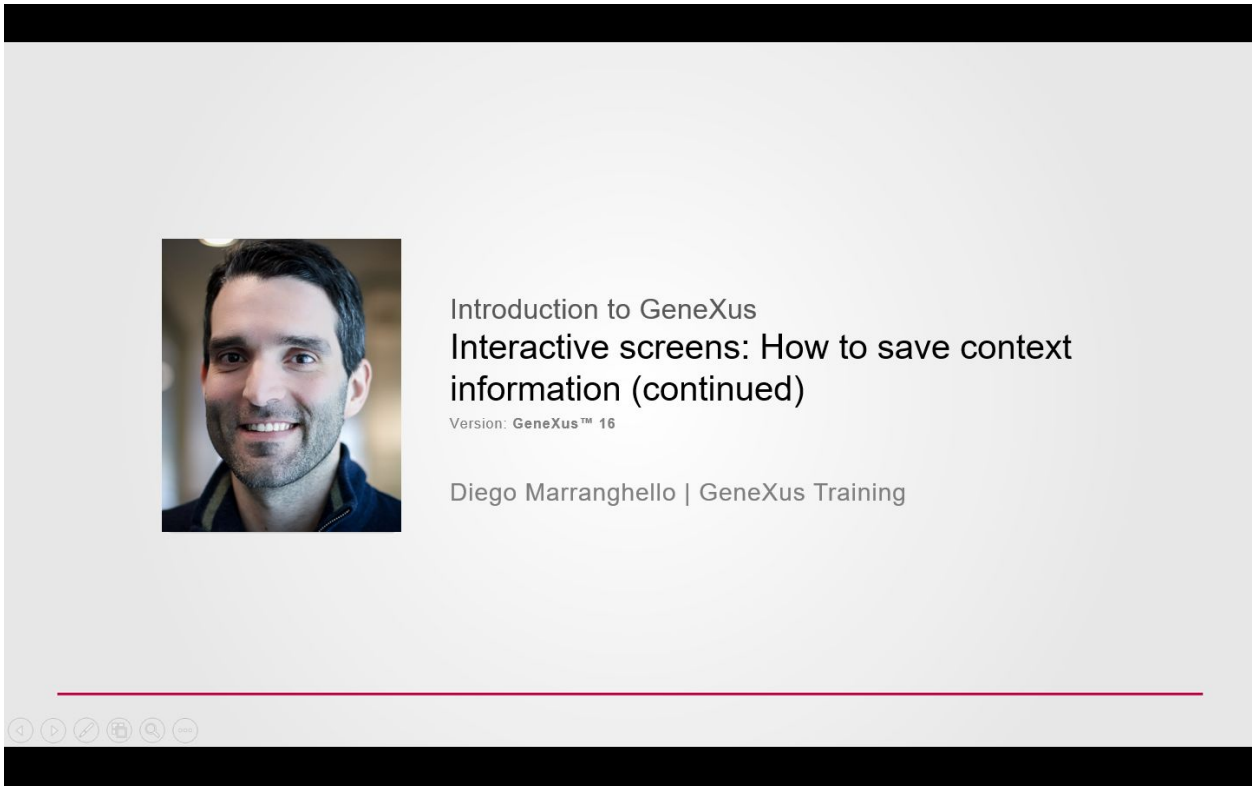
Pantallas interactivas

Como guardar información de contexto (continuación)



En el video anterior vimos cómo mantener datos en memoria, evitando que los mismos se pierdan luego de llamar a otro objeto y posteriormente volver al objeto llamador.

Para esto, lo que hicimos fue crear una variable del tipo WebSession, y guardamos en ella los datos que nos interesaban, que en nuestro caso eran los valores de los tres filtros disponibles.

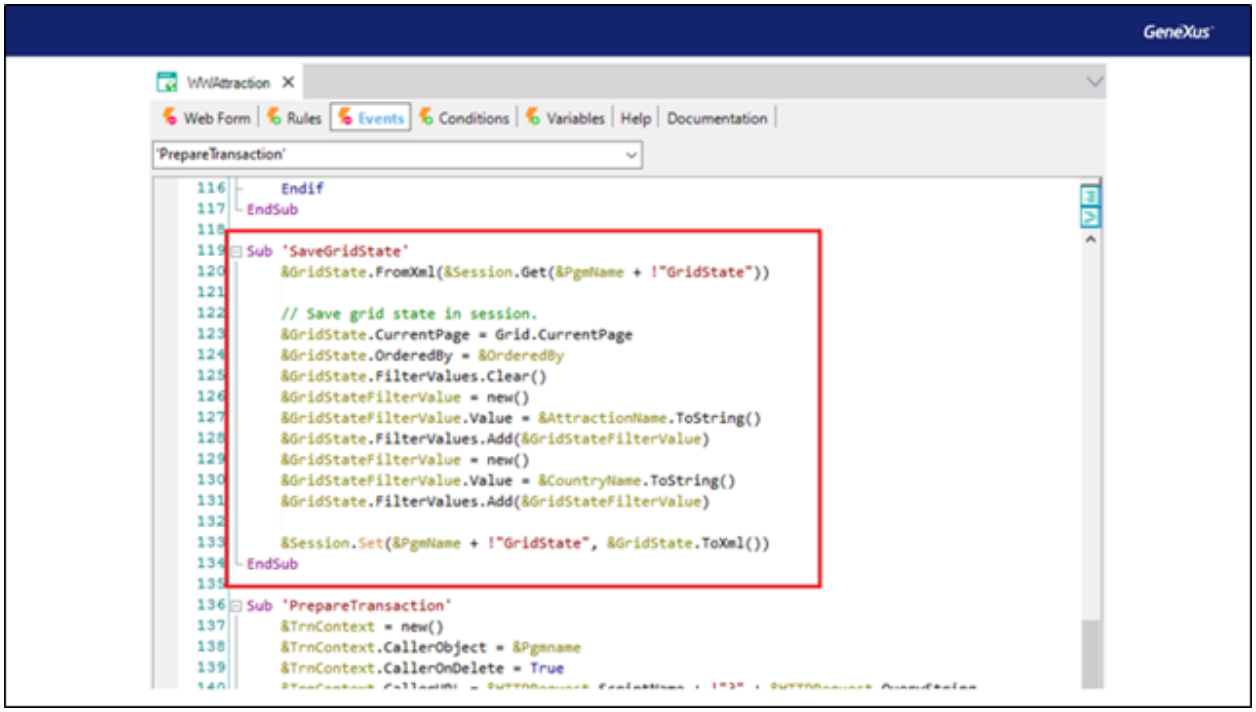
Decidimos guardar cada uno de estos datos en el evento &Update.Click mediante el método SET de la variable de sesión, para posteriormente recuperarlos en el evento Start, mediante el método GET de nuestra variable WebSession.

Esta fue una solución que hicimos a nuestro modo, y como prometimos en el video anterior, vamos a ver ahora cómo lo hace automáticamente el Pattern Work With de la transacción Attraction

Recordemos, que desde este Work With generado automáticamente por GeneXus, cuando ingresamos valores en los filtros y luego actualizamos algún registro desde la acción Update, al retornar los valores se mantienen. Veamos cómo lo hace.

Accedamos al objeto WWAttracion, y luego a la sección eventos de este.

Vemos que hay generada una subrutina de nombre “SaveGridState”



Una subrutina es un bloque de código identificado con un nombre, la cual podrá ejecutarse luego dentro del mismo objeto.

Las subrutinas se definen con el comando Sub.

Una de las ventajas de las subrutinas es que hace más claro el código, facilitando su lectura.

Otra gran ventaja es que permiten reutilizar el bloque de código declarado dentro. De esta manera, si necesitamos ejecutar el mismo bloque de código desde varios lugares del objeto, se escribe una sola vez y puede invocarse desde varias ubicaciones utilizando el comando DO, esto lo veremos en un momento.

Concentrémonos en la siguiente sección del código de esta subrutina, que es lo que nos interesa en este caso.

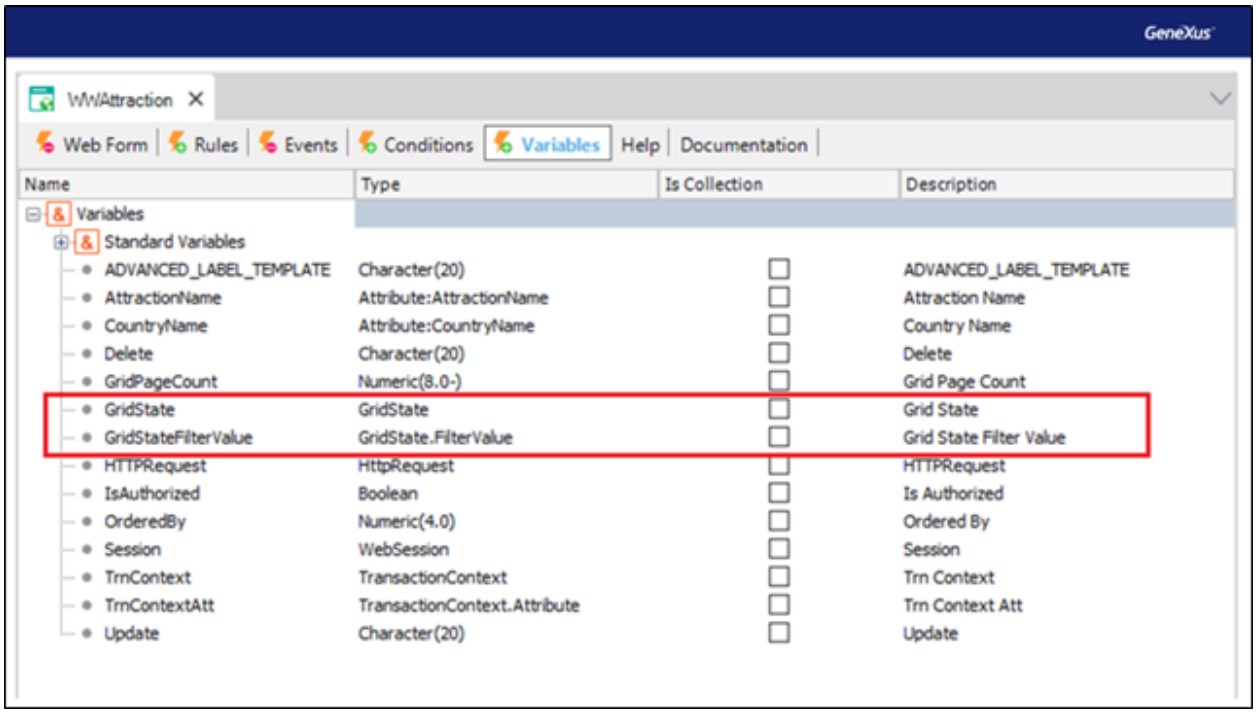
GeneXus

```
// Save grid state in session.
&GridState.CurrentPage = Grid.CurrentPage
&GridState.OrderedBy = &OrderedBy
&GridState.FilterValues.Clear()
&GridStateFilterValue = new()
&GridStateFilterValue.Value = &AttractionName.ToString()
&GridState.FilterValues.Add(&GridStateFilterValue)
&GridStateFilterValue = new()
&GridStateFilterValue.Value = &CountryName.ToString()
&GridState.FilterValues.Add(&GridStateFilterValue)

&Session.Set(&PgmName + !"GridState", &GridState.ToXml())
```

Vemos primero que nada que hay creada una variable de nombre GridState y otra de nombre GridStateFilterValue.

Observemos desde la sección variables como fueron declaradas.

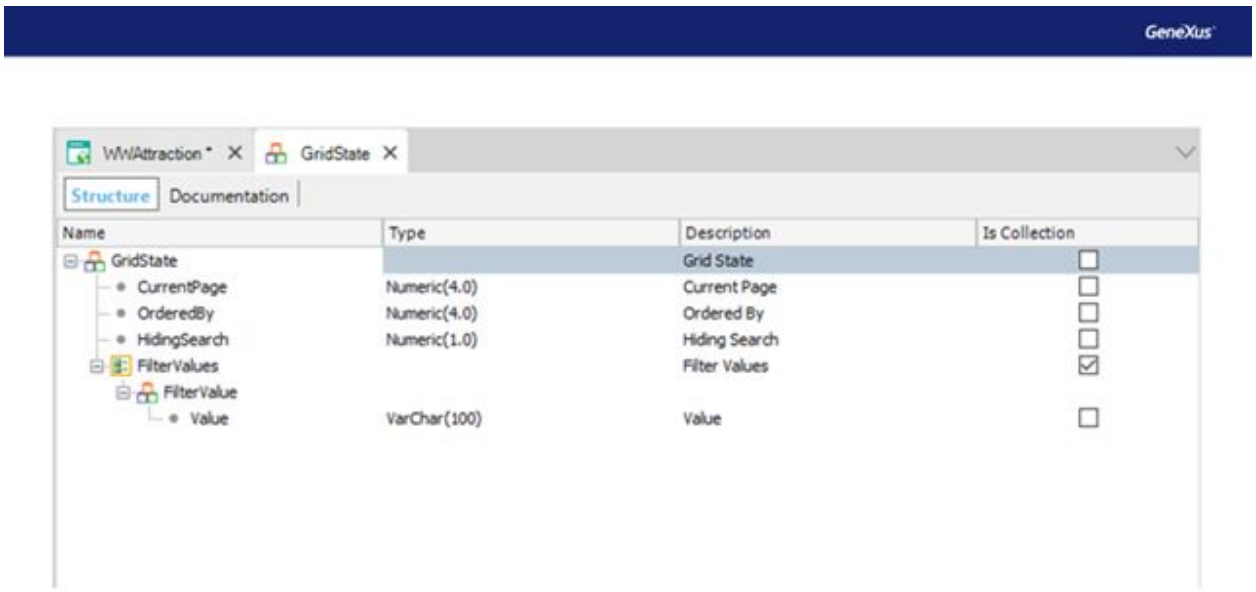


Vemos que GridState es del tipo GridState, y GridStateFilterValue del tipo GridState.FilterValue.

Ahora, ¿a que refieren estos dos tipos?

Si buscamos dentro de los objetos de nuestra KB, vemos que refiere a un objeto del tipo SDT, este fue generado por GeneXus automáticamente al crear el WorkWith.

Veamos qué estructura tiene este SDT.



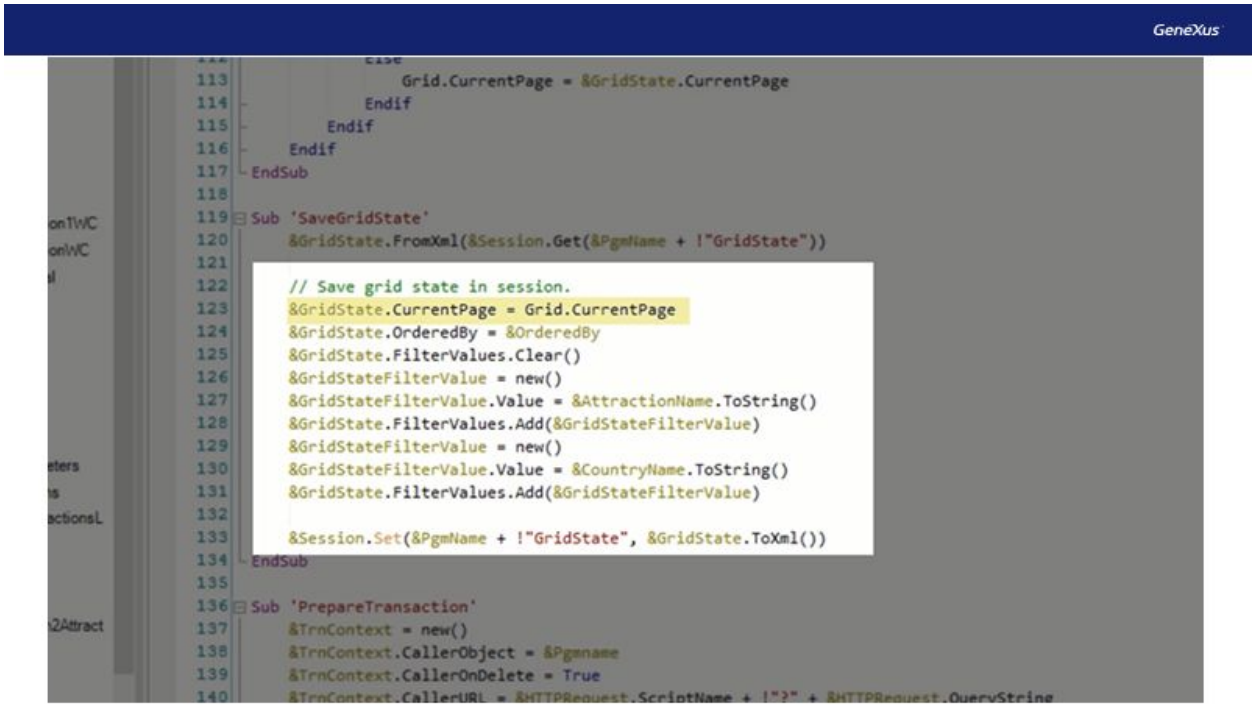
Vemos que hay declarados tres miembros en su estructura principal, de nombres CurrentPage, OrderedBy y HidingSearch. Los tres del tipo Numérico.

Vemos también que hay una subestructura de nombre FilterValue, la cual será una colección.

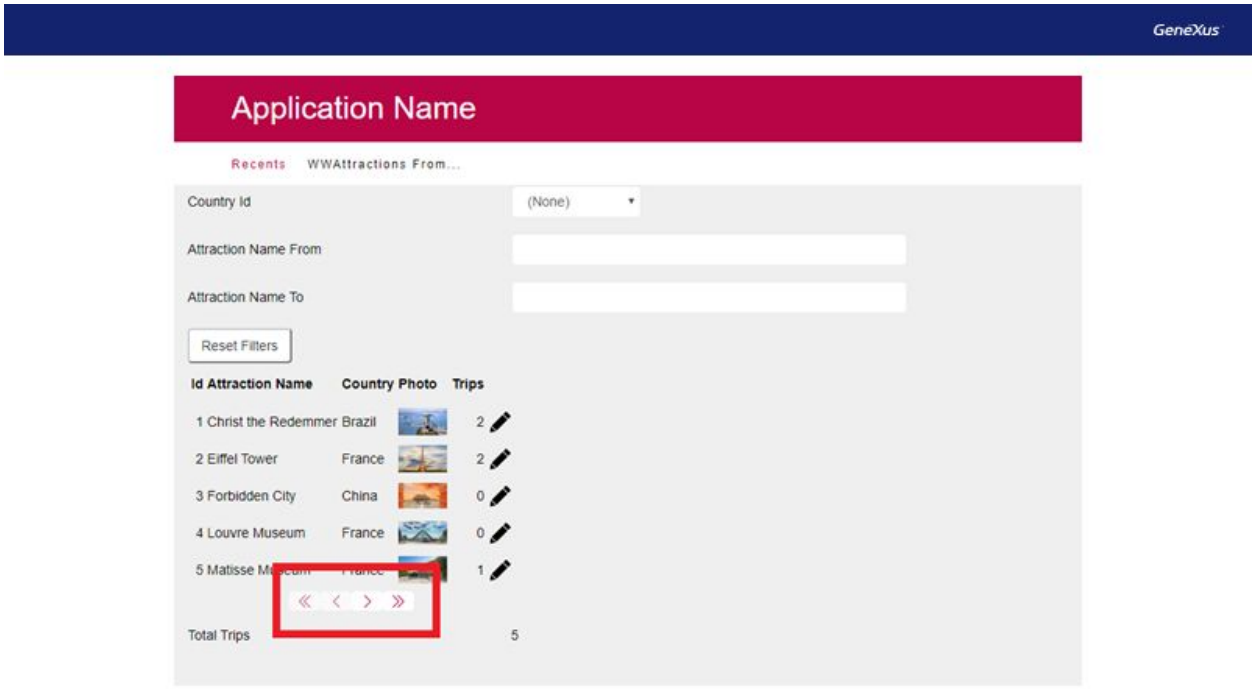
Y dentro de esta subestructura hay declarado un miembro de nombre Value, del tipo Varchar.

Volvamos a la sección eventos del objeto WWAttraction.

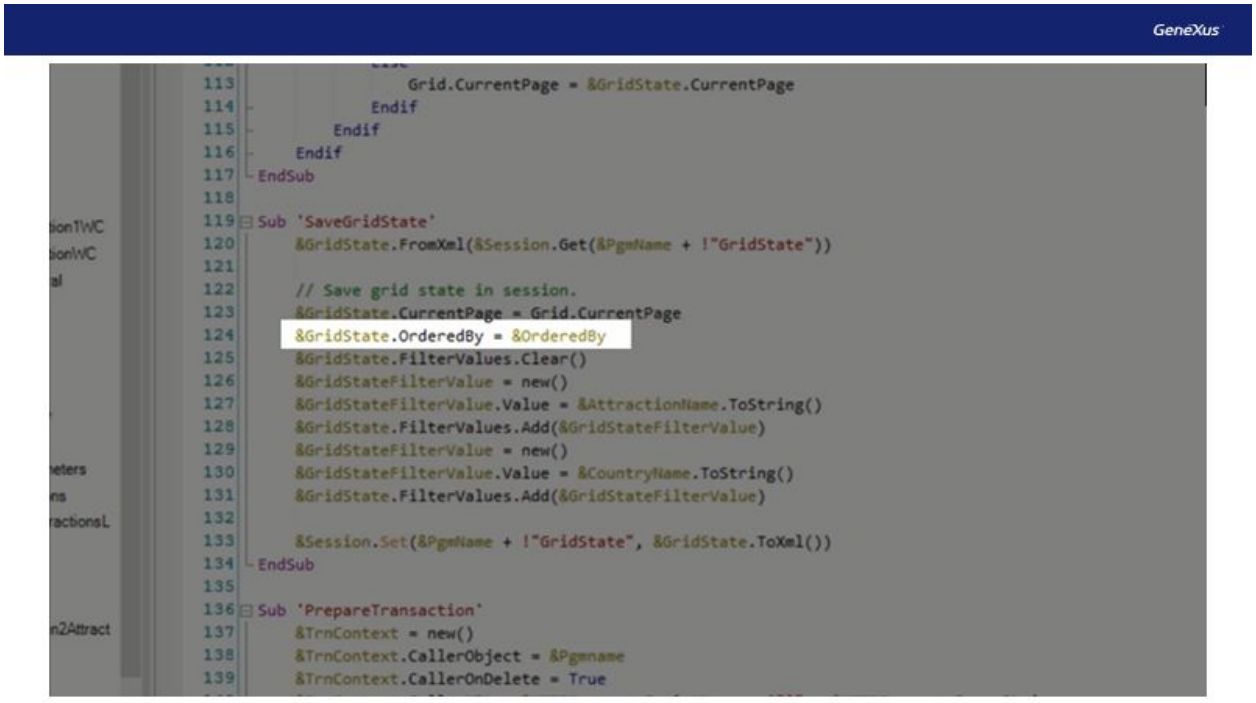
Dentro del código que nos interesa observar, vemos que la primera línea guarda en el miembro CurrentPage de la variable GridState, el número de página del grid en la que estamos.



En nuestro caso no teníamos paginación en el grid, pero por ejemplo si en nuestro WorkWith, configurásemos la propiedad Row del grid, y le ponemos un valor, por ejemplo 5, aquí lo que le diremos es que queremos mostrar 5 filas por página.
Vemos al actualizar que aparecerán solo los primeros 5 elementos, y luego dará la opción de avanzar de página para observar los restantes datos. En ese caso nos interesaría también guardar la página en la que estábamos posicionados. Es por eso que así lo hace el Pattern, para cubrir ese escenario.

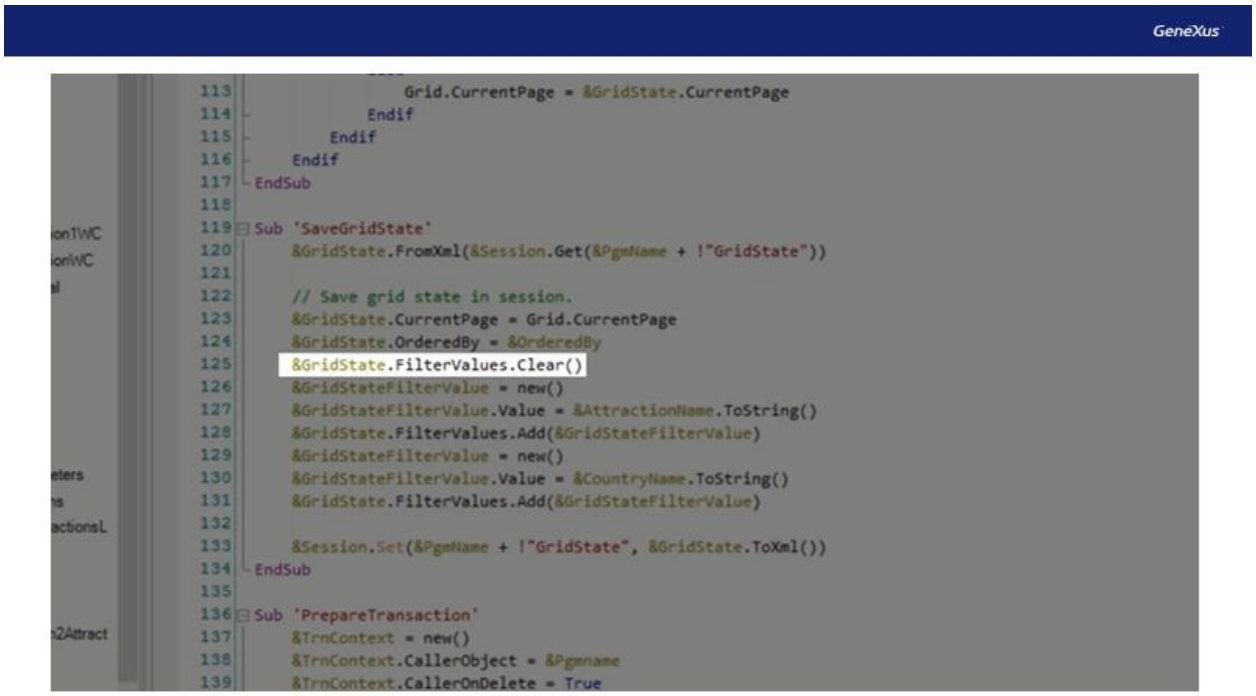


En la siguiente línea se guardará el orden que se le haya podido aplicar a los datos, y se guardarán en el miembro OrderedBy del SDT GridState.



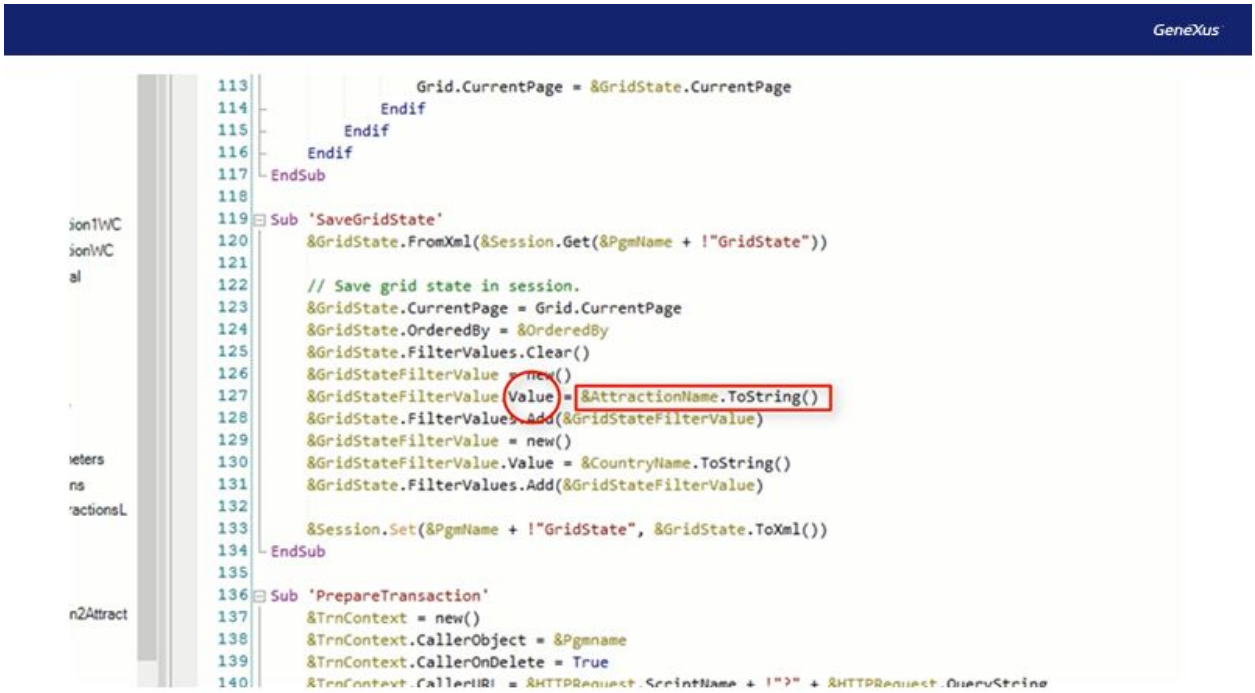
Luego con el método Clear(), se eliminará cualquier dato que hubiese cargado en la colección FilterValues.

Ahora aparecerá en acción la variable GridStateFilterValue, la cual como vimos es del tipo GridState apuntando a la subestructura FilterValue.



Primero que nada, se le asigna el operador new(), el cual devuelve una nueva instancia inicializada de la variable SDT. Para posteriormente poder cargar un valor en ella.

Esto lo hace en el miembro Value, asignándole la variable AttractionName, que es la que utiliza el WorkWith para ingresar el filtro por nombre.



Luego se le agrega a la colección FilterValues esta variable.

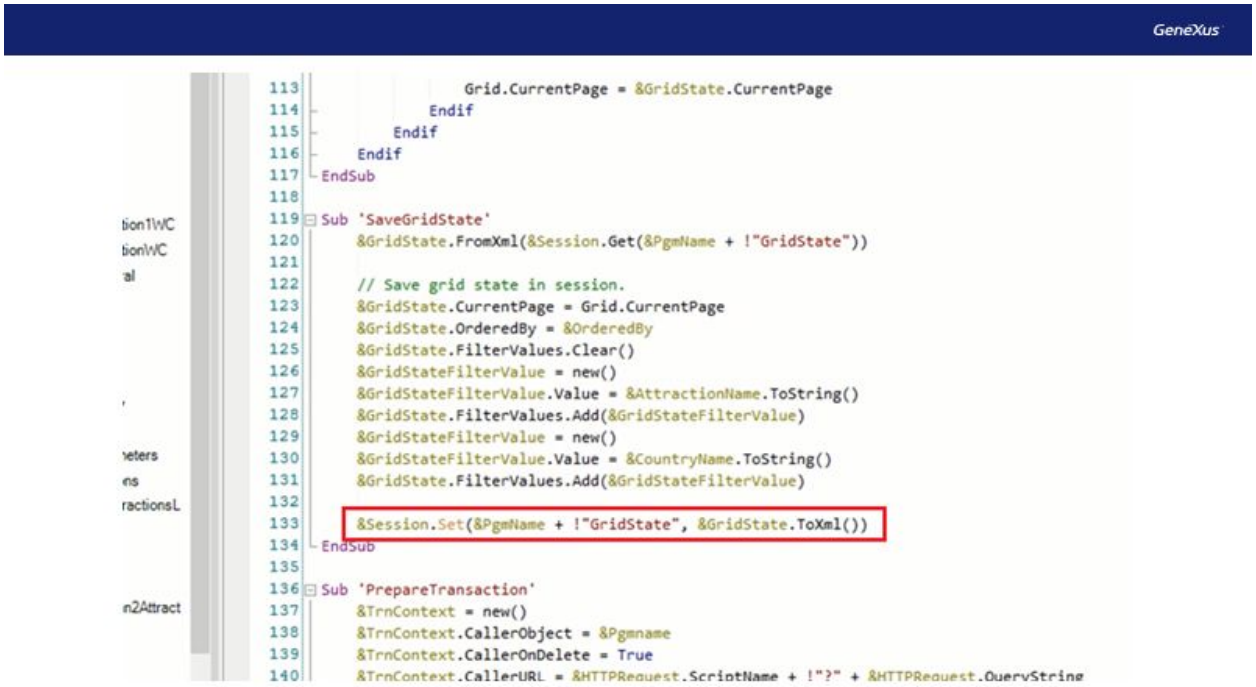
El mismo procedimiento se vuelve a repetir, pero en este caso con la variable CountryName, la cual contendrá el valor del filtro por nombre de país.

Observamos luego que hay una variable de nombre Session, si vamos a la sección variables vemos que esta es del tipo WebSession.

Esta variable será la que guardará la información recopilada en las líneas anteriores.

Para esto, se utilizará el método Set sobre la variable de sesión, pasándole por parámetros una clave y un valor.

Como clave utilizará el valor de la variable PgmName concatenado con el texto GridState.



PgmName, es una variable que almacena el nombre del programa activo. O sea el nombre especificado en la propiedad name del objeto. En este caso “WWAttraction”.

Y como valor guardará la variable GridState. Que como vimos es del tipo SDT y es donde fueron guardados los datos que se quieren almacenar.

Veamos ahora, donde se hace la invocación a esta subrutina, o sea en qué momento será utilizada.

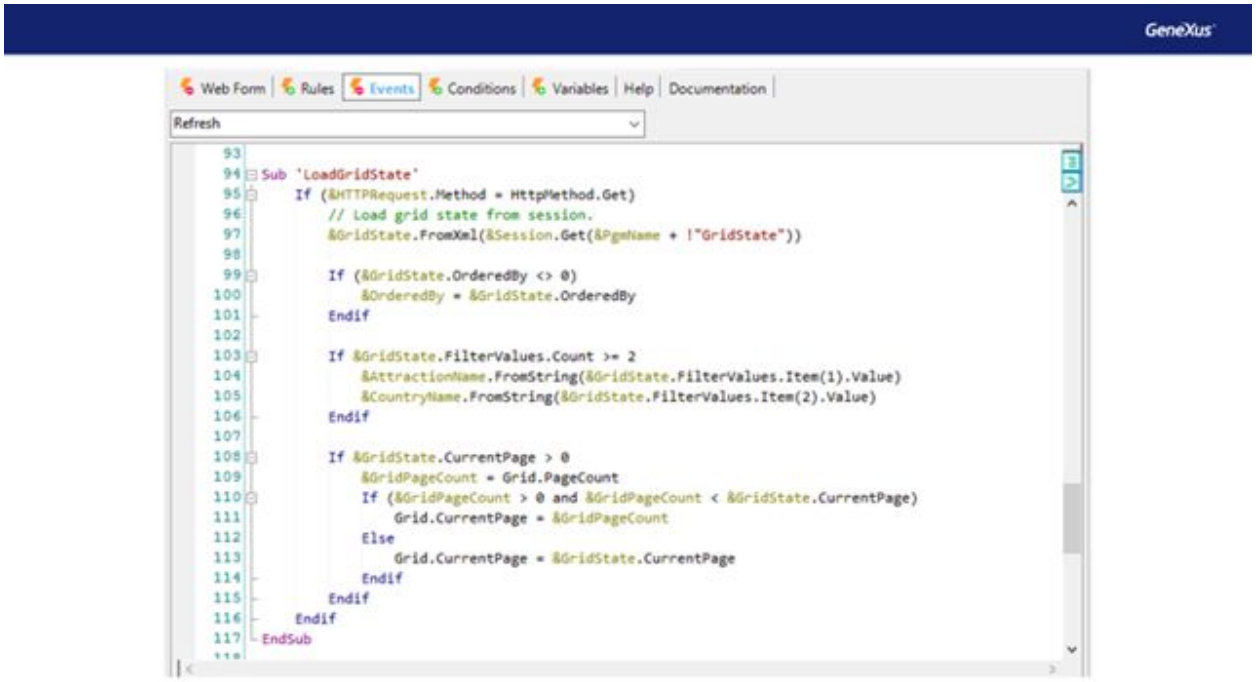
Vemos que es llamada mediante el comando “DO” en el evento Refresh. O sea que cada vez que se dispare este evento, se guardarán los valores de los filtros que interesan.

En el video anterior, para la solución en nuestro WebPanel, nos planteamos guardar los valores de nuestros filtros en este evento. Y decidimos que no era el mejor momento ya que cada vez que se dispara el evento Refresh se guardaran estos valores, y tal vez no tenemos interés en guardarlos porque no vamos a tener la necesidad de recuperarlos luego, y sin embargo lo estamos guardando una y otra vez. Por ejemplo, cada vez que se cambie el valor de uno de los filtros se dispara este evento y se procederá a guardar estos valores.

El Pattern lo hace de esta manera justamente por ser un patrón, y de esta forma contempla todas las situaciones que se puedan dar, el pattern no sabe exactamente el uso que le vamos a dar a la aplicación, es por eso que de esta manera cubre todos los escenarios posibles.

Bien, veamos ahora cómo y dónde es que se recuperará esta información que guardamos.

Observamos que se generó otra subrutina de nombre LoadGridState. Es aquí donde se recuperará la información guardada en la subrutina SaveGridState.



En esta línea, se cargará en la variable GridState, que recordemos es del tipo SDT, la información guardada anteriormente en la variable de sesión Session.

Esto lo hace mediante el método FromXml, el cual recibe una cadena XML desde la cual se carga la información en el objeto SDT.

A ese método, se le pasará por parámetro de donde se quiere recuperar esa información, que en este caso será de la variable Session. Vemos que utiliza el método Get para indicar qué valor quiere recuperar mediante la clave que fue guardado el mismo. Recordemos que la clave asignada para guardar la información, fue la concatenación de la variable PgmName, o sea el nombre del objeto, más la cadena de texto GridState.

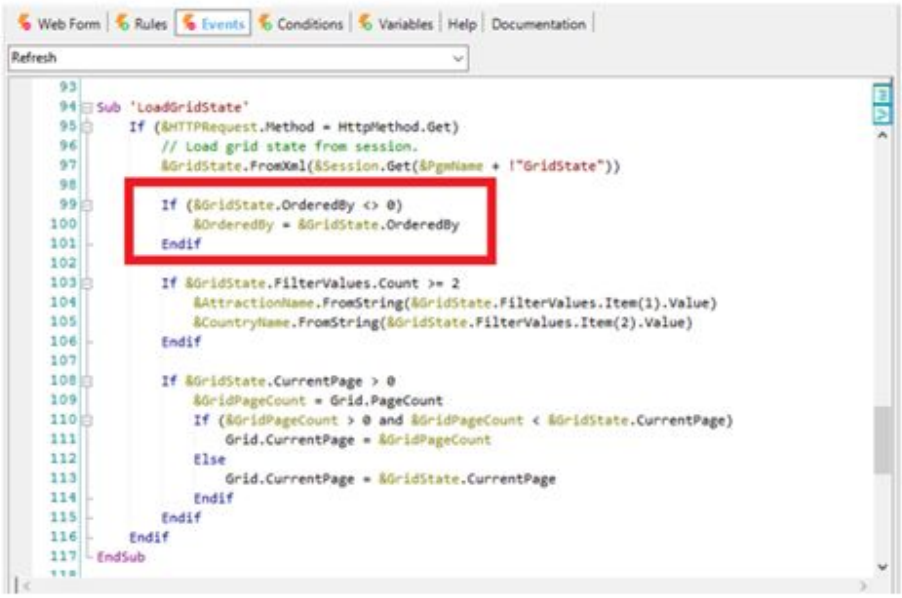
Luego vemos que se hacen varias verificaciones, mediante comandos IF.

```
89      Attraction(TrnMode.Insert, nullvalue(AttractionId))
90  EndEvent
91
92  /*** Subroutines used to load and save the grid state. ***/
93
94  Sub 'LoadGridState'
95      If (&HTTPRequest.Method = HttpMethod.Get)
96          // Load grid state from session.
97          &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))
98
99          If (&GridState.OrderedBy <> 0)
100             &OrderedBy = &GridState.OrderedBy
101         Endif
102
103         If &GridState.FilterValues.Count >= 2
104             &AttractionName.FromString(&GridState.FilterValues.Item(1).Value)
105             &CountryName.FromString(&GridState.FilterValues.Item(2).Value)
106         Endif
107
108         If &GridState.CurrentPage > 0
109             &GridPageCount = Grid.PageCount
110             If (&GridPageCount > 0 and &GridPageCount < &GridState.CurrentPage)
111                 Grid.CurrentPage = &GridPageCount
112             Else
113                 Grid.CurrentPage = &GridState.CurrentPage
114             Endif
115         Endif
116     Endif
117 EndSub
118
119 Sub 'SaveGridState'
120     &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))
121
```

En esta sección se verifica que el miembro OrderedBy de esta variable SDT sea diferente de 0, o sea, en otras palabras, verifica si hay algún orden guardado, para saber si es necesario recuperarlo o no.

Si es diferente de 0 recupera esa información guardada en una variable de nombre OrderedBy.

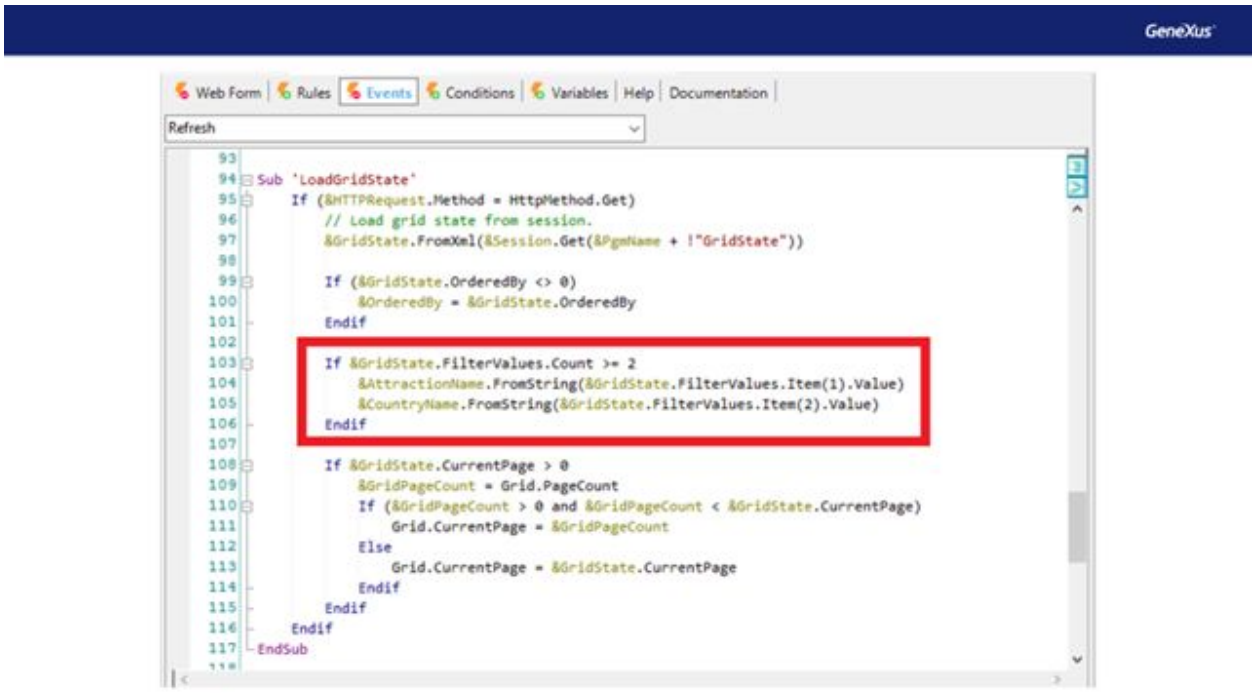
GeneXus



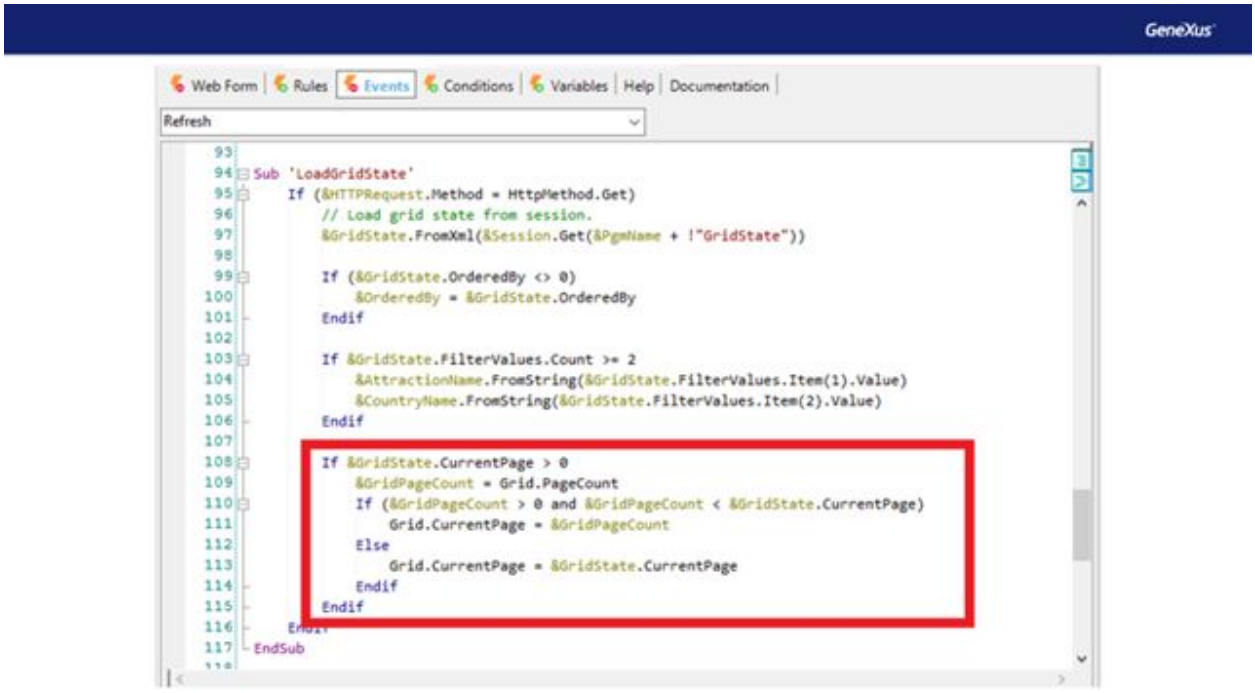
Luego, verifica si la cantidad de datos guardados en la colección filterValues es mayor o igual a dos. Este valor lo asigna ya que son dos los filtros que hay en el WorkWith. Comprueba que estén guardados estos datos para posteriormente recuperarlos.

En este bloque de código es donde se recuperará los valores de los filtros.

El filtro del nombre de la atracción será guardado en la variable AttractionName, mediante el método FromString. Lo mismo con el filtro del nombre de país, guardando el valor en la variable CountryName.

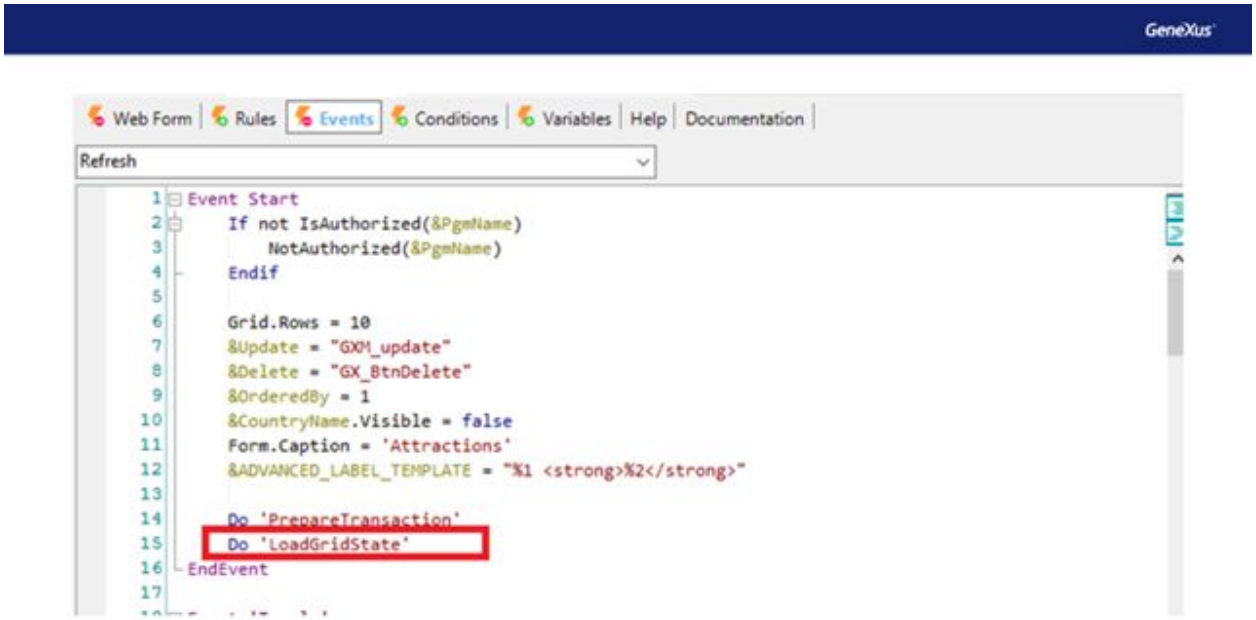


Lo último que se verifica es si el miembro CurrentPage, utilizado para guardar la página del grid, es mayor a 0. De ser así recupera dicho valor y lo asigna directamente a la propiedad CurrentPage del grid.



Ahora, ¿cuándo será el momento que se llamará a esta subrutina?

Como lo analizamos cuando hicimos nuestro propio Web Panel, el único momento posible será en el evento Start. Evento que se ejecuta una única vez cuando se carga el sitio por primera vez.



Otra diferencia que presenta este objeto con el creado por nosotros, la podemos apreciar en el evento Load del grid.

Para acceder a actualizar la información de algún registro del grid, tanto nuestro web Panel como el generado por el pattern, utilizan la variable &update, haciendo click sobre ella nos lleva a la pantalla correspondiente, que en este caso es a la transacción attraction en modo update.

Recordemos cómo lo implementamos en nuestro Web Panel.

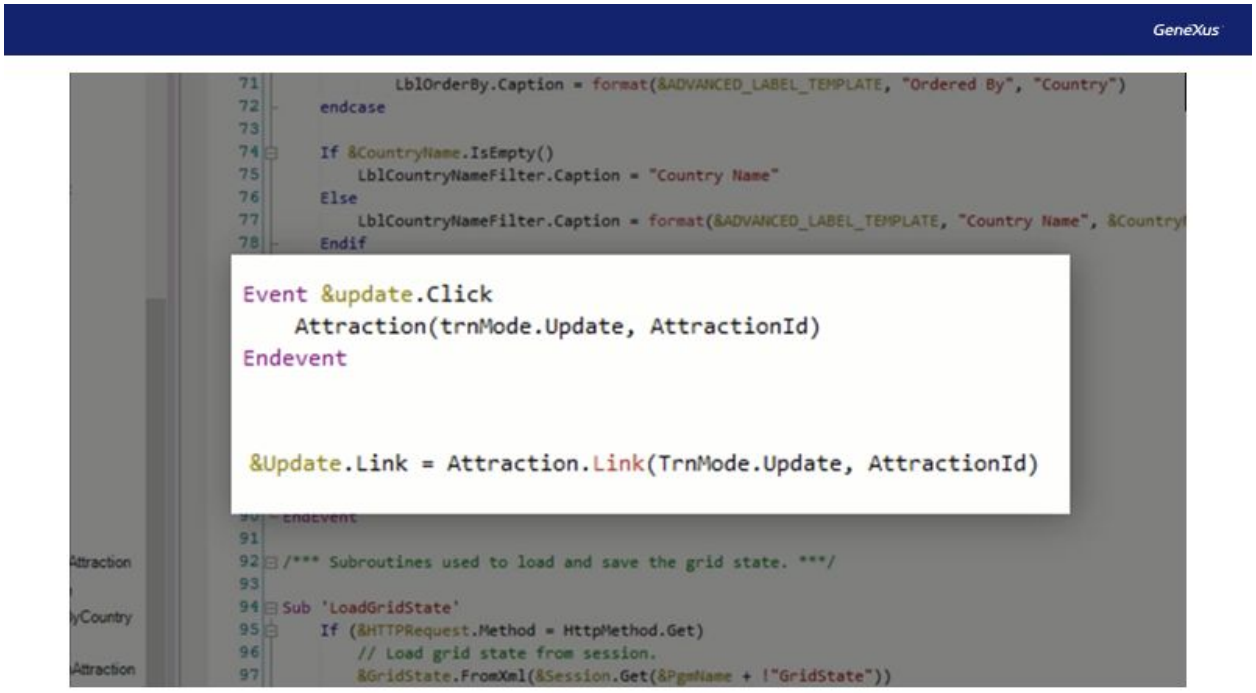
Lo hicimos dentro del evento &update.click, invocando directamente la transacción Attraction, pasándole por parámetro el modo que deseamos se ejecute, y el Id de la atracción.

El pattern, lo hace en el evento Grid.Load, el cual se ejecutará tantas veces como registros a ser cargados en el grid.

Observemos que el Pattern le aplicó la propiedad Link a la variable Update. Y luego le indica que ese valor será igual a la función Link() de la transacción Attraction, pasándole por parámetro el modo en el que se desea ejecutar esa transacción y el ID que la identifica, en este caso AttractionId. O sea, asocia la función Link a la propiedad link de la variable, teniendo como resultado que al hacer clic sobre esta variable se realiza el llamado al objeto web Attraction.

Aunque no lo estemos ejemplificando, lo mismo hace con la variable &delete, pasándole por parámetro el modo correspondiente.

En este caso, ambas maneras de implementarlo tendrán la misma funcionalidad.
Cuando hagamos click sobre la variable update, tanto en una como en otra opción, se llamará al objeto Attraction, indicando que queremos realizar una actualización, pasandole el Id del elemento seleccionado, para que se pueda a partir de esta clave cargar en pantalla todos los datos.



Como vemos, para la invocación a otros objetos web, se podrá utilizar tanto el parámetro y función Link como la invocación directa del objeto.

Si bien para este ejemplo podemos usar cualquiera de las dos opciones, hay diferencias entre estas.
Por ejemplo, si el objeto que queremos llamar es un procedimiento, sólo podremos hacerlo mediante la invocación directa con el nombre el objeto.

Ejemplo:

```
Event &ProcedureLink.Click
    Procedure()
EndEvent
```

Por otra parte, si por ejemplo queremos hacer referencia a una página HTML estática, deberemos utilizar la función Link.

Ejemplo:

```
Event Enter
    Link('http://www.genexus.com')
EndEvent
```

Ahora volvamos a la sección eventos de nuestro Web Panel.

Probemos y practiquemos el uso que acabamos de ver de las subrutinas, de la misma forma que las utiliza en el Work With.

De esta manera nos quedará un código más prolijo, y con posibilidad de en algún momento que podamos necesitarlo, reutilizar esas subrutinas en algún otro evento dentro del mismo objeto.

Crearemos primero la subrutina donde guardaremos los datos de los filtros.

Para esto utilizaremos el comando 'Sub' y le pondremos el mismo nombre que le asignó el Work With, 'SaveGridState'

Ahora, le pondremos dentro el código creado anteriormente que programamos en el evento &update.click, donde le asignamos los métodos set a la variable webSession, pasándole por parámetro como clave y valor cada variable.

Luego hacemos lo mismo, pero con la subrutina 'LoadGridState'. Donde recuperaremos los valores pasandole una clave, y ese valor se lo asignamos a cada variable de filtro. Esto lo teníamos ubicado hasta el momento en el evento Start.

Ahora solo nos restaría llamar esas subrutinas desde los eventos correspondientes.

En el caso de la subrutina 'SaveGridState', vamos a llamarla en el evento &update.Click, por los motivos que explicamos en el video anterior. Aunque vimos que si lo hacemos desde el evento Refresh, como lo genera el Work With automáticamente, también cumplirá nuestros requisitos.

Y ahora, ¿donde debemos ubicar el llamado a la subrutina 'LoadGridState'?

Como ya lo analizamos, será en el evento Start.

De esta manera hemos hecho un mix entre como lo habíamos implementado a nuestra manera en un principio, y como lo implementa automáticamente el Work With.

Incluso, si se quisiera, podríamos implementarlo también con un objeto SDT y una variable de este tipo de datos, de la misma forma que lo hace el Pattern.
No lo vamos a hacer en este ejemplo, pero sería una buena práctica que luego de este video lo intentes hacer tú mismo.

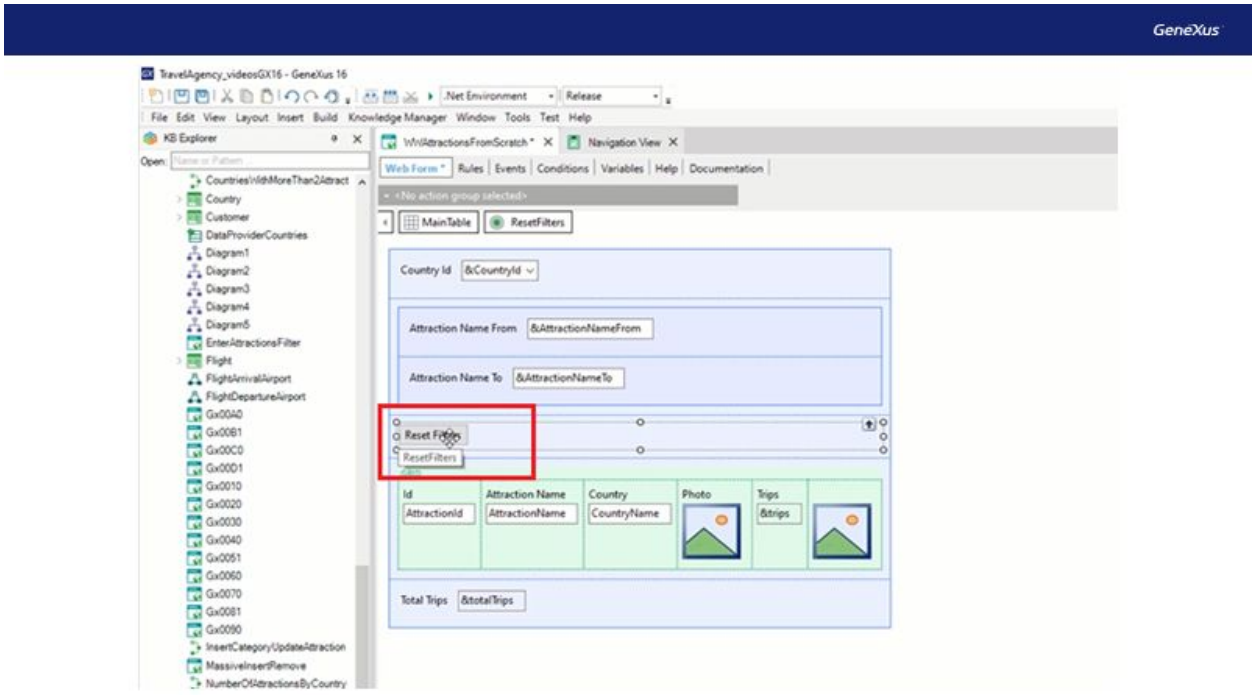
Que los filtros permanezcan entre ejecuciones de pantallas, puede ser ventajoso al momento que buscamos esta funcionalidad. Pero tal vez resulte engorroso cuando no queramos que se mantengan. Ya que se deberán eliminar uno a uno los filtros que tengamos ingresados.

¿Cómo podríamos hacer para de una manera más sencilla limpiar estos filtros?

Una de las opciones sería agregarle a nuestro Web Panel un botón que nos implemente esta funcionalidad.
Básicamente necesitaremos que este botón cumpla tres funciones:

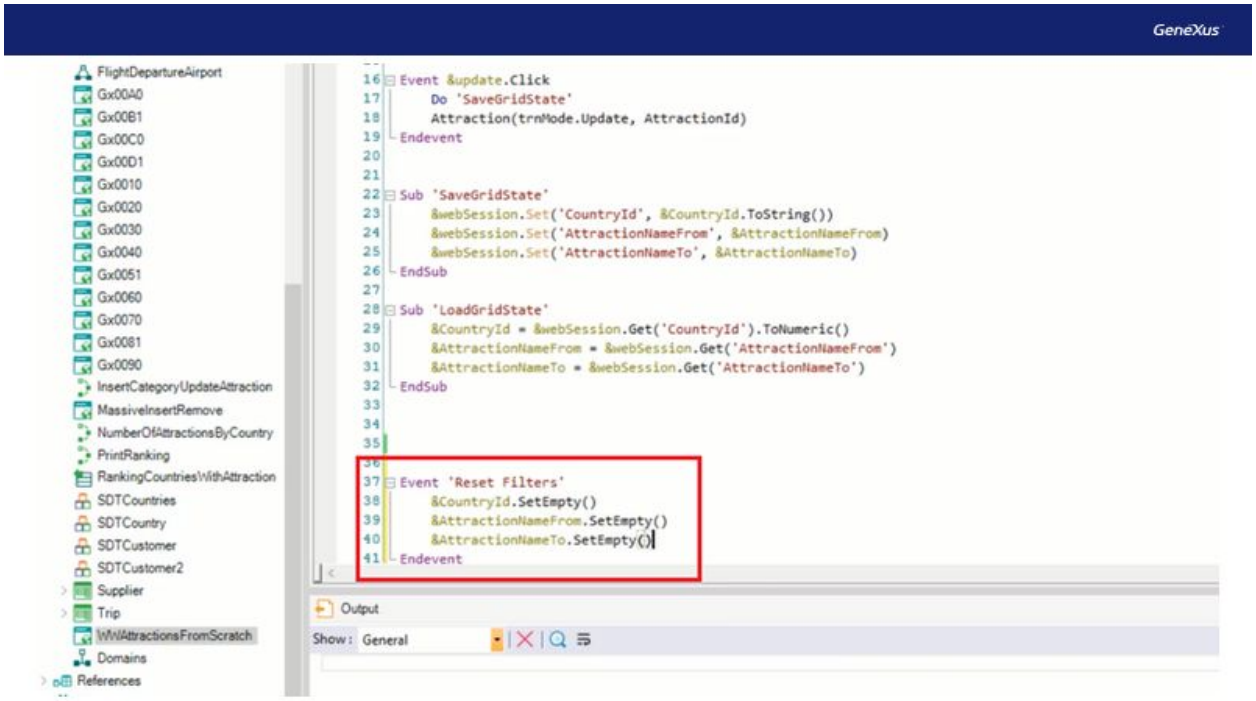
- Vaciar las variables que utilizamos para los filtros.
- Vaciar la variable del tipo Web Session.
- Refrescar la grilla

Para esto arrastraremos un control del tipo Button al Web Panel, y le ponemos como nombre del evento “Reset Filters”



Luego, le damos doble click en el botón, y nos llevará para programar el evento del mismo.

Como dijimos, la primera de las funcionalidades que necesitamos, será que las variables utilizadas para los filtros queden vacías.
Para esto a cada variable le aplicaremos el método SetEmpty(). De esta forma “vaciamos” cada una de ellas.



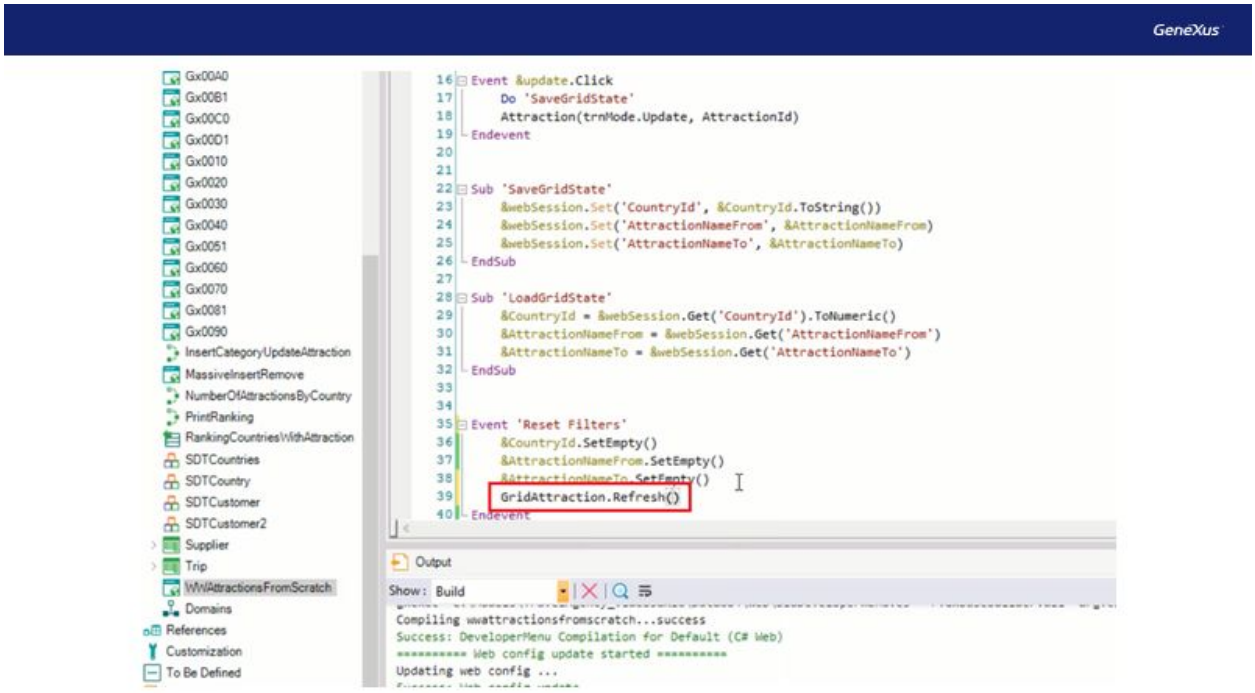
Probemos ejecutar ahora.

Ingresamos por ejemplo el País China, y filtraremos la atracciones de la A hasta la J.

Ahora hacemos click en el botón “Reset Filters”.

Vemos que los filtros quedan vacíos, pero la grilla no se actualiza.
Esto es porque no se le ha dado la orden a la grilla de actualizarse. ¿Cómo hacemos esto?

Deberemos agregarle dentro del evento del botón, el método Refresh() a la grilla que queramos se actualice, en este caso de nombre GridAttraction



Ahora sí, ejecutemos nuevamente y probemos ingresar valores en los filtros para luego dar click en el botón Reset Filters. Vemos que ahora funciona correctamente y actualiza la grilla como deseábamos.

De esta manera hemos visto cómo el Work With del Pattern, implementa la funcionalidad para guardar los valores de las variables que impactan en el grid. Como lo son, el orden aplicado a la grilla, los valores de los campos editables de filtro y el número de página correspondiente al grid.
Y comparamos esta solución con la hecha a nuestro modo, que planteamos en el primer video, explicando las diferencias entre ellas.
Por último hicimos modificaciones en nuestro WebPanel utilizando subrutinas, para explicar su funcionamiento y entender su uso, al igual que lo hace el WorkWith.

Los invitamos ingresar a nuestra Wiki para profundizar sobre los temas vistos:
https://wiki.genexus.com/