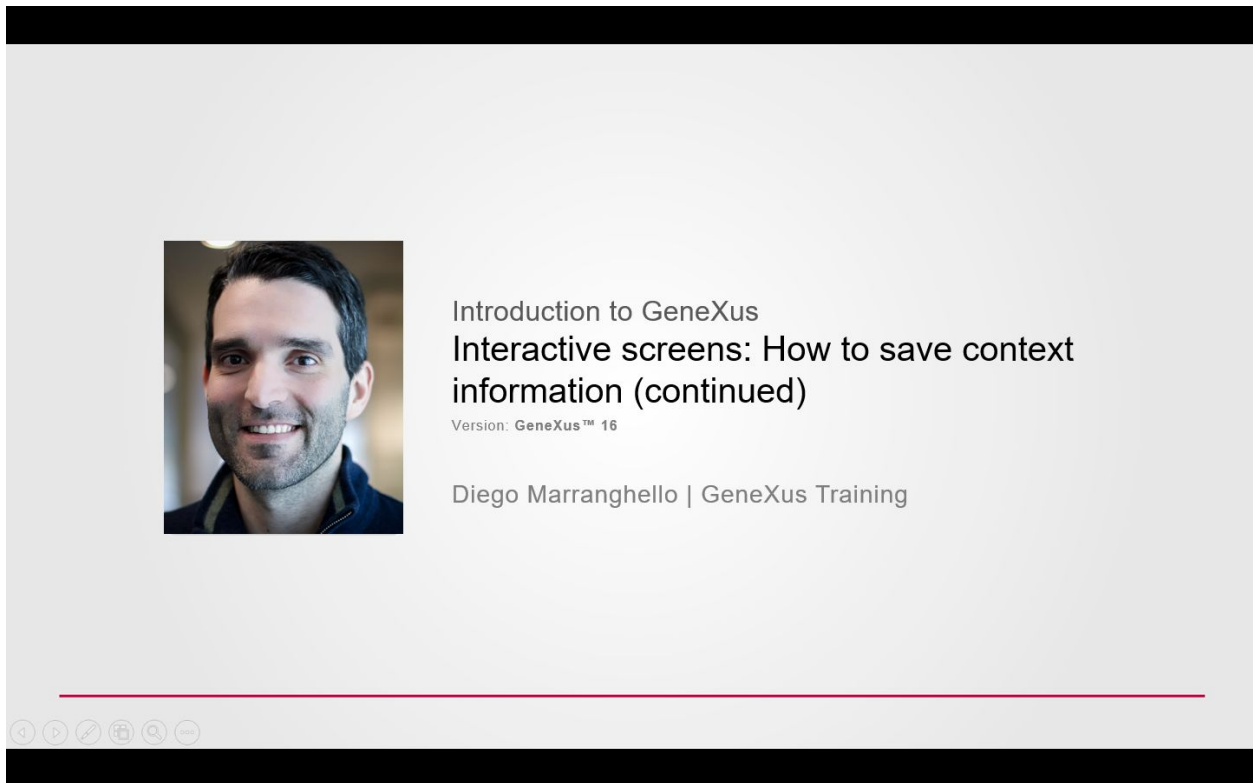


# Pantallas interactivas

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`.

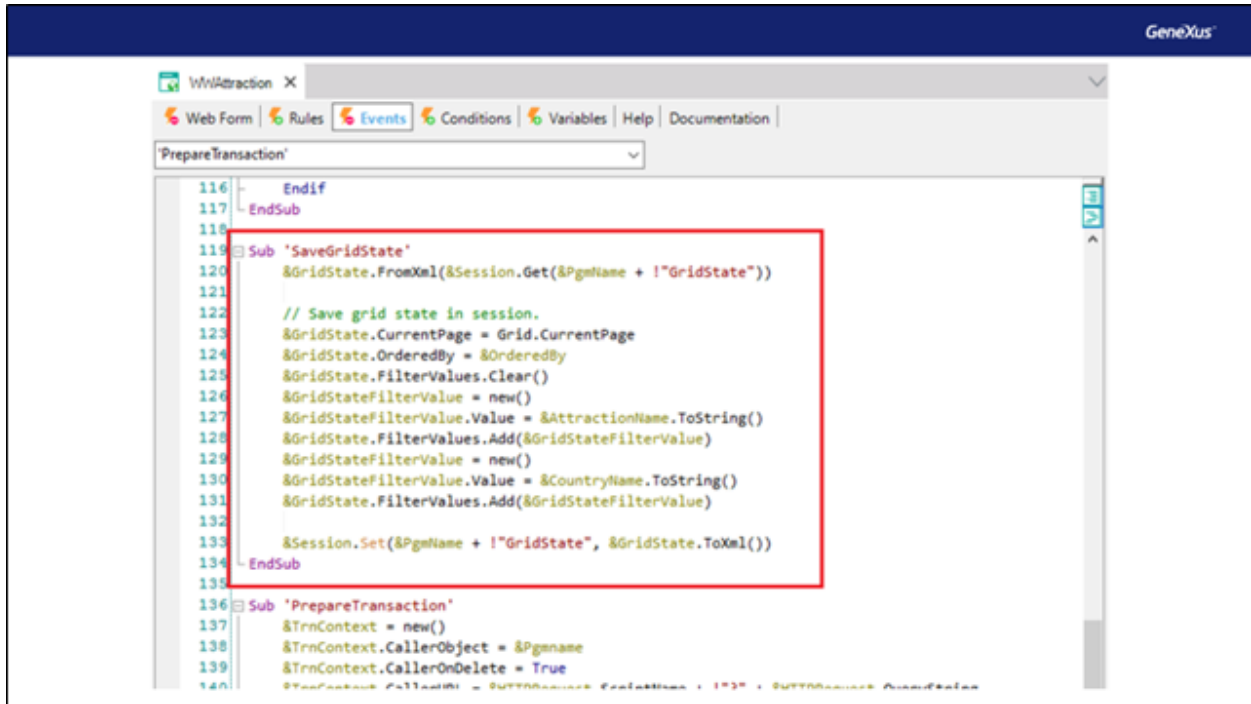
```
1
2 Event Load
3     &trips = Count(TripDate)
4     &totalTrips = &totalTrips + &trips
5 -Endevent
6 |
7 Event Refresh
8     &totalTrips = 0
9 -Endevent
10
11 Event Start
12     &update.FromImage(updateIcon)
13     &CountryId = &webSession.Get('CountryId').ToNumeric()
14     &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
15     &AttractionNameTo = &webSession.Get('AttractionNameTo')
16 -Endevent
17
18 Event &update.Click
19     &webSession.Set('CountryId', &CountryId.ToString())
20     &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
21     &webSession.Set('AttractionNameTo', &AttractionNameTo)
22     Attraction(trnMode.Update, AttractionId)
23 -Endevent
24
25
```

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"



The screenshot shows the GeneXus IDE interface. At the top, there's a menu bar with 'Web Form', 'Rules', 'Events', 'Conditions', 'Variables', 'Help', and 'Documentation'. Below the menu, a dropdown menu shows 'PrepareTransaction'. The main editor area displays code with line numbers from 116 to 140. A red box highlights a subroutine named 'SaveGridState' starting at line 119 and ending at line 134. The code inside the box includes comments and several method calls on objects like &GridState and &GridStateFilterValue. Below the highlighted section, another subroutine 'PrepareTransaction' is partially visible, starting at line 136.

```
116 Endif
117 EndSub
118
119 Sub 'SaveGridState'
120 &GridState.FromXml(&Session.Get(&PgName + !"GridState"))
121
122 // Save grid state in session.
123 &GridState.CurrentPage = Grid.CurrentPage
124 &GridState.OrderBy = &OrderedBy
125 &GridState.FilterValues.Clear()
126 &GridStateFilterValue = new()
127 &GridStateFilterValue.Value = &AttractionName.ToString()
128 &GridState.FilterValues.Add(&GridStateFilterValue)
129 &GridStateFilterValue = new()
130 &GridStateFilterValue.Value = &CountryName.ToString()
131 &GridState.FilterValues.Add(&GridStateFilterValue)
132
133 &Session.Set(&PgName + !"GridState", &GridState.ToXml())
134 EndSub
135
136 Sub 'PrepareTransaction'
137 &TrnContext = new()
138 &TrnContext.CallerObject = &PgName
139 &TrnContext.CallerOnDelete = True
140 &TrnContext.CallerOnDelete = &PgName + !"GridState" + &PgName
```

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.

```

// Save grid state in session.
&GridState.CurrentPage = Grid.CurrentPage
&GridState.OrderBy = &OrderBy
&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.

The screenshot shows the 'Variables' window in GeneXus. The window title is 'WWAttraction X'. The 'Variables' tab is selected, showing a list of variables. The variables are organized into a tree structure under '& Variables' and '& Standard Variables'. The variables 'GridState' and 'GridStateFilterValue' are highlighted with a red box.

Name	Type	Is Collection	Description
& Variables			
& Standard Variables			
• ADVANCED_LABEL_TEMPLATE	Character(20)	<input type="checkbox"/>	ADVANCED_LABEL_TEMPLATE
• AttractionName	Attribute:AttractionName	<input type="checkbox"/>	Attraction Name
• CountryName	Attribute:CountryName	<input type="checkbox"/>	Country Name
• Delete	Character(20)	<input type="checkbox"/>	Delete
• GridPageCount	Numeric(8.0-)	<input type="checkbox"/>	Grid Page Count
• GridState	GridState	<input type="checkbox"/>	Grid State
• GridStateFilterValue	GridState.FilterValue	<input type="checkbox"/>	Grid State Filter Value
• HTTPRequest	HttpRequest	<input type="checkbox"/>	HttpRequest
• IsAuthorized	Boolean	<input type="checkbox"/>	Is Authorized
• OrderBy	Numeric(4.0)	<input type="checkbox"/>	Ordered By
• Session	WebSession	<input type="checkbox"/>	Session
• TrnContext	TransactionContext	<input type="checkbox"/>	Trn Context
• TrnContextAtt	TransactionContext.Attribute	<input type="checkbox"/>	Trn Context Att
• Update	Character(20)	<input type="checkbox"/>	Update

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.



The screenshot shows the 'Structure' window in GeneXus. The window title is 'GridState X'. The 'Structure' tab is active. The table below represents the structure shown in the window:

Name	Type	Description	Is Collection
GridState		Grid State	<input type="checkbox"/>
CurrentPage	Numeric(4.0)	Current Page	<input type="checkbox"/>
OrderedBy	Numeric(4.0)	Ordered By	<input type="checkbox"/>
HidingSearch	Numeric(1.0)	Hiding Search	<input type="checkbox"/>
FilterValues		Filter Values	<input checked="" type="checkbox"/>
FilterValue			<input type="checkbox"/>
Value	VarChar(100)	Value	<input type="checkbox"/>

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.

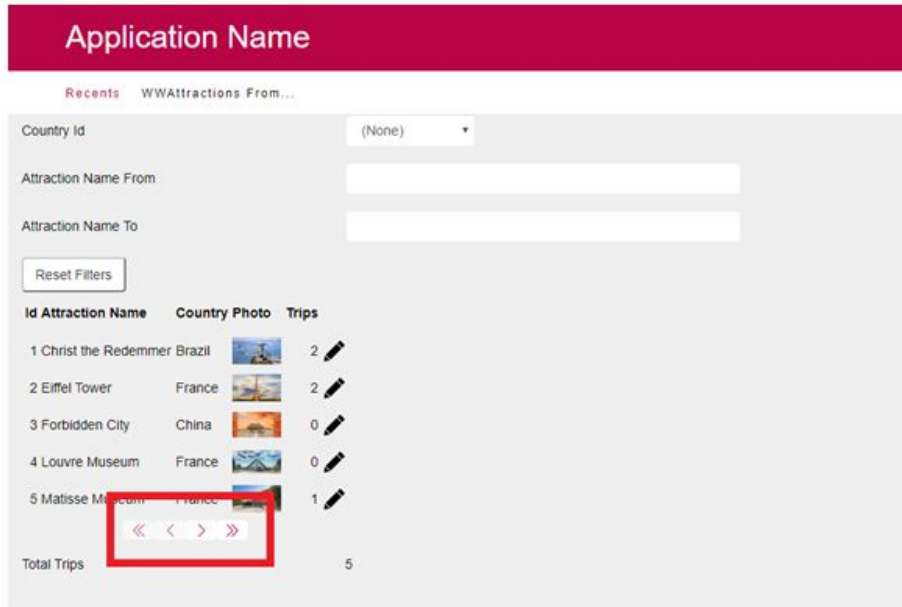
```

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
122     // Save grid state in session.
123     &GridState.CurrentPage = Grid.CurrentPage
124     &GridState.OrderBy = &OrderedBy
125     &GridState.FilterValues.Clear()
126     &GridStateFilterValue = new()
127     &GridStateFilterValue.Value = &AttractionName.ToString()
128     &GridState.FilterValues.Add(&GridStateFilterValue)
129     &GridStateFilterValue = new()
130     &GridStateFilterValue.Value = &CountryName.ToString()
131     &GridState.FilterValues.Add(&GridStateFilterValue)
132
133     &Session.Set(&PgmName + !"GridState", &GridState.ToXml())
134 EndSub
135
136 Sub 'PrepareTransaction'
137     &TrnContext = new()
138     &TrnContext.CallerObject = &Pgname
139     &TrnContext.CallerOnDelete = True
140     &TrnContext.CallerURL = &HTTPRequest.ScriptName + "?" + &HTTPRequest.QueryString

```

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 `OrderBy` del SDT `GridState`.

```

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
122     // Save grid state in session.
123     &GridState.CurrentPage = Grid.CurrentPage
124     &GridState.OrderBy = &OrderBy
125     &GridState.FilterValues.Clear()
126     &GridStateFilterValue = new()
127     &GridStateFilterValue.Value = &AttractionName.ToString()
128     &GridState.FilterValues.Add(&GridStateFilterValue)
129     &GridStateFilterValue = new()
130     &GridStateFilterValue.Value = &CountryName.ToString()
131     &GridState.FilterValues.Add(&GridStateFilterValue)
132
133     &Session.Set(&PgmName + !"GridState", &GridState.ToXml())
134 EndSub
135
136 Sub 'PrepareTransaction'
137     &TrnContext = new()
138     &TrnContext.CallerObject = &PgmName
139     &TrnContext.CallerOnDelete = True

```

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.

```
113         Grid.CurrentPage = &GridState.CurrentPage
114     Endif
115 Endif
116 Endif
117 EndSub
118
119 Sub 'SaveGridState'
120     &GridState.FromXml(&Session.Get(&PgName + !"GridState"))
121
122     // Save grid state in session.
123     &GridState.CurrentPage = Grid.CurrentPage
124     &GridState.OrderBy = &OrderBy
125     &GridState.FilterValues.Clear()
126     &GridStateFilterValue = new()
127     &GridStateFilterValue.Value = &AttractionName.ToString()
128     &GridState.FilterValues.Add(&GridStateFilterValue)
129     &GridStateFilterValue = new()
130     &GridStateFilterValue.Value = &CountryName.ToString()
131     &GridState.FilterValues.Add(&GridStateFilterValue)
132
133     &Session.Set(&PgName + !"GridState", &GridState.ToXml())
134 EndSub
135
136 Sub 'PrepareTransaction'
137     &TrnContext = new()
138     &TrnContext.CallerObject = &PgName
139     &TrnContext.CallerOnDelete = True
```

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.



```

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
122     // Save grid state in session.
123     &GridState.CurrentPage = Grid.CurrentPage
124     &GridState.OrderBy = &OrderBy
125     &GridState.FilterValues.Clear()
126     &GridStateFilterValue = new()
127     &GridStateFilterValue.Value = &AttractionName.ToString()
128     &GridState.FilterValues.Add(&GridStateFilterValue)
129     &GridStateFilterValue = new()
130     &GridStateFilterValue.Value = &CountryName.ToString()
131     &GridState.FilterValues.Add(&GridStateFilterValue)
132
133     &Session.Set(&PgmName + !"GridState", &GridState.ToXml())
134 EndSub
135
136 Sub 'PrepareTransaction'
137     &TrnContext = new()
138     &TrnContext.CallerObject = &Pgmname
139     &TrnContext.CallerOnDelete = True
140     &TrnContext.CallerURI = &HTTPRequest.ScriptName + !"?" + &HTTPRequest.QueryString

```

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.

```

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
122     // Save grid state in session.
123     &GridState.CurrentPage = Grid.CurrentPage
124     &GridState.OrderBy = &OrderBy
125     &GridState.FilterValues.Clear()
126     &GridState.FilterValue = new()
127     &GridState.FilterValue.Value = &AttractionName.ToString()
128     &GridState.FilterValues.Add(&GridState.FilterValue)
129     &GridState.FilterValue = new()
130     &GridState.FilterValue.Value = &CountryName.ToString()
131     &GridState.FilterValues.Add(&GridState.FilterValue)
132
133     &Session.Set(&PgmName + !"GridState", &GridState.ToXml())
134 EndSub
135
136 Sub 'PrepareTransaction'
137     &TrnContext = new()
138     &TrnContext.CallerObject = &Pgmname
139     &TrnContext.CallerOnDelete = True
140     &TrnContext.CallerURL = &HttpRequest.ScriptName + !"?" + &HttpRequest.QueryString

```

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.

```

59     CountryNameFilterContainer.Class = ThemeClass:AdvancedContainerItem
60     &CountryName.Visible = false
61   endif
62 EndEvent
63
64 Event Refresh
65   Do 'SaveGridState'
66
67   do case
68     case &OrderBy = 1
69       LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Name")
70     case &OrderBy = 2
71       LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Country")
72     endcase
73
74   If &CountryName.IsEmpty()
75     LblCountryNameFilter.Caption = "Country Name"
76   Else
77     LblCountryNameFilter.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Country Name", &CountryName)
78   Endif
79 EndEvent
80
81 Event Grid.Load
82   &Update.Link = Attraction.Link(TrnMode.Update, AttractionId)
83   &Delete.Link = Attraction.Link(TrnMode.Delete, AttractionId)
84   AttractionName.Link = ViewAttraction.Link(AttractionId, "")
85   CountryName.Link = ViewCountry.Link(CountryId, "")
86 EndEvent
87

```

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 guardarán 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.



```
93 Refresh
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.OrderBy <> 0)
100       &OrderBy = &GridState.OrderBy
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
```

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(&PgName + !"GridState"))
98
99      If (&GridState.OrderBy <> 0)
100         &OrderBy = &GridState.OrderBy
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(&PgName + !"GridState"))
121
122

```

En esta sección se verifica que el miembro `OrderBy` 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 `OrderBy`.

```

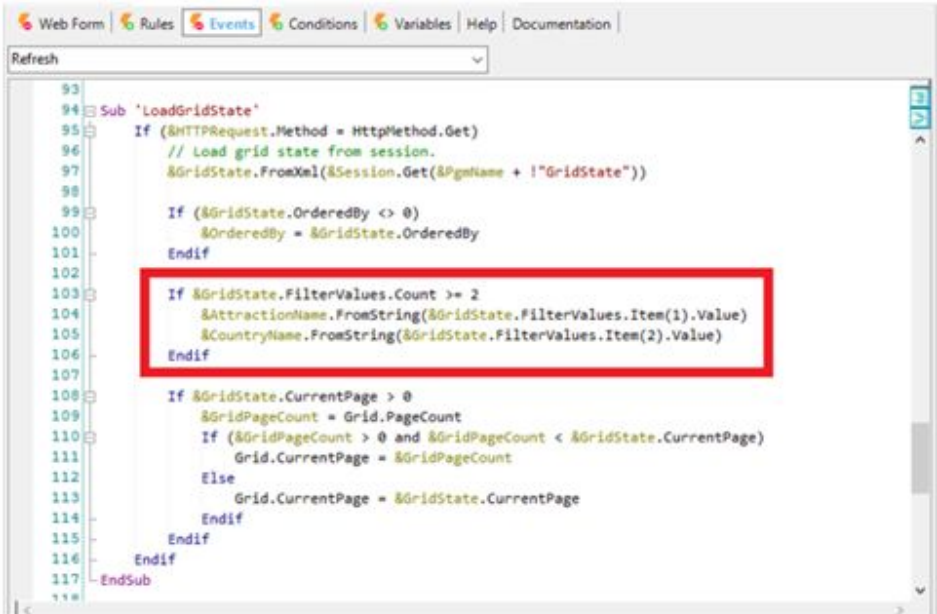
93
94 Sub 'LoadGridState'
95 If (&HttpRequest.Method = HttpMethod.Get)
96     // Load grid state from session.
97     &GridState.FromXml(&Session.Get(&PgName + !"GridState"))
98
99     If (&GridState.OrderBy <> 0)
100         &OrderBy = &GridState.OrderBy
101     Endif
102
103     If &GridState.FilterValues.Count >= 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

```

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.

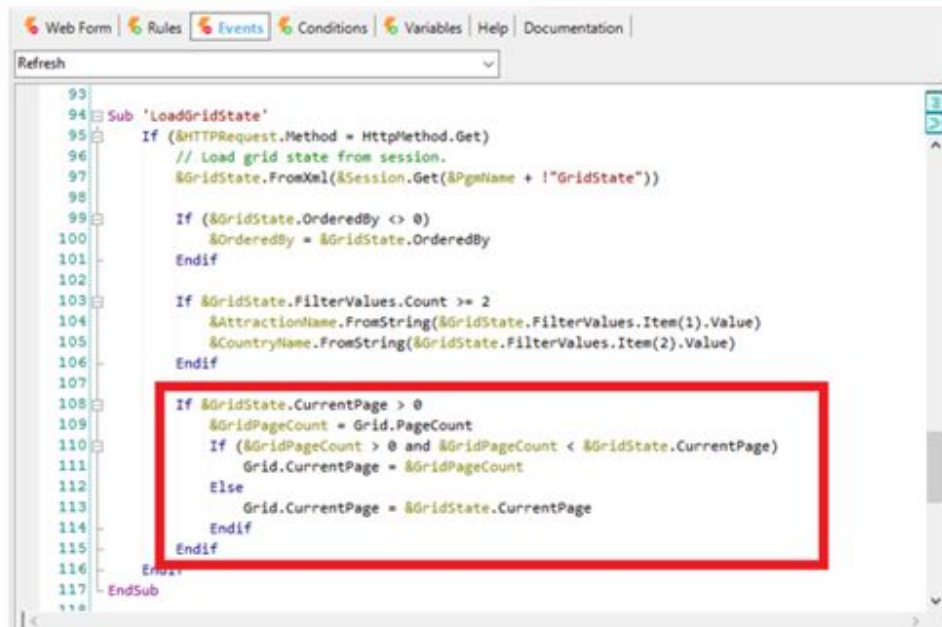


The screenshot shows the GeneXus IDE interface with the 'Events' tab selected. The code editor displays the following code for the 'Refresh' event:

```
93  
94 Sub 'LoadGridState'  
95   If (&HttpRequest.Method = HttpMethod.Get)  
96     // Load grid state from session.  
97     &GridState.FromXml(&Session.Get(&PgName + !"GridState"))  
98  
99     If (&GridState.OrderBy <> 0)  
100       &OrderBy = &GridState.OrderBy  
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
```

The code block between lines 103 and 106 is highlighted with a red rectangular box.

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.



```
93  
94 Sub 'LoadGridState'  
95   If (&HttpRequest.Method = HttpMethod.Get)  
96     // Load grid state from session.  
97     &GridState.FromXml(&Session.Get(&PgName + !"GridState"))  
98  
99     If (&GridState.OrderBy <> 0)  
100       &OrderBy = &GridState.OrderBy  
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 EndSub  
117  
118
```

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.

```

Web Form | Rules | Events | Conditions | Variables | Help | Documentation |
Refresh
1 Event Start
2   If not IsAuthorized(&PgName)
3     NotAuthorized(&PgName)
4   Endif
5
6   Grid.Rows = 10
7   &Update = "GX!_update"
8   &Delete = "GX_BtnDelete"
9   &OrderBy = 1
10  &CountryName.Visible = false
11  Form.Caption = 'Attractions'
12  &ADVANCED_LABEL_TEMPLATE = "%1 <strong>%2</strong>"
13
14  Do 'PrepareTransaction'
15  Do 'LoadGridState'
16 EndEvent
17

```

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

```

80 -----
81 Event Grid.Load
82   &Update.Link = Attraction.Link(TrnMode.Update, AttractionId)
83   &Delete.Link = Attraction.Link(TrnMode.Delete, AttractionId)
84   AttractionName.Link = ViewAttraction.Link(AttractionId, "")
85   CountryName.Link = ViewCountry.Link(CountryId, "")
86 EndEvent
87

```



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.

GeneXus

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
    Attraction(trnMode.Update, AttractionId)
Endevent
```

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.

```

71         LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Country")
72     endcase
73
74     If &CountryName.IsEmpty()
75         LblCountryNameFilter.Caption = "Country Name"
76     Else
77         LblCountryNameFilter.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Country Name", &Country
78     Endif
79 EndEvent
80
81 Event Grid.Load
82     &Update.Link = Attraction.Link(TrnMode.Update, AttractionId)
83     &Delete.Link = Attraction.Link(TrnMode.Delete, AttractionId)
84     AttractionName.Link = ViewAttraction.Link(AttractionId, "")
85     CountryName.Link = ViewCountry.Link(CountryId, "")
86 EndEvent
87
88 Event 'DoInsert'
89     Attraction(TrnMode.Insert, nullvalue(AttractionId))
90 EndEvent
91
Attraction
92 /** Subroutines used to load and save the grid state. ***/
93
yCountry
94 Sub 'LoadGridState'
Attraction
95     If (&HTTPRequest.Method = HttpMethod.Get)
96         // Load grid state from session.
97         &GridState.FromXml(&Session.Get(&PgName + !"GridState"))

```

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, pasándole el Id del elemento seleccionado, para que se pueda a partir de esta clave cargar en pantalla todos los datos.

```

71     LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Country")
72     endcase
73
74     If &CountryName.IsEmpty()
75         LblCountryNameFilter.Caption = "Country Name"
76     Else
77         LblCountryNameFilter.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Country Name", &CountryName)
78     Endif

```

```

Event &update.Click
    Attraction(trnMode.Update, AttractionId)
EndEvent

&Update.Link = Attraction.Link(TrnMode.Update, AttractionId)

```

```

90     EndEvent
91
Attraction
92     /** Subroutines used to load and save the grid state. **/
93
yCountry
94     Sub 'LoadGridState'
95         If (&HttpRequest.Method = HttpMethod.Get)
96             // Load grid state from session.
Attraction
97             &GridState.FromXel(&Session.Get(&PageName + !"GridState"))

```

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.



```
2 | Event Load
3 |     &trips = Count(TripDate)
4 |     &totalTrips = &totalTrips + &trips
5 | Endevent
6 |
7 | Event Refresh
8 |     &totalTrips = 0
9 | Endevent
10 |
11 | Event Start
12 |     &update.FromImage(updateIcon)
13 |     &CountryId = &webSession.Get('CountryId').ToNumeric()
14 |     &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
15 |     &AttractionNameTo = &webSession.Get('AttractionNameTo')
16 | Endevent
17 |
18 | Event &update.Click
19 |     Attraction(trnMode.Update, AttractionId)
20 | Endevent
21 |
22 | Sub 'SaveGridState'
23 |     &webSession.Set('CountryId', &CountryId.ToString())
24 |     &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
25 |     &webSession.Set('AttractionNameTo', &AttractionNameTo)
26 | EndSub
27 |
28 |
29 |
30 |
31 |
32 |
33 |
```

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

```
2 | Event Load
3 |     &trips = Count(TripDate)
4 |     &totalTrips = &totalTrips + &trips
5 | -Endevent
6 |
7 | Event Refresh
8 |     &totalTrips = 0
9 | -Endevent
10 |
11 | Event Start
12 |     &update.FromImage(updateIcon)
13 |
14 | -Endevent
15 |
16 | Event &update.Click
17 |     Attraction(trnMode.Update, AttractionId)
18 | -Endevent
19 |
20 |
21 | Sub 'SaveGridState'
22 |     &webSession.Set('CountryId', &CountryId.ToString())
23 |     &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
24 |     &webSession.Set('AttractionNameTo', &AttractionNameTo)
25 | -EndSub
26 |
27 | Sub 'LoadGridState'
28 |     &CountryId = &webSession.Get('CountryId').ToNumeric()
29 |     &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
30 |     &AttractionNameTo = &webSession.Get('AttractionNameTo')
31 | -EndSub
32 |
33 |
```

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.

```

5  | Endevent
6  |
7  | Event Refresh
8  |     &totalTrips = 0
9  | Endevent
10 |
11 | Event Start
12 |     &update.FromImage(updateIcon)
13 |     Do 'LoadGridState'
14 | Endevent
15 |
16 | Event &update.Click
17 |     Do 'SaveGridState'
18 |     Attraction(trnNode.Update, AttractionId)
19 | Endevent
20 |
21 |
22 | Sub 'SaveGridState'
23 |     &webSession.Set('CountryId', &CountryId.ToString())
24 |     &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
25 |     &webSession.Set('AttractionNameTo', &AttractionNameTo)
26 | EndSub
27 |
28 | Sub 'LoadGridState'
29 |     &CountryId = &webSession.Get('CountryId').ToNumeric()
30 |     &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
31 |     &AttractionNameTo = &webSession.Get('AttractionNameTo')
32 | EndSub
33 |

```

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.

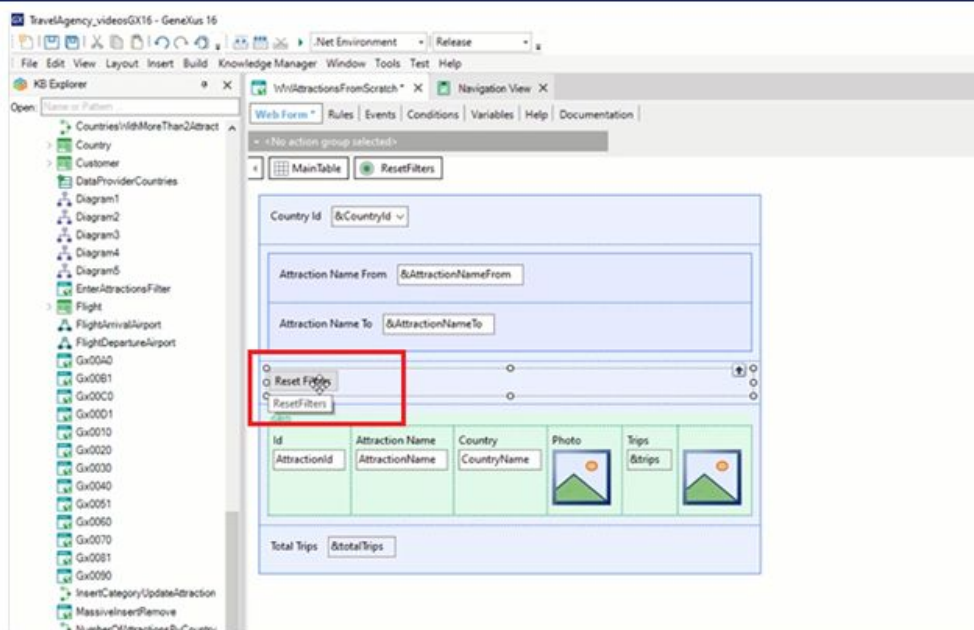
¿Cómo podríamos hacer para de una manera más sencilla limpiar estos filtros?

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.

```

16 Event &update.Click
17   Do 'SaveGridState'
18     Attraction(trnMode.Update, AttractionId)
19 EndEvent
20
21
22 Sub 'SaveGridState'
23   &webSession.Set('CountryId', &CountryId.ToString())
24   &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
25   &webSession.Set('AttractionNameTo', &AttractionNameTo)
26 EndSub
27
28 Sub 'LoadGridState'
29   &CountryId = &webSession.Get('CountryId').ToNumeric()
30   &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
31   &AttractionNameTo = &webSession.Get('AttractionNameTo')
32 EndSub
33
34
35
36
37 Event 'Reset Filters'
38   &CountryId.SetEmpty()
39   &AttractionNameFrom.SetEmpty()
40   &AttractionNameTo.SetEmpty()
41 EndEvent

```

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 queremos se actualice, en este caso de nombre GridAttraction



```

16 Event &update.Click
17 Do 'SaveGridState'
18 Attraction(trnMode.Update, AttractionId)
19 EndEvent
20
21
22 Sub 'SaveGridState'
23 &webSession.Set('CountryId', &CountryId.ToString())
24 &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
25 &webSession.Set('AttractionNameTo', &AttractionNameTo)
26 EndSub
27
28 Sub 'LoadGridState'
29 &CountryId = &webSession.Get('CountryId').ToNumeric()
30 &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
31 &AttractionNameTo = &webSession.Get('AttractionNameTo')
32 EndSub
33
34
35 Event 'Reset Filters'
36 &CountryId.SetEmpty()
37 &AttractionNameFrom.SetEmpty()
38 &AttractionNameTo.SetEmpty()
39 GridAttraction.Refresh()
40 EndEvent

```

Output

Show: Build

Compiling wattractionsfromscratch...success  
 Success: DeveloperMenu Compilation for Default (C# Web)  
 \*\*\*\*\* Web config update started \*\*\*\*\*  
 Updating web config ...

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/>