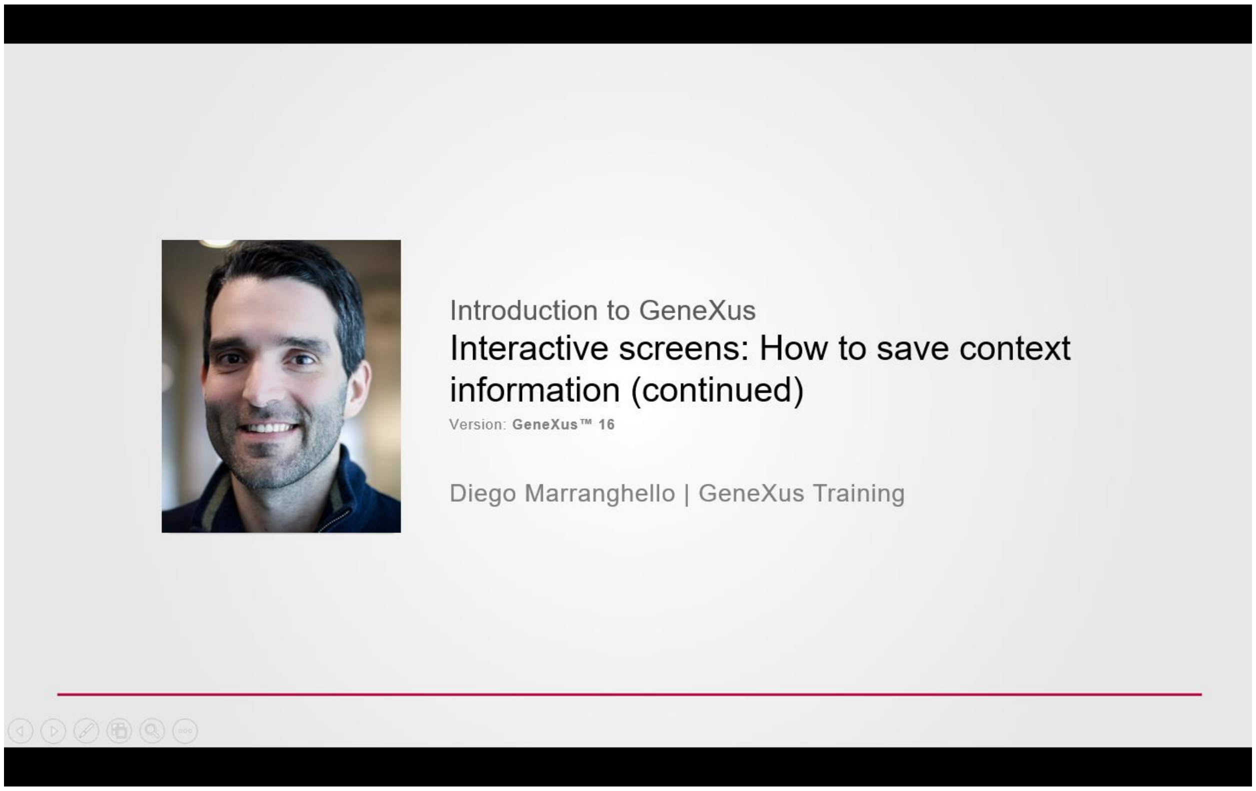
Interactive Screens

How to Save Context Information (continued)


Introduction to GeneXus

Interactive screens: How to save context information (continued)

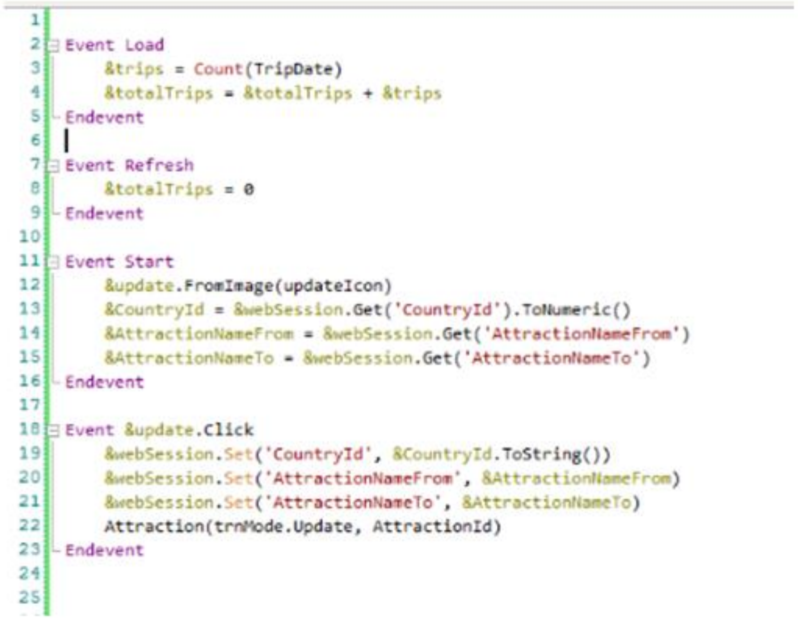
Version: GeneXus™ 16

Diego Marranghello | GeneXus Training

The previous video showed how to keep data in memory –preventing it from being lost after calling another object– and then return to the first one.

To do this, we created a WebSession-type variable and saved in it the data we were interested in, which in this case included the values of the three available filters.

We decided to save this data in the event &Update.Click using the SET method of the session variable, to later retrieve them in the Start event through the GET method of the WebSession variable.

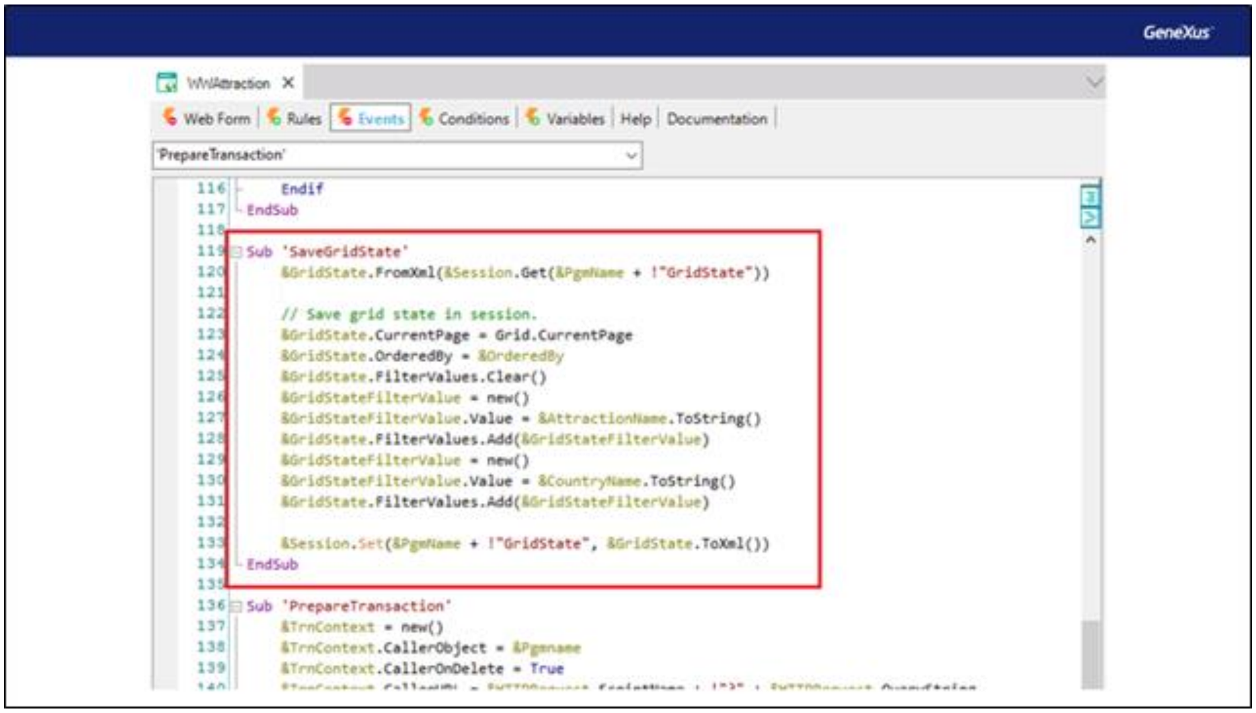


```
Event Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
    &totalTrips = 0
Endevent

Event Start
    &update.FromImage(updateIcon)
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
Endevent

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

This was a solution made in our own way, and as promised in the previous video, let's now see how the Work With Pattern of the Attraction transaction does it automatically.

Remember that in this Work With automatically generated by GeneXus, when you enter values in the filters and then update some record from the Update action, the values are kept when retrieved. To see how it’s done, we access the WWAttraction object and then its events section.

Note that a subroutine called “SaveGridState” has been generated.



```
    Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

    // 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())
EndSub

Sub 'PrepareTransaction'
    &TrnContext = new()
    &TrnContext.CallerObject = &Pgmname
    &TrnContext.CallerOnDelete = True
```

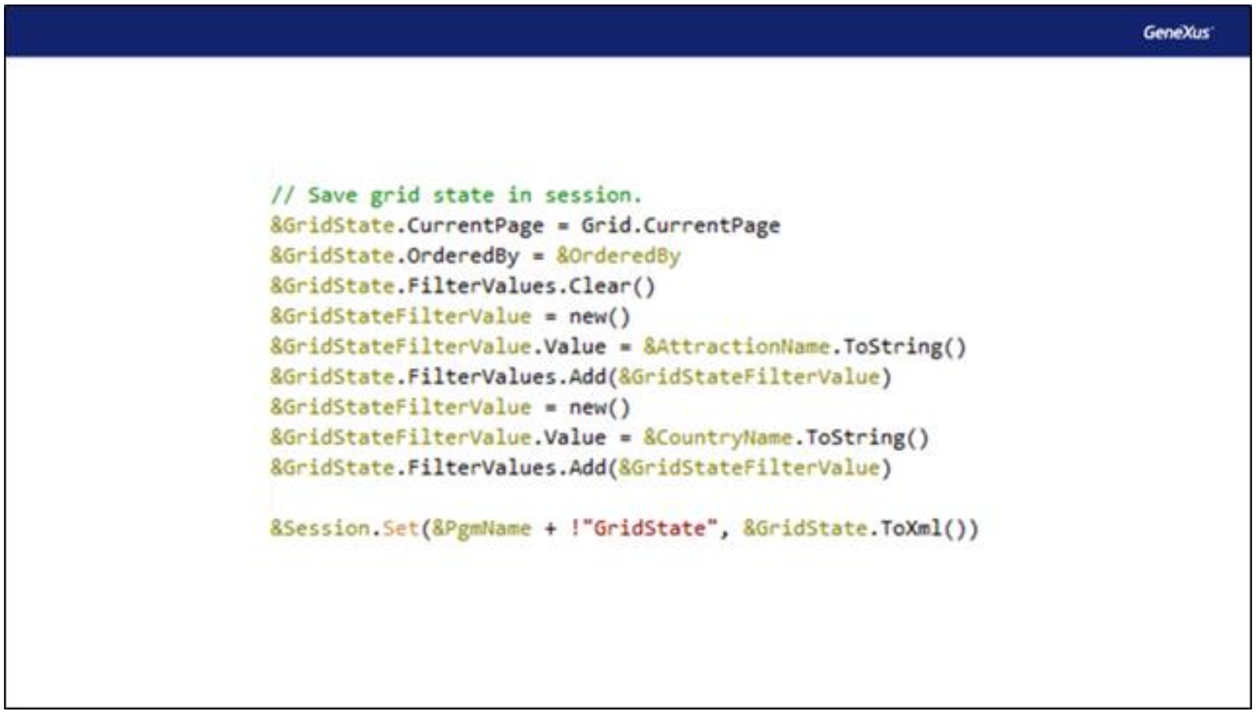
A subroutine is a block of code which is identified by a name and can then be run within the same object.

Subroutines are defined using the Sub command.

One of the advantages of subroutines is that they make the code clearer and easier to read.

Another great advantage is that they allow reusing the block of code declared within it. This way, if you need to execute the same block of code from several places in the object, you write it only once and it can be invoked from several locations using the DO command; we'll see this in a moment.

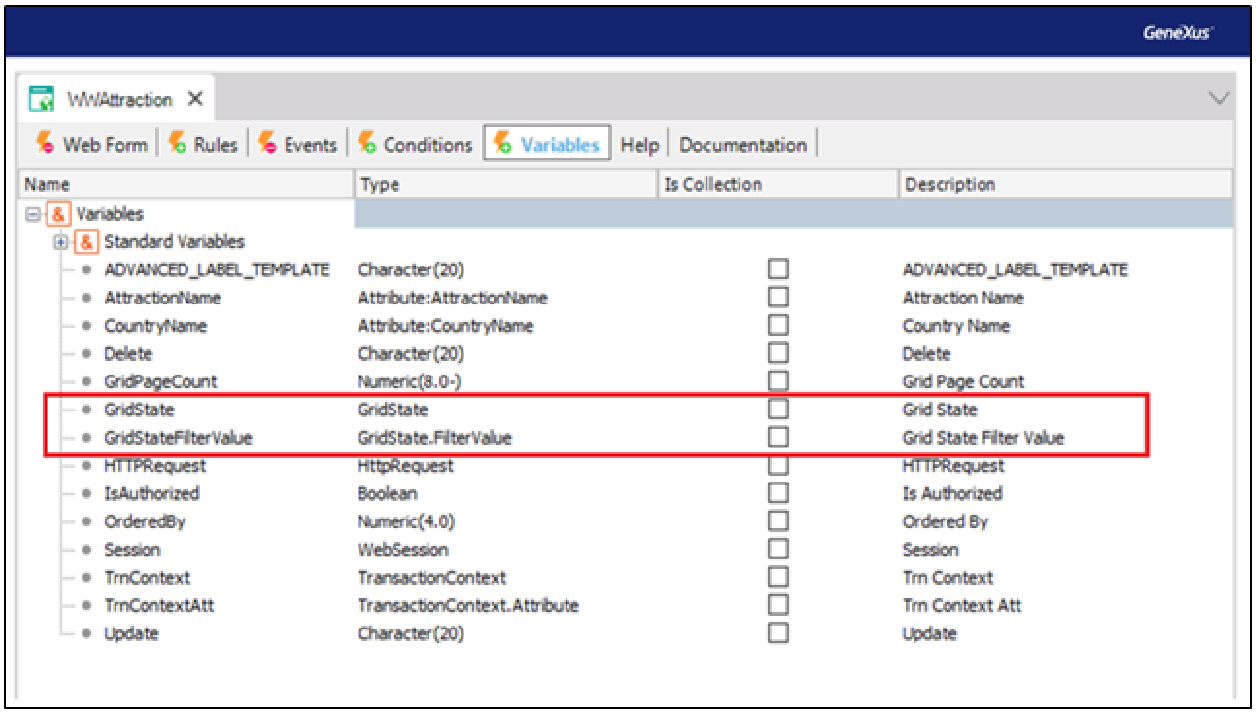
Let's focus on the next section of this subroutine code, which is what we are interested in now.



```
// 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())
```

First of all, note that a variable called GridState and another one called GridStateFilterValue have been created.

Note how they have been declared from the variables section.



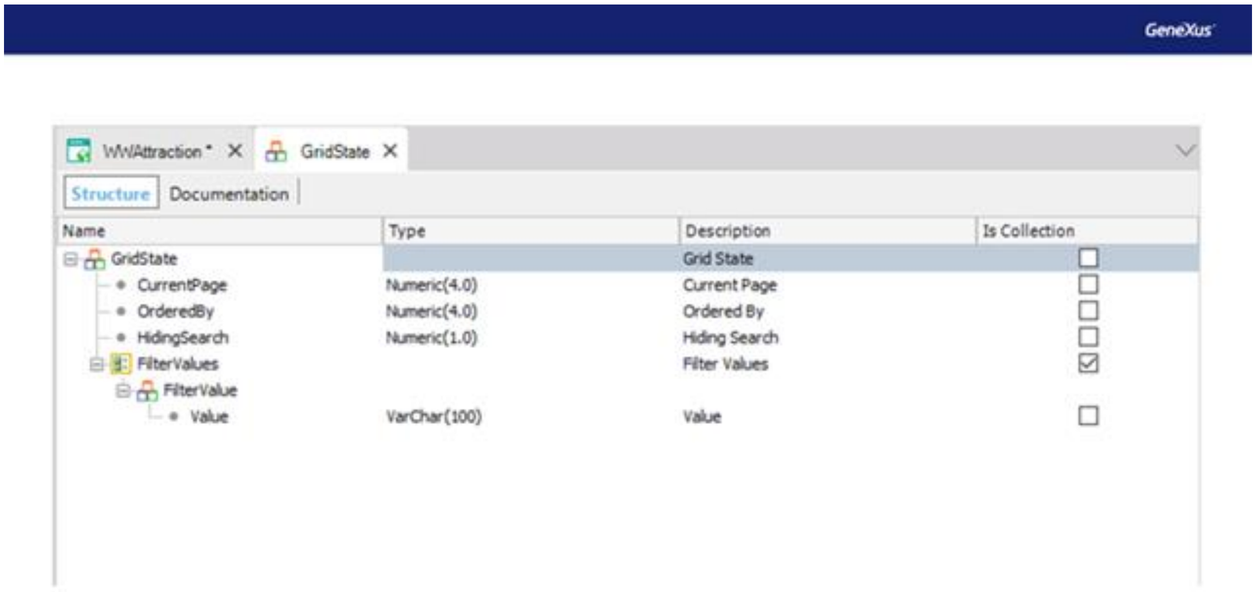
Name	Type	Is Collection	Description
Variables			
Standard Variables			
ADVANCED_LABEL_TEMPLATE	Character(20)	☐	ADVANCED_LABEL_TEMPLATE
AttractionName	Attribute:AttractionName	☐	Attraction Name
CountryName	Attribute:CountryName	☐	Country Name
Delete	Character(20)	☐	Delete
GridPageCount	Numeric(8.0-)	☐	Grid Page Count
GridState	GridState	☐	Grid State
GridStateFilterValue	GridState.FilterValue	☐	Grid State Filter Value
HTTPRequest	HttpRequest	☐	HTTPRequest
IsAuthorized	Boolean	☐	Is Authorized
OrderedBy	Numeric(4.0)	☐	Ordered By
Session	WebSession	☐	Session
TrnContext	TransactionContext	☐	Trn Context
TrnContextAtt	TransactionContext.Attribute	☐	Trn Context Att
Update	Character(20)	☐	Update

Note that GridState is of GridState type, and GridStateFilterValue is of GridState.FilterValue type.

What do these types make reference to?

If you consider the objects in your KB, they make reference to an object of SDT type, which was automatically generated by GeneXus when the WorkWith was created.

Let’s see the structure of this SDT.



Name	Type	Description	Is Collection
GridState		Grid State	☐
CurrentPage	Numeric(4.0)	Current Page	☐
OrderedBy	Numeric(4.0)	Ordered By	☐
HidingSearch	Numeric(1.0)	Hiding Search	☐
FilterValues		Filter Values	☑
FilterValue			
Value	VarChar(100)	Value	☐

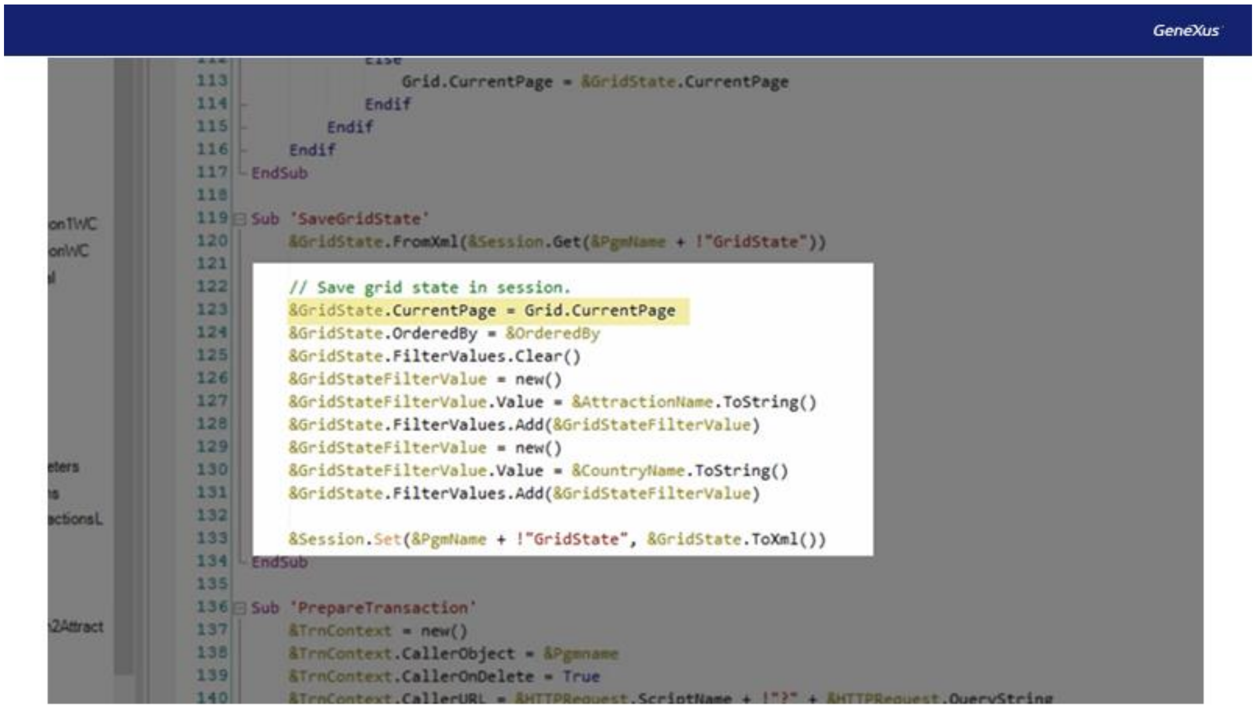
Its main structure is made up of three members: CurrentPage, OrderedBy, and HidingSearch. All three are of Numeric type.

Also, note that there is a substructure called FilterValue, which will be a collection.

Within this substructure a member called Value, of VarChar type, has been declared.

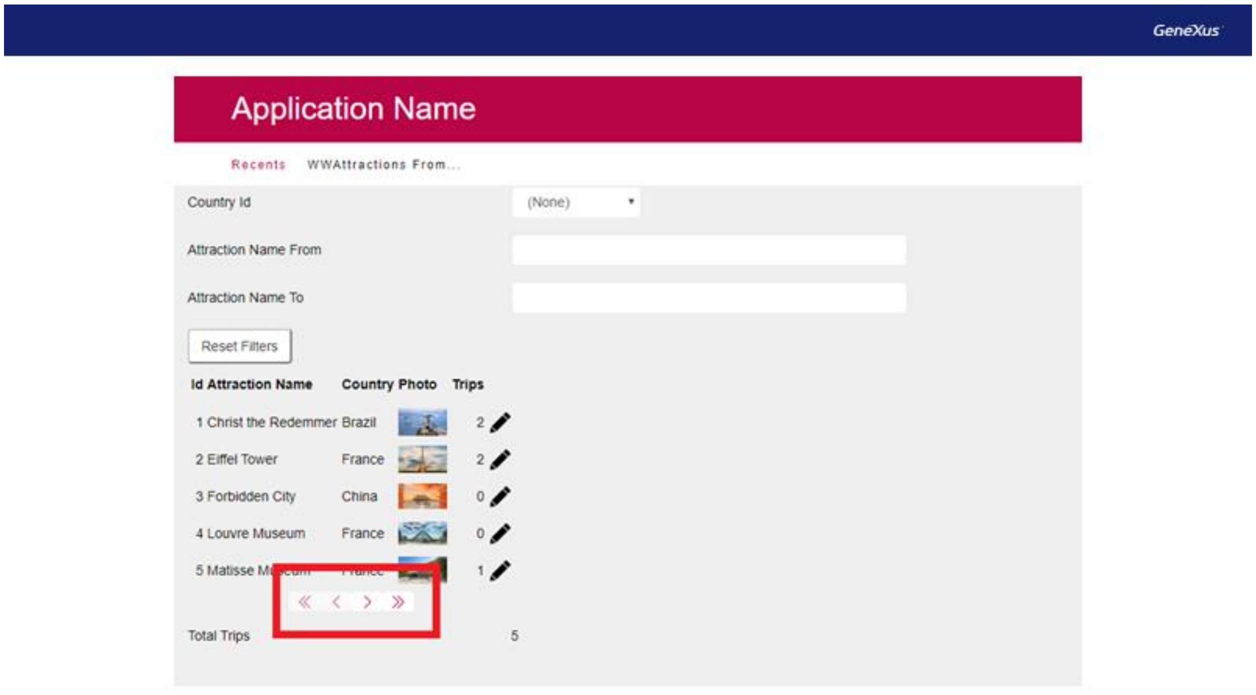
Let’s go back to the events section of the WWAttraction object.

Within the code we are interested in, note that the first line stores ‑in the CurrentPage member of the GridState variable‑ the number of the grid page we are in.



```
                Else
                    Grid.CurrentPage = &GridState.CurrentPage
                Endif
            Endif
        Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

    // 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())
EndSub

Sub 'PrepareTransaction'
    &TrnContext = new()
    &TrnContext.CallerObject = &Pgmname
    &TrnContext.CallerOnDelete = True
    &TrnContext.CallerURL = &HTTPRequest.ScriptName + !"?" + &HTTPRequest.QueryString
```

In this case, there was no paging in the grid. However, if in this WorkWith we configured the Row property of the grid, and set a value for it, such as 5, what we'll say here is that we want to show 5 rows per page. When updating, we see that only the first 5 elements will appear, and then it will give the option to advance the page to see the remaining data. In that case, we would also be interested in saving the page on which we were positioned. That's why the Pattern does it, to cover that scenario.

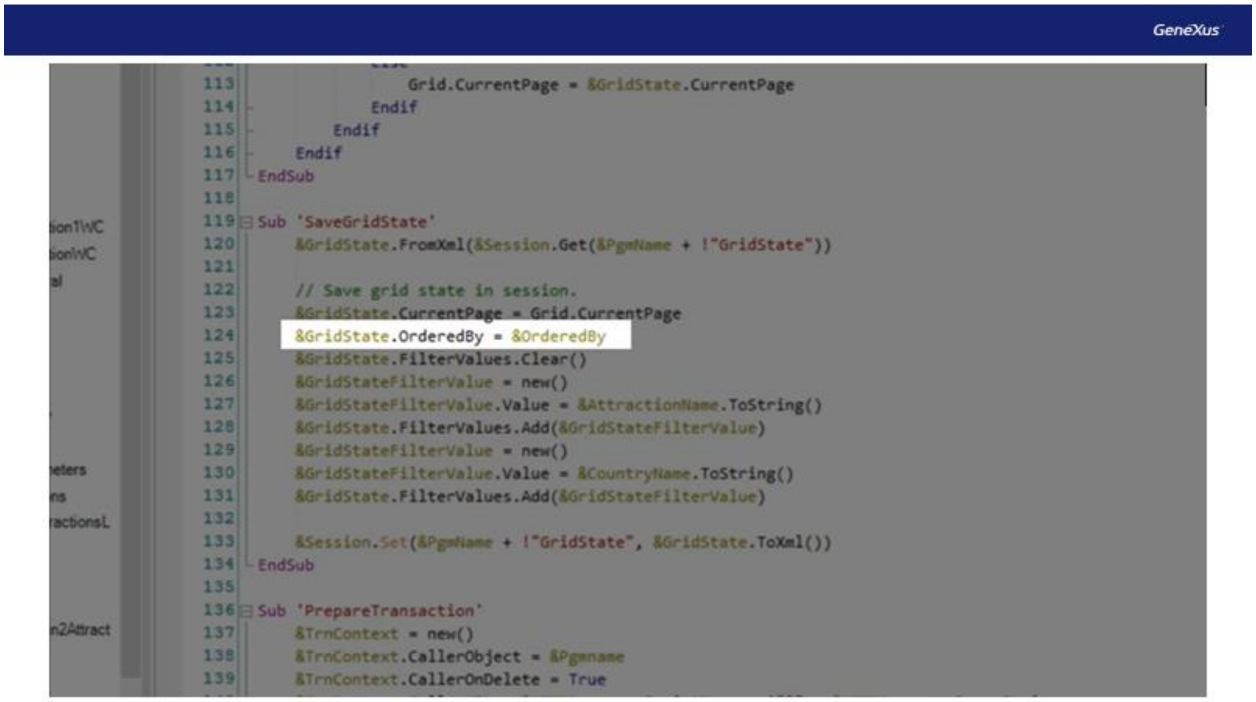


Country Id	(None)
Attraction Name From	
Attraction Name To	

Id	Attraction Name	Country	Photo	Trips
1	Christ the Redemmer	Brazil		2
2	Eiffel Tower	France		2
3	Forbidden City	China		0
4	Louvre Museum	France		0
5	Matisse Museum	France		1

Total Trips 5

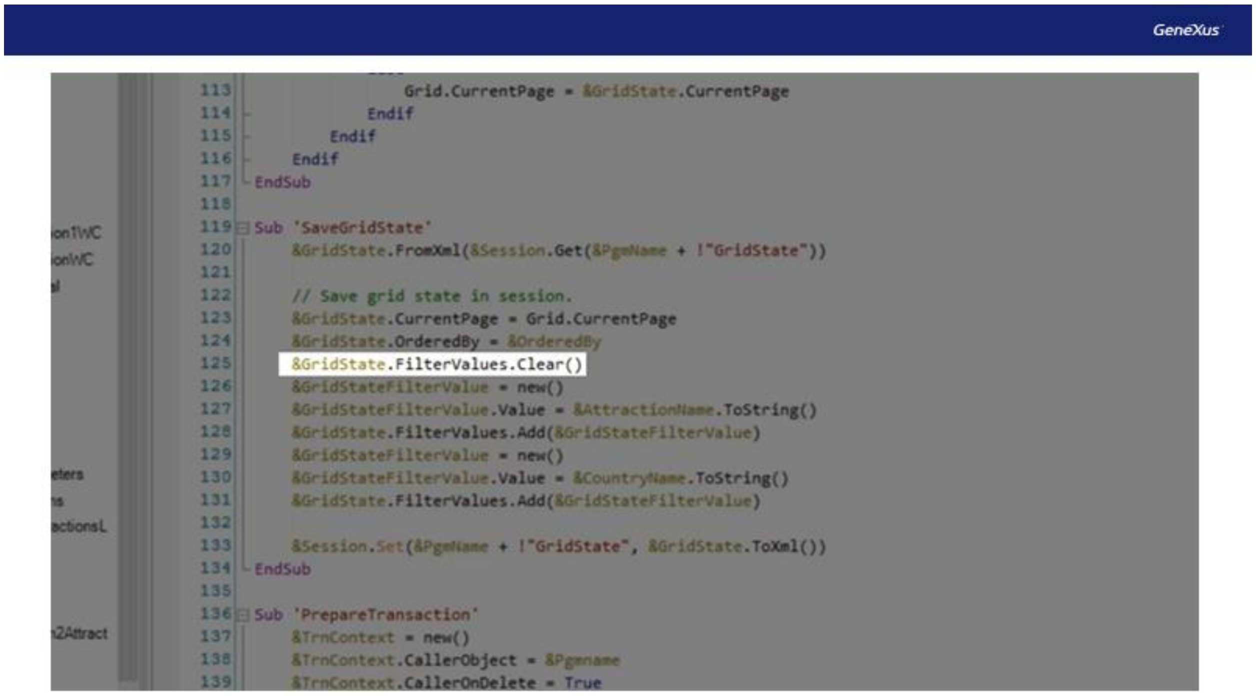
The order applied to the data will be saved in the next line, in the OrderedBy member of the GridState SDT.



```
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

    // 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())
EndSub

Sub 'PrepareTransaction'
    &TrnContext = new()
    &TrnContext.CallerObject = &Pgmname
    &TrnContext.CallerOnDelete = True
```

Then, with the Clear() method, any data loaded in the FilterValues collection will be deleted.

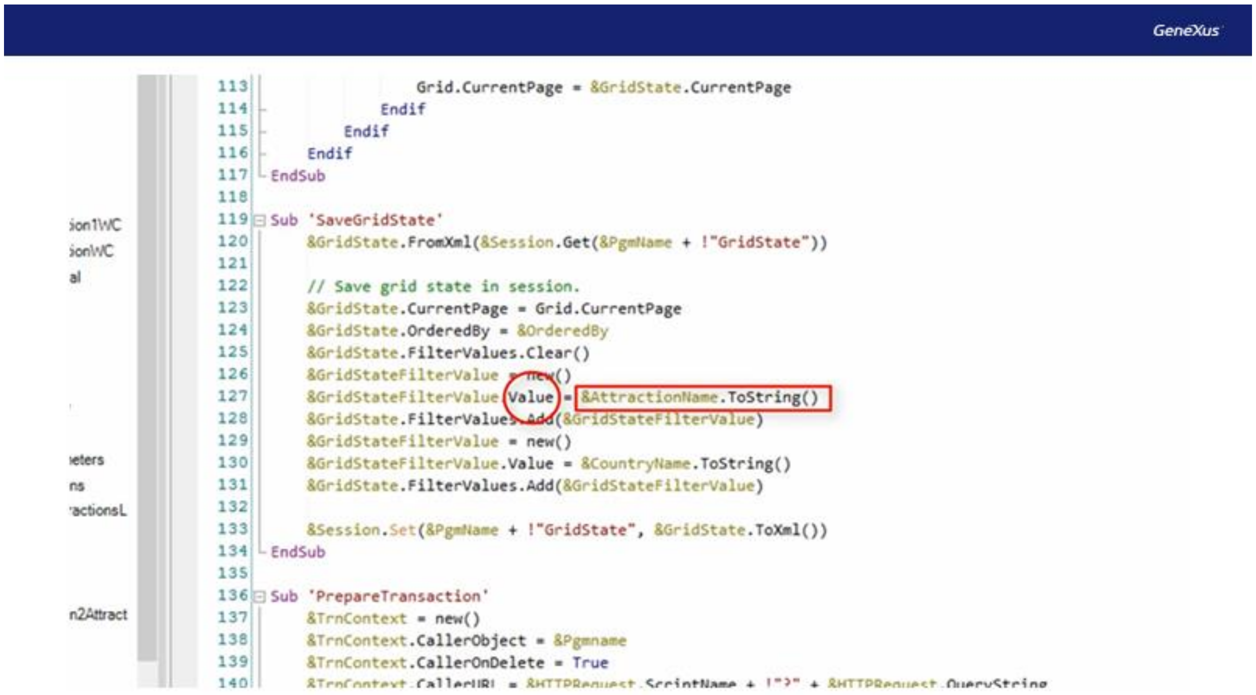
Now the GridStateFilterValue variable will be displayed in action; as we've seen it is of the GridState type pointing to the FilterValue substructure.



```
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

    // 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())
EndSub

Sub 'PrepareTransaction'
    &TrnContext = new()
    &TrnContext.CallerObject = &Pgmname
    &TrnContext.CallerOnDelete = True
```

First of all, it is assigned the new() operator, which returns a new initialized instance of the SDT variable, so that a value can be loaded in it later.

This is done in the Value member, assigning it the AttractionName variable, which is the one used by the WorkWith to enter the filter by name.



```
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

    // 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())
EndSub

Sub 'PrepareTransaction'
    &TrnContext = new()
    &TrnContext.CallerObject = &Pgmname
    &TrnContext.CallerOnDelete = True
    &TrnContext.CallerURL = &HTTPRequest.ScriptName + !"?" + &HTTPRequest.QueryString
```

This variable is then added to the FilterValues collection.

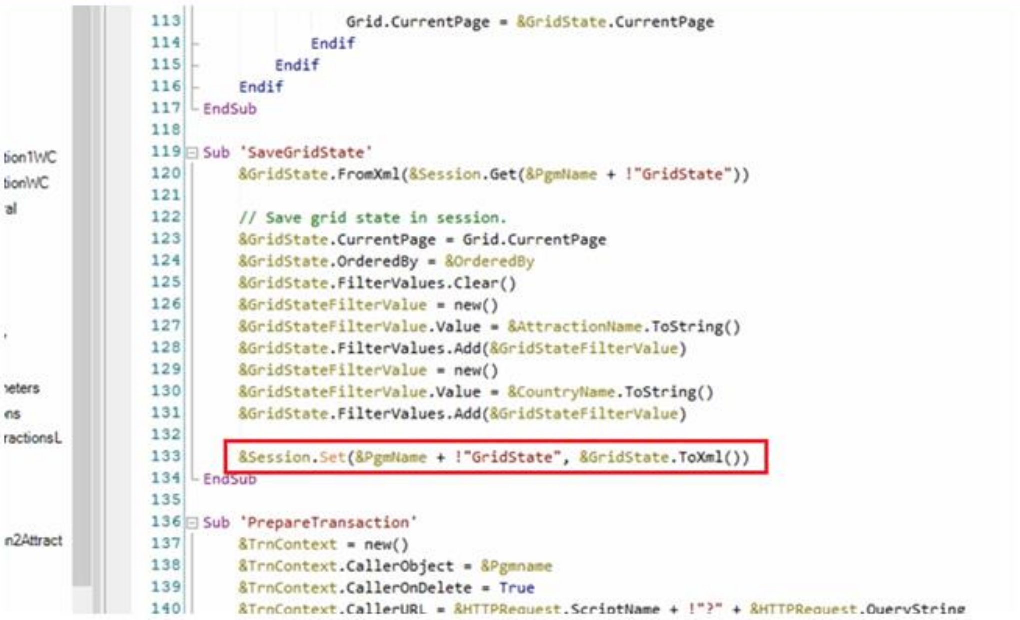
The same procedure is repeated again, but in this case with the CountryName variable, which will contain the value of the filter by country name.

Next, note there is a variable named Session, and if you go to the variables section you’ll see that it is of WebSession type.

This variable will store the data collected in the previous lines.

For this, the Set method will be used on the session variable, passing it a key and a value through parameters.

As a key it will use the value of the variable PgmName concatenated with the text GridState.



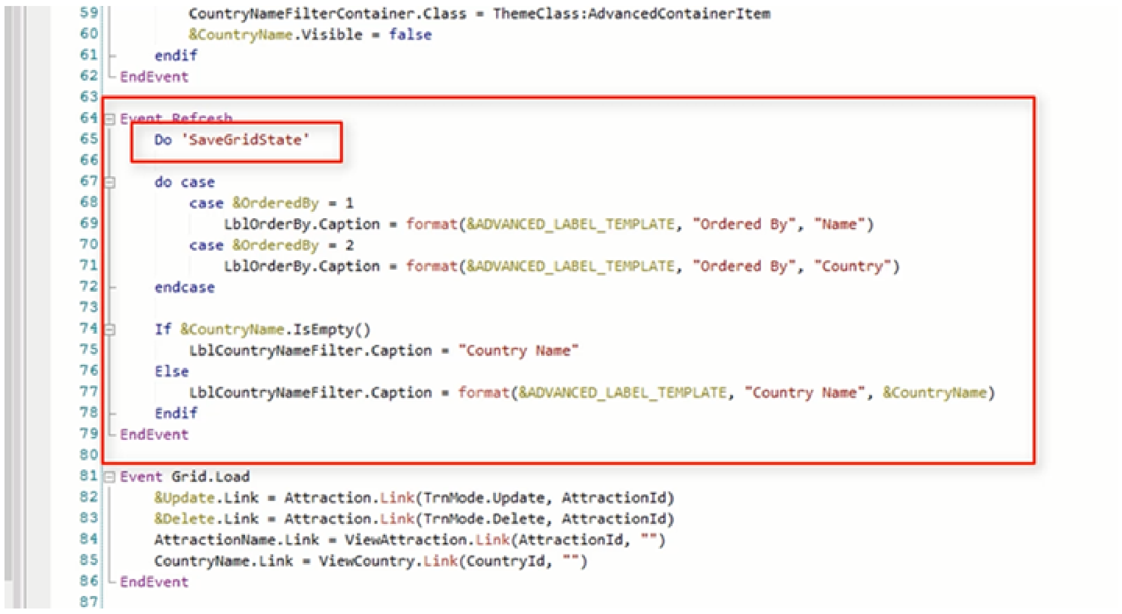
```
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

    // 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())
EndSub

Sub 'PrepareTransaction'
    &TrnContext = new()
    &TrnContext.CallerObject = &Pgmname
    &TrnContext.CallerOnDelete = True
    &TrnContext.CallerURL = &HTTPRequest.ScriptName + !"?" + &HTTPRequest.QueryString
```

PgmName is a variable that stores the name of the active program. That is to say, the name indicated in the object name property. In this case it is “WWAttraction.”

As a value it will save the GridState variable. As we’ve seen, it is of the SDT type and where the data you want to store was saved.

Now, note where the invocation to this subroutine is made; that is, the moment in which it will be used.

It is called using the “DO” command in the Refresh event. Every time this event is triggered, the values of the desired filters will be saved.



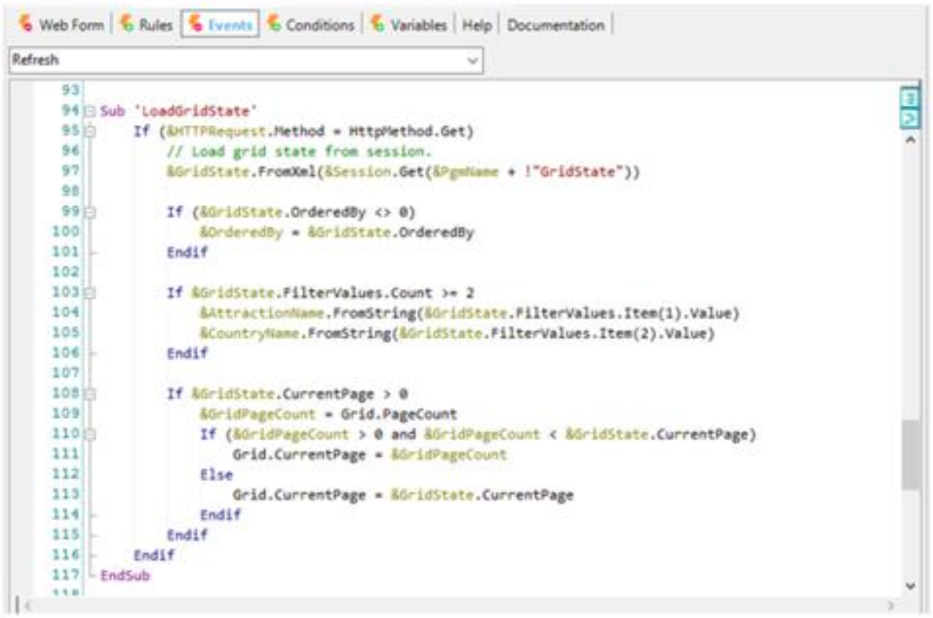
```
        CountryNameFilterContainer.Class = ThemeClass:AdvancedContainerItem
        &CountryName.Visible = false
    endif
EndEvent

Event Refresh
    Do 'SaveGridState'

    do case
        case &OrderedBy = 1
            LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Name")
        case &OrderedBy = 2
            LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Country")
    endcase

    If &CountryName.IsEmpty()
        LblCountryNameFilter.Caption = "Country Name"
    Else
        LblCountryNameFilter.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Country Name", &CountryName)
    Endif
EndEvent

Event Grid.Load
    &Update.Link = Attraction.Link(TrnMode.Update, AttractionId)
    &Delete.Link = Attraction.Link(TrnMode.Delete, AttractionId)
    AttractionName.Link = ViewAttraction.Link(AttractionId, "")
    CountryName.Link = ViewCountry.Link(CountryId, "")
EndEvent
```

In the previous video, for the solution in the WebPanel, we thought about saving the filter values in this event. And we decided it wasn't the best moment because every time the Refresh event is triggered these values will be saved. Maybe we're not interested in saving them because it won't be necessary to retrieve them later, and yet we are saving them again and again. For example, every time the value of one filter is changed, this event is triggered and these values will be saved.

The Pattern does it this way precisely because it is a pattern, and as a result it addresses all possible situations. Since the pattern doesn't know exactly what we are going to use the application for, it covers all possible scenarios.

Now let's see how and where this information we saved will be retrieved.

Note that another subroutine called LoadGridState has been generated. Here is where the information saved in the SaveGridState subroutine will be retrieved.



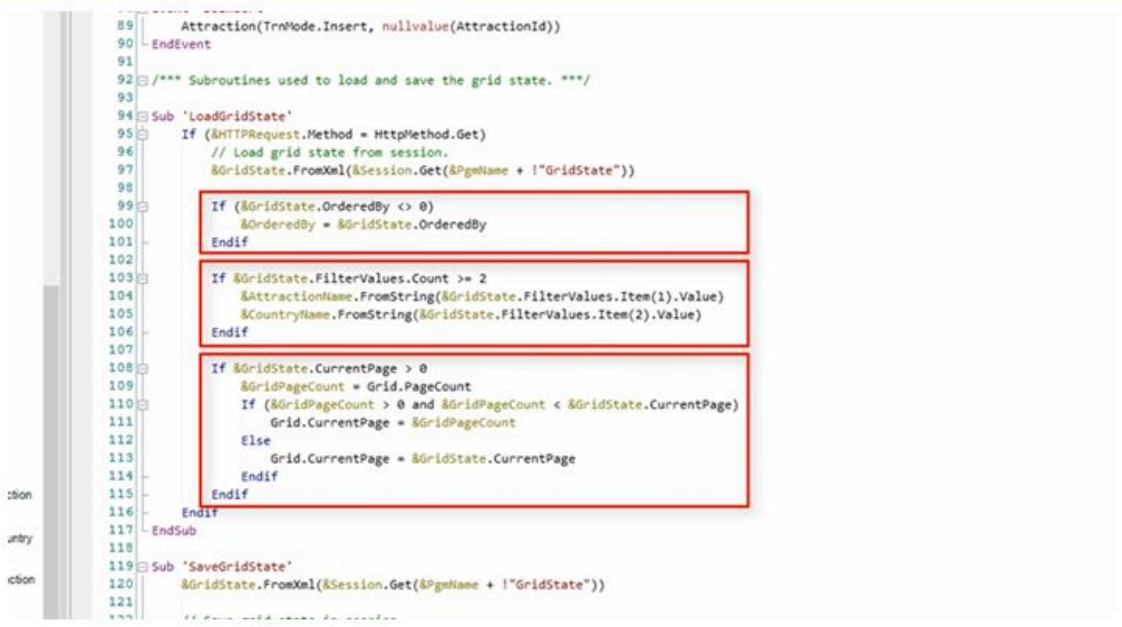
```
Sub 'LoadGridState'
    If (&HTTPRequest.Method = HttpMethod.Get)
        // Load grid state from session.
        &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

        If (&GridState.OrderedBy <> 0)
            &OrderedBy = &GridState.OrderedBy
        Endif

        If &GridState.FilterValues.Count >= 2
            &AttractionName.FromString(&GridState.FilterValues.Item(1).Value)
            &CountryName.FromString(&GridState.FilterValues.Item(2).Value)
        Endif

        If &GridState.CurrentPage > 0
            &GridPageCount = Grid.PageCount
            If (&GridPageCount > 0 and &GridPageCount < &GridState.CurrentPage)
                Grid.CurrentPage = &GridPageCount
            Else
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub
```

In this line; the information previously saved in the Session session variable will be loaded in the GridState variable, which is of SDT type.

This is done through the FromXml method, which receives an XML string from where the information is loaded in the SDT object.

This method will receive in a parameter the location from where the data is to be retrieved, which in this case will be the Session variable. Note that it uses the Get method to indicate the value to retrieve using the key that was saved for it. Remember that the key assigned to save the data was the concatenation of the PgmName variable; that is, the name of the object plus the GridState text string.

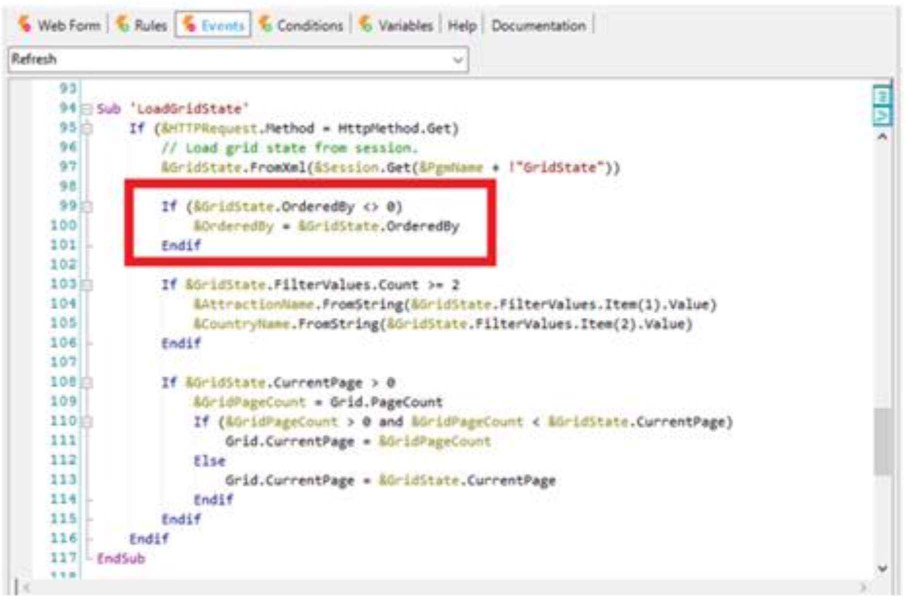
Then we see that several checks are made, using IF commands.



```
    Attraction(TrnMode.Insert, nullvalue(AttractionId))
EndEvent

/*** Subroutines used to load and save the grid state. ***/

Sub 'LoadGridState'
    If (&HTTPRequest.Method = HttpMethod.Get)
        // Load grid state from session.
        &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

        If (&GridState.OrderedBy <> 0)
            &OrderedBy = &GridState.OrderedBy
        Endif

        If &GridState.FilterValues.Count >= 2
            &AttractionName.FromString(&GridState.FilterValues.Item(1).Value)
            &CountryName.FromString(&GridState.FilterValues.Item(2).Value)
        Endif

        If &GridState.CurrentPage > 0
            &GridPageCount = Grid.PageCount
            If (&GridPageCount > 0 and &GridPageCount < &GridState.CurrentPage)
                Grid.CurrentPage = &GridPageCount
            Else
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub

Sub 'SaveGridState'
    &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

```

This section verifies that the OrderedBy member of this SDT variable is different from 0; in other words, it checks if there is any saved order, to find out if it is necessary to retrieve it or not.

If it is different from 0 it retrieves this information saved in a variable called OrderedBy.

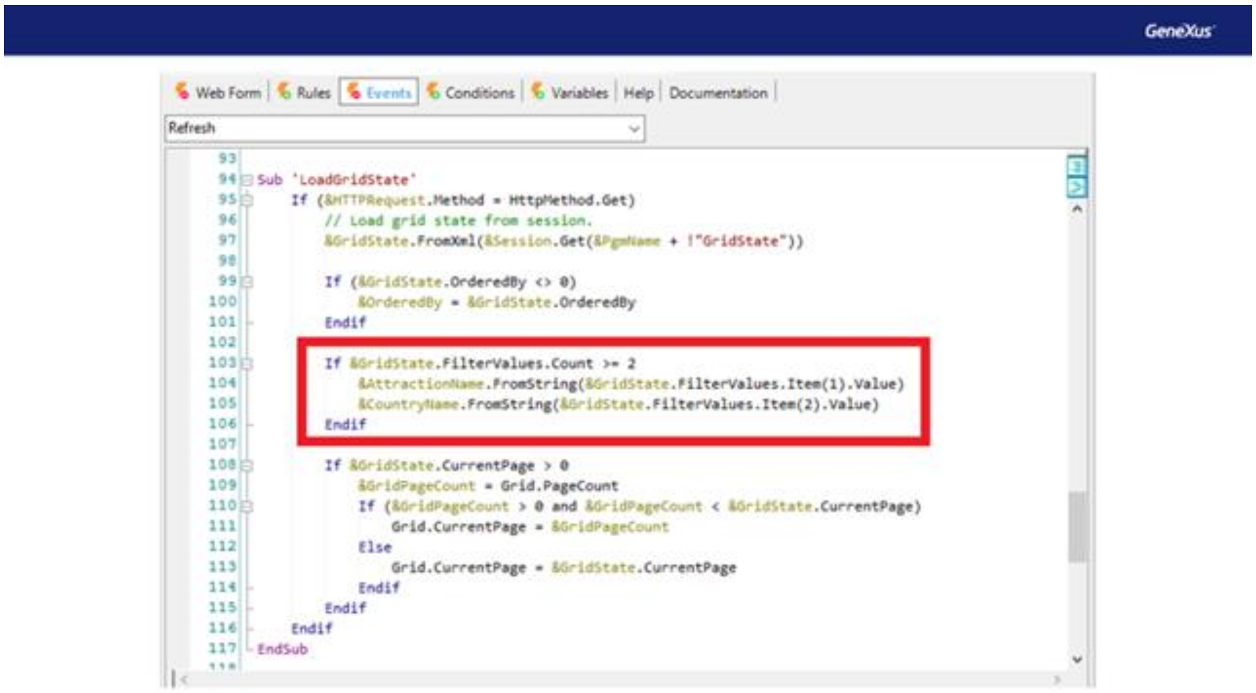


```

Sub 'LoadGridState'
    If (&HTTPRequest.Method = HttpMethod.Get)
        // Load grid state from session.
        &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

        If (&GridState.OrderedBy <> 0)
            &OrderedBy = &GridState.OrderedBy
        Endif

        If &GridState.FilterValues.Count >= 2
            &AttractionName.FromString(&GridState.FilterValues.Item(1).Value)
            &CountryName.FromString(&GridState.FilterValues.Item(2).Value)
        Endif

        If &GridState.CurrentPage > 0
            &GridPageCount = Grid.PageCount
            If (&GridPageCount > 0 and &GridPageCount < &GridState.CurrentPage)
                Grid.CurrentPage = &GridPageCount
            Else
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub
```

Then, it checks if the number of data items stored in the filterValues collection is greater than or equal to two. This value is assigned because there are two filters in the WorkWith. It checks that this data is saved to be retrieved later.

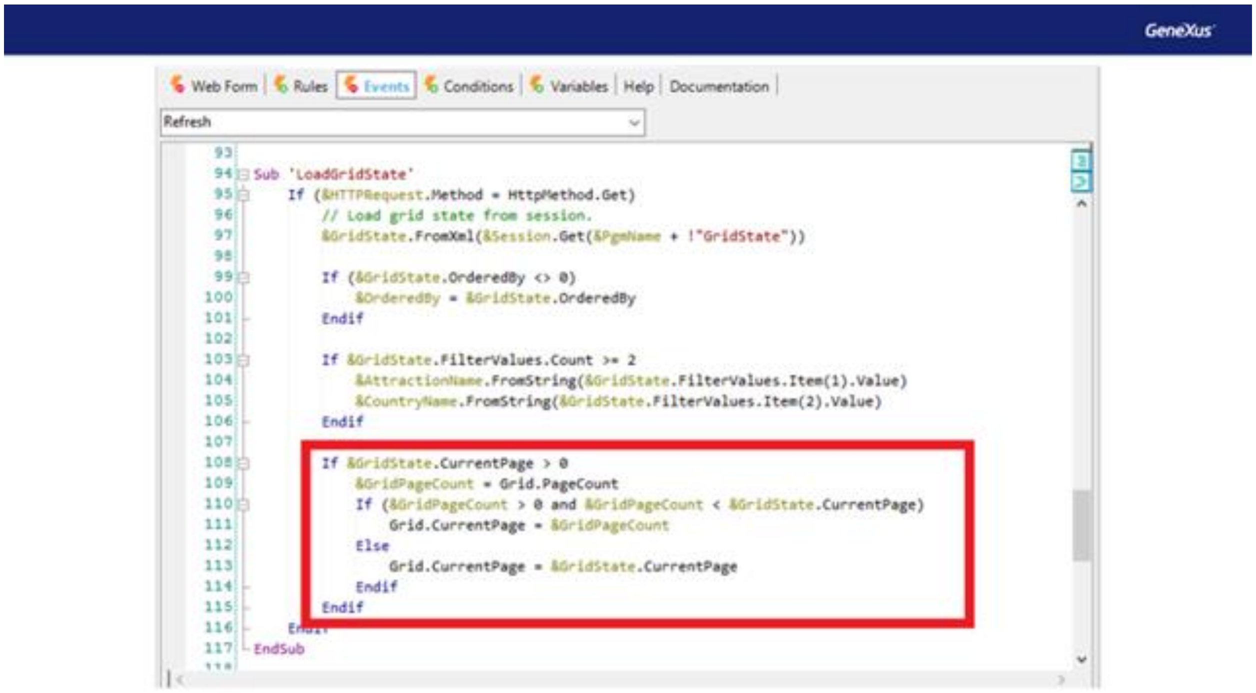
This block of code is where the filter values will be retrieved.

The filter of the attraction name will be saved in the AttractionName variable through the FromString method. The same happens with the country name filter, saving the value in the CountryName variable.



```
Sub 'LoadGridState'
    If (&HTTPRequest.Method = HttpMethod.Get)
        // Load grid state from session.
        &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

        If (&GridState.OrderedBy <> 0)
            &OrderedBy = &GridState.OrderedBy
        Endif

        If &GridState.FilterValues.Count >= 2
            &AttractionName.FromString(&GridState.FilterValues.Item(1).Value)
            &CountryName.FromString(&GridState.FilterValues.Item(2).Value)
        Endif

        If &GridState.CurrentPage > 0
            &GridPageCount = Grid.PageCount
            If (&GridPageCount > 0 and &GridPageCount < &GridState.CurrentPage)
                Grid.CurrentPage = &GridPageCount
            Else
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    Endif
EndSub
```

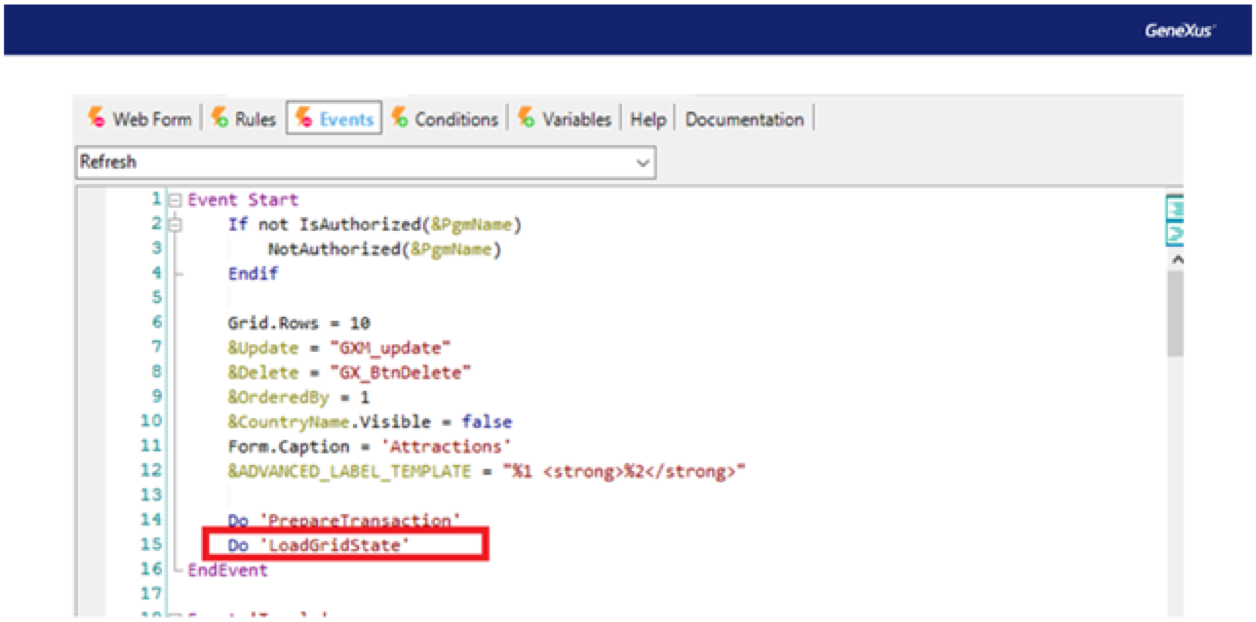
The last thing to check is whether the CurrentPage member, used to save the grid page, is greater than 0. If so, it retrieves that value and assigns it directly to the CurrentPage property of the grid.



```
Sub 'LoadGridState'
    If (&HTTPRequest.Method = HttpMethod.Get)
        // Load grid state from session.
        &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))

        If (&GridState.OrderedBy <> 0)
            &OrderedBy = &GridState.OrderedBy
        Endif

        If &GridState.FilterValues.Count >= 2
            &AttractionName.FromString(&GridState.FilterValues.Item(1).Value)
            &CountryName.FromString(&GridState.FilterValues.Item(2).Value)
        Endif

        If &GridState.CurrentPage > 0
            &GridPageCount = Grid.PageCount
            If (&GridPageCount > 0 and &GridPageCount < &GridState.CurrentPage)
                Grid.CurrentPage = &GridPageCount
            Else
                Grid.CurrentPage = &GridState.CurrentPage
            Endif
        Endif
    En
EndSub
```

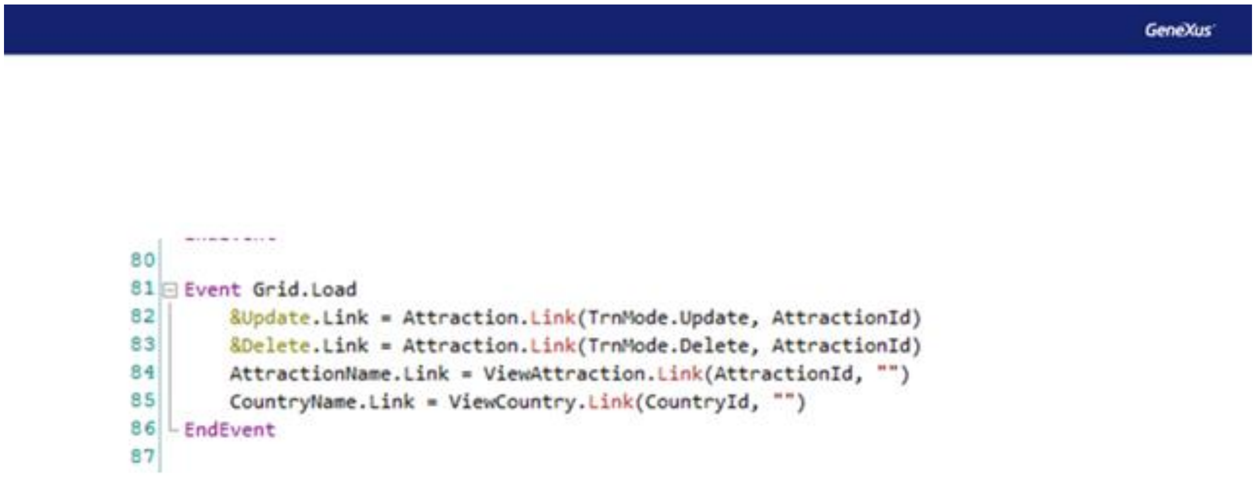
When will this subroutine be called?

As discussed when creating our own Web Panel, the only time possible will be at the Start event. This event is run only once when the website is first loaded.



```
Event Start
    If not IsAuthorized(&PgmName)
        NotAuthorized(&PgmName)
    Endif

    Grid.Rows = 10
    &Update = "GXM_update"
    &Delete = "GX_BtnDelete"
    &OrderedBy = 1
    &CountryName.Visible = false
    Form.Caption = 'Attractions'
    &ADVANCED_LABEL_TEMPLATE = "%1 <strong>%2</strong>"

    Do 'PrepareTransaction'
    Do 'LoadGridState'
EndEvent
```

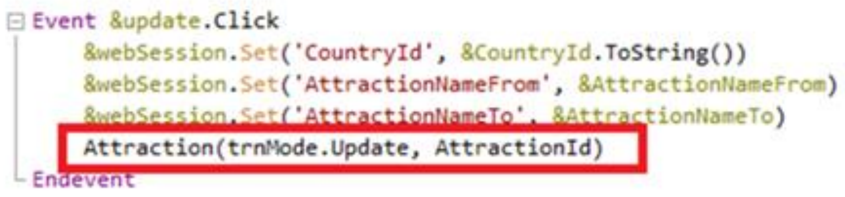
Another difference between this object and the one we created can be seen in the Load event of the grid.



```
Event Grid.Load
    &Update.Link = Attraction.Link(TrnMode.Update, AttractionId)
    &Delete.Link = Attraction.Link(TrnMode.Delete, AttractionId)
    AttractionName.Link = ViewAttraction.Link(AttractionId, "")
    CountryName.Link = ViewCountry.Link(CountryId, "")
EndEvent
```

To update the information of a grid record, both our web Panel and the one generated by the pattern, the &update variable is used. Clicking on it takes us to the corresponding screen, which in this case is the Attraction transaction in Update mode.

Let's remember how we implemented it in our Web Panel.

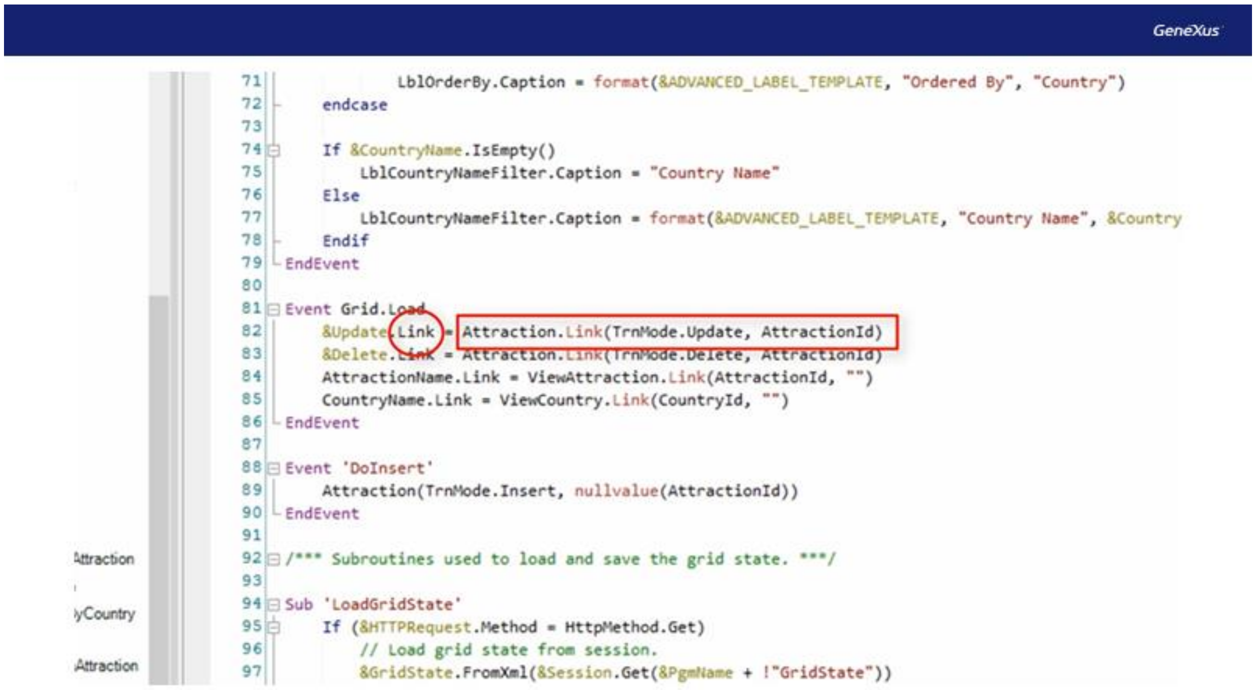


```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
    Attraction(trnMode.Update, AttractionId)
Endevent
```

We did it inside the &update.click event, invoking the Attraction transaction directly, passing it by parameter the mode to execute, and the attraction ID.

The pattern does it in the Grid.Load event, which will be executed as many times as records are to be loaded in the grid.

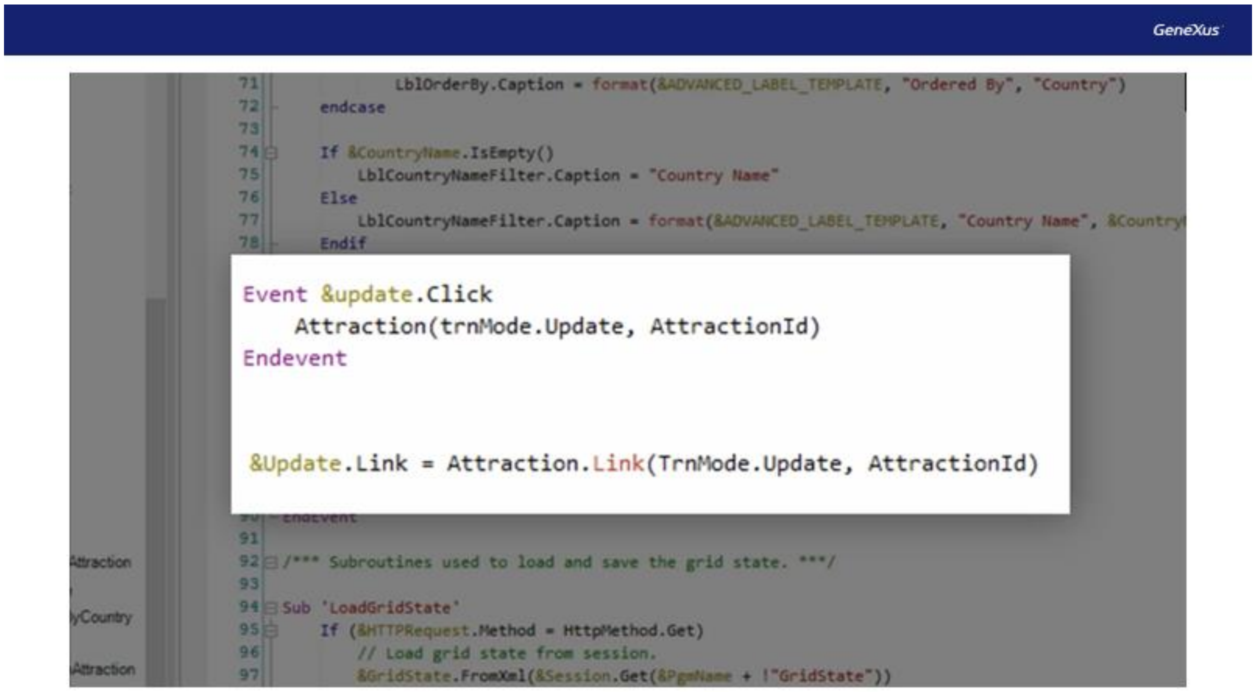
Note that the Pattern applied the Link property to the Update variable. And then it indicates that this value will be equal to the Link() function of the Attraction transaction, passing by parameter the mode in which you want to execute this transaction and the ID that identifies it, in this case AttractionId. That is, it associates the Link function with the link property of the variable. As a result, when you click on this variable, the call to the Attraction web object is made.



```
            LblOrderBy.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Ordered By", "Country")
    endcase

    If &CountryName.IsEmpty()
        LblCountryNameFilter.Caption = "Country Name"
    Else
        LblCountryNameFilter.Caption = format(&ADVANCED_LABEL_TEMPLATE, "Country Name", &Country
    Endif
EndEvent

Event Grid.Load
    &Update.Link = Attraction.Link(TrnMode.Update, AttractionId)
    &Delete.Link = Attraction.Link(TrnMode.Delete, AttractionId)
    AttractionName.Link = ViewAttraction.Link(AttractionId, "")
    CountryName.Link = ViewCountry.Link(CountryId, "")
EndEvent

Event 'DoInsert'
    Attraction(TrnMode.Insert, nullvalue(AttractionId))
EndEvent

/*** Subroutines used to load and save the grid state. ***/

Sub 'LoadGridState'
    If (&HTTPRequest.Method = HttpMethod.Get)
        // Load grid state from session.
        &GridState.FromXml(&Session.Get(&PgmName + !"GridState"))
```

Although we are not providing an example, the same thing is done with the &delete variable, passing the corresponding mode as a parameter.

In this case, both ways of implementing it will have the same functionality.
When you click on the update variable, in both options, the Attraction object will be called. This indicates that you want to make an update, passing it the ID of the selected element, so that all the data can be loaded on the screen from this key.



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


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

As you can see, for the invocation of other web objects, it will be possible to use both the parameter and Link function as the direct invocation of the object.

While you can use either option in this example, there are differences between them.
For example, if the object you want to call is a procedure, you can only do it by directly invoking the object's name.

Example:

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

On the other hand, to make reference to a static HTML page, for example, the Link function must be used.

Example:

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

Now let's go back to the events section of the Web Panel.

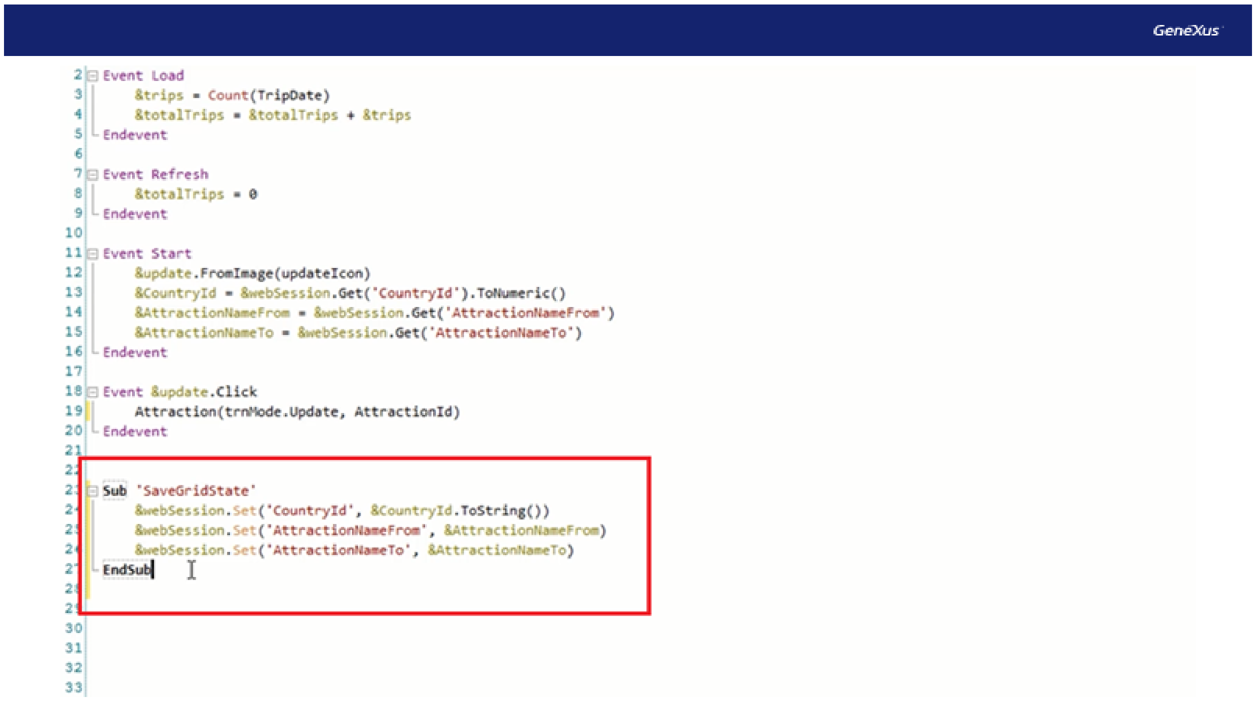
Let's try and practice using the subroutines we've just seen, in the same way you use them in the Work With.

Thus, you will have a cleaner code, and it will be possible to reuse those subroutines in some other event within the same object as needed.

First, create the subroutine to save the filter data.

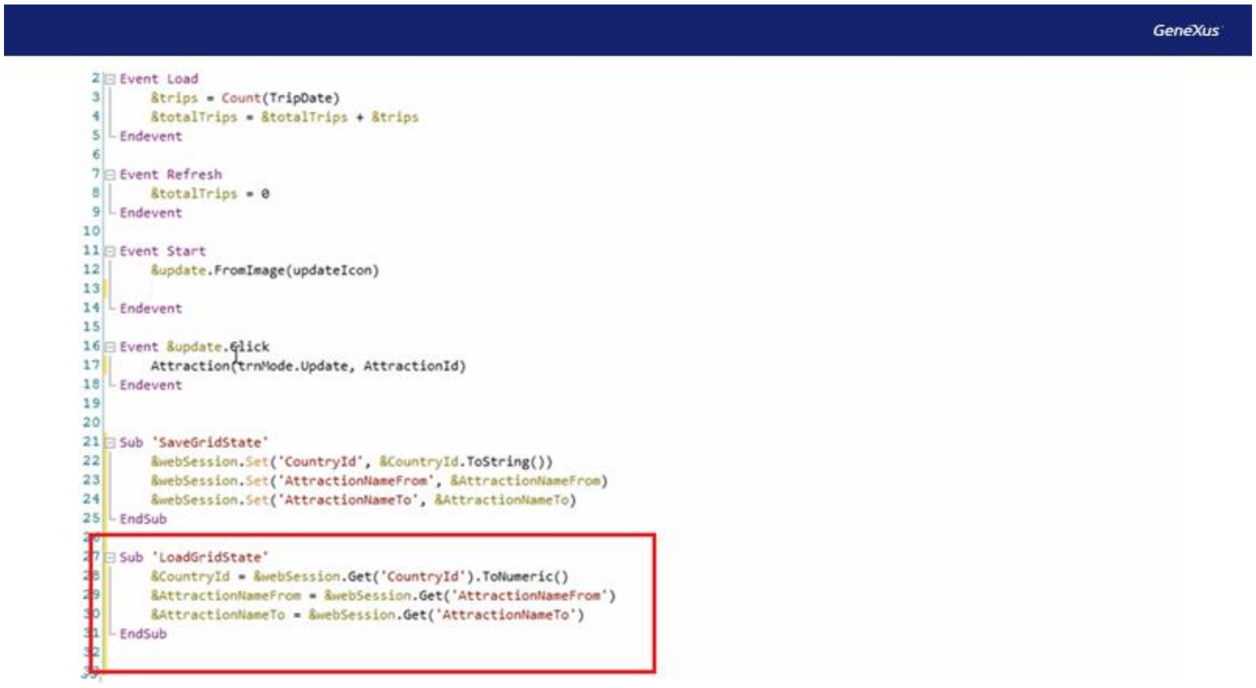
To do this, use the 'Sub' command and give it the same name that was assigned to it by the Work With, 'SaveGridState.'

Now, place in it the previously created code that you programmed in the &update.click event, where you assigned the set methods to the webSession variable, passing each variable as a key and value.



```
Event Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
    &totalTrips = 0
Endevent

Event Start
    &update.FromImage(updateIcon)
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
Endevent

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


Sub 'SaveGridState'
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
EndSub
```

Then do the same, but with the 'LoadGridState' subroutine. Here you will retrieve the values by passing a key to it, and that value is assigned to each filter variable. So far, this was located in the Start event.



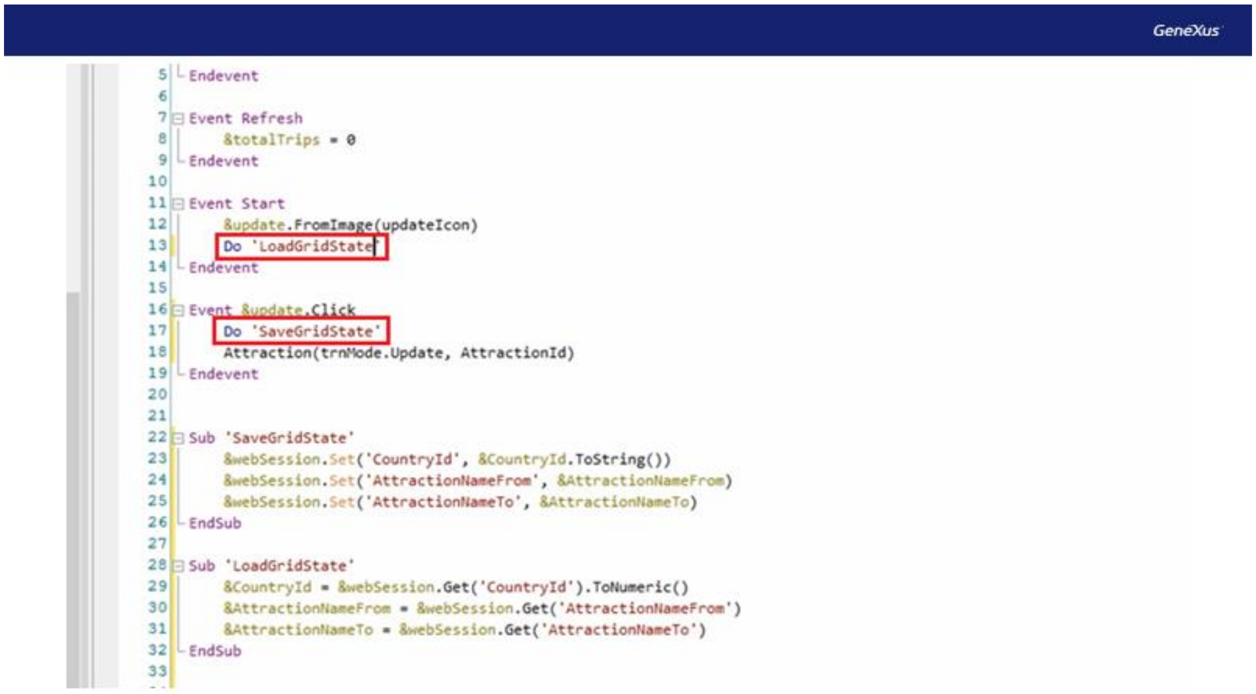
```
Event Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
    &totalTrips = 0
Endevent

Event Start
    &update.FromImage(updateIcon)

Endevent

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


Sub 'SaveGridState'
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
EndSub

Sub 'LoadGridState'
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
EndSub
```

Now you would only have to call those subroutines from the corresponding events.

In the case of the 'SaveGridState' subroutine, we will call it in the event &update.Click, for the reasons explained in the previous video. However, we saw that if we do it from the Refresh event, since it is generated by the Work With automatically, it will also meet our requirements.

Now, where should we place the call to the 'LoadGridState' subroutine?

As discussed, it will be in the Start event.



```
Endevent

Event Refresh
    &totalTrips = 0
Endevent

Event Start
    &update.FromImage(updateIcon)
    Do 'LoadGridState'
Endevent

Event &update.Click
    Do 'SaveGridState'
    Attraction(trnMode.Update, AttractionId)
Endevent


Sub 'SaveGridState'
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
EndSub

Sub 'LoadGridState'
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
EndSub
```

In this way, we have made a mix between how we had implemented it in our way at the beginning, and how the Work With implements it automatically.

In addition, we could also implement it with an SDT object and a variable of this data type, in the same way the Pattern does.
We are not going to do it in this example, but it would be a good practice that after this video you try to do it yourself.

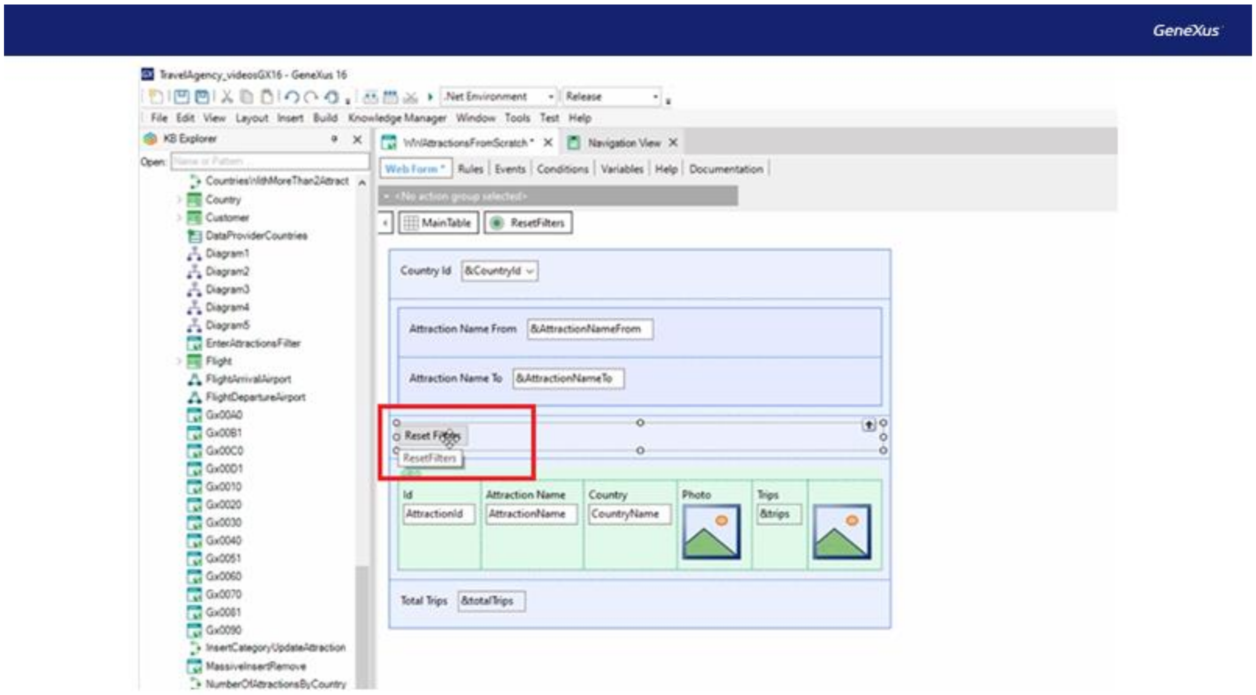
Keeping the filters between screen runs can be beneficial when trying to implement this functionality. However, it may be cumbersome when we don't want them to be kept. The filters we have entered must be removed one by one.

How can we make it easier to clean these filters?

One option would be to add to your Web Panel a button that implements this functionality.
Basically, you'll need this button to perform three functions:

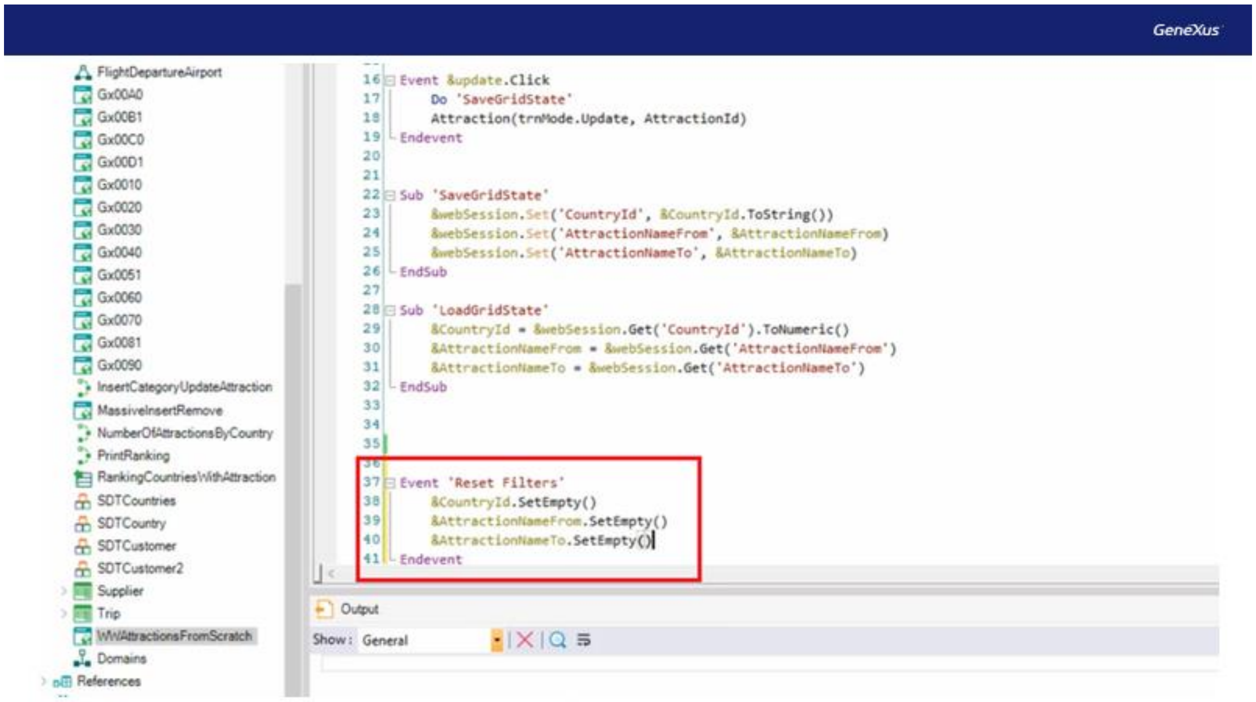
- Empty the variables used for the filters.
- Empty the Web Session type variable.
- Refresh the grid.

To do this, drag a control of the Button type to the Web Panel, and name it “Reset Filters.”



Next, double-click on the button, and it will take you to program its event.

As we’ve said, the first feature you need is to empty the variables used for the filters.
To do this, apply the SetEmpty() method to each variable. This way you “empty” each one of them.



```
Event &update.Click
    Do 'SaveGridState'
    Attraction(trnMode.Update, AttractionId)
Endevent


Sub 'SaveGridState'
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
EndSub

Sub 'LoadGridState'
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
EndSub




Event 'Reset Filters'
    &CountryId.SetEmpty()
    &AttractionNameFrom.SetEmpty()
    &AttractionNameTo.SetEmpty()
Endevent
```

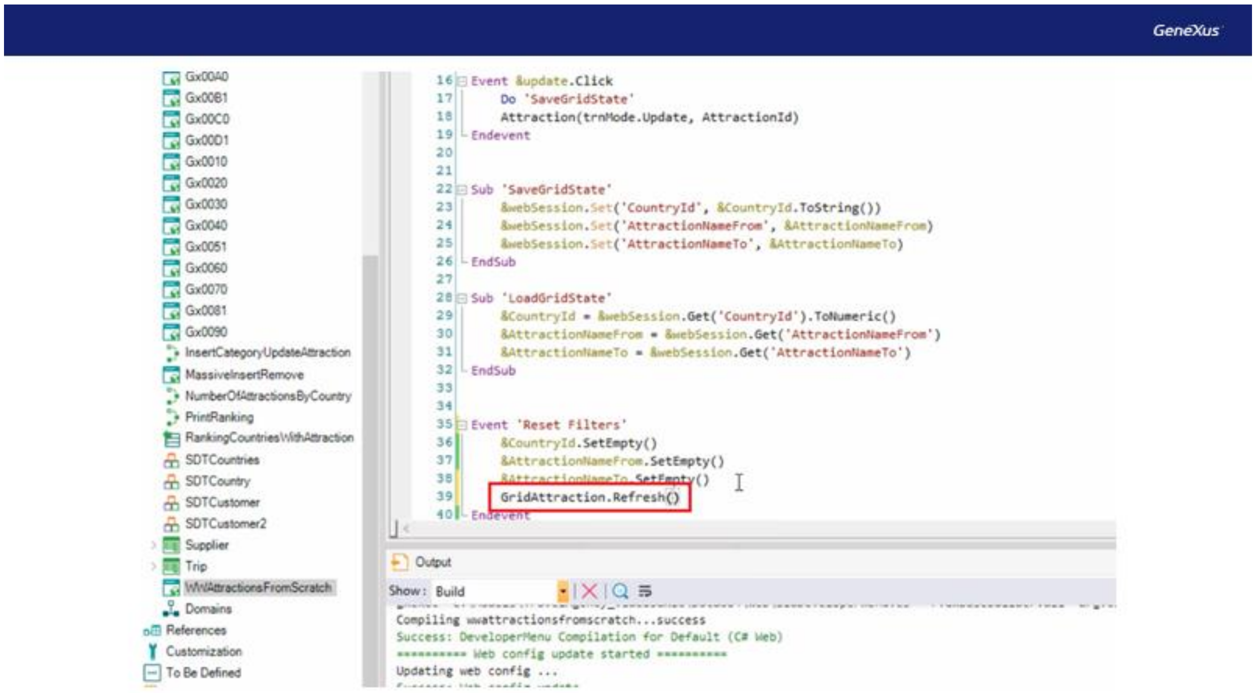
Let’s try to run it now.

Enter, for example, the country China and filter the attractions from A to J.

Now click on the “Reset Filters” button.

Note that the filters are empty, but the grid is not updated.
This is because the grid has not been ordered to be updated. How do you do this?

Add, within the button event, the Refresh() method to the grid you want to update, in this case GridAttraction.



```
Event &update.Click
    Do 'SaveGridState'
    Attraction(trnMode.Update, AttractionId)
Endevent


Sub 'SaveGridState'
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
EndSub

Sub 'LoadGridState'
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
EndSub


Event 'Reset Filters'
    &CountryId.SetEmpty()
    &AttractionNameFrom.SetEmpty()
    &AttractionNameTo.SetEmpty()
    GridAttraction.Refresh()
Endevent
```

```
Compiling wwattractionsfromscratch...success
Success: DeveloperMenu Compilation for Default (C# Web)
========== Web config update started ==========
Updating web config ...
```

Run it again and try to enter values in the filters, and then click on the Reset Filters button. Note that it now works properly and updates the grid as desired.

We’ve seen how the Pattern Work With implements the functionality to save the values of the variables that impact the grid. They are the order applied to the grid, the values of the editable filter fields, and the page number corresponding to the grid.
Also, we compared this solution with the one we made in our own way, which was proposed in the first video, explaining the differences between them.
Finally, we made changes in our WebPanel using subroutines, to explain its operation and understand its use, as well as the WorkWith.

Visit our Wiki to learn more about the topics discussed: https://wiki.genexus.com/