Offline Introduction

1.02.07 Table Controls

Table management has changed in the current release of Offline. Users are now able to define multiple table controls in an Offline screen. Users can also change the way in which tables are displayed on screens. Columns can be sorted, resized or even moved and with the sole exception of sorting, these changes can be saved in the database so that the next time the same user logs in, the table layout will be precisely as it was on the previous login. If the user does not wish to save the table settings, they can be reset to return to the application defaults.

Table controls are notated using the 'type' marker in a script. The syntax used is shown as follows:

```
{type:SCR_TABLE,caption
:"table_caption",position:{
top:T,left:L,bottom:B,right:R},
linkedtableinfo:"table_info
",linkedshowinfo:[
"plant","work_center","description"],
fieldname:"orderTableType",allowmultiplerowselection:true},
```

The parameters and options used in defining table controls are as follows:

- **Caption:** The caption or label used for the table on the screen.
- **Position:** Defines the location of the table using four parameters (top, bottom, left and right), which are defined by numerical digits. The numbers refer to the on-screen position, using row and column numbers. The top and bottom numbers are the row numbers while the left and right numbers are column numbers.
- **Linkedtableinfo:** The table in the SAP database from which the actual data is being pulled.
- **LinkedShowInfo:** The list of columns to display, contained in an array.

Managing Table Control Data

When users display data in a table control, Offline can be configured to display the data based on a given condition without selecting all the records from that table in question. This is contrary to the default operations - normally it is necessary to select all records from a particular table before the user can set conditions for displaying that data from that table.

This feature is very useful for users who want to define the conditions for displaying data in process before output (PBO) functions. These functions run before the screen is displayed in large data tables.

In the following example, we are displaying the table control generation code.

```
vQTableDefinition = this.findTableDefinition();
```

```
var objQTableControl = this.m_RealTimeScreen.FindTableListControl( vQT
ableDefinition.position.top,
vQTableDefinition.position.left); //Find the table control
if (objQTableControl !=null) { //if the table control is not empty
objQTableControl.RemoveControl();
//remove the control
objQTableControl = null; //null the object
vQTableDefinition.loadrecords = 1;
var chkUser = SR3_arConnections[this.m_CurrentTransaction.m_strCon
nectionName].m_strLogonUserName.toUpperCase();
vQTableDefinition.linkedtableinfo = striw2x_wqTCode + "^" + striw2x_wq
TCode + ".g_iw2x_wq_planner = '" + chkUser + "'";
}
if
(this.createTableListControl(vQTableDefinition) == false)
{
return;
}
```

The code that instructs Offline not to load all records in the table is the following line:

```
vQTableDefinition.loadrecords = 1;
```

This parameter instructs Offline how many records to load from the table. If the parameter is not used, then Offline will load all records, regardless of the retrieval condition specified in the PBO function.

Note: This parameter applies to and is true only for PBO functions.

The code that refers to the table refresh is highlighted in blue in the preceding example. This code is as follows:

```
if (this.createTableListControl(vQTableDefinition) == false)
{
return;
}
```

This code uses an If statement to return if the given table control definition is not accurate. In other words, the screen will be refreshed.

The section of the code in the example that actually contains the description of the

condition to display data is shown below:

```
vQTableDefinition.linkedtableinfo = striw2x_wqTCode + "^" + striw2x_wq
TCode + ".g_iw2x_wq_planner = '" + chkUser + "'";
}
```

In this code, the definition of precisely what data is to be displayed is specified. In the example, we are sending an SQL query which is defined as follows:

- Select all records from the 'striw2x_wqTCode' table where the 'g_iw2x_wq_planner' column that exists in the 'striw2x_wqTCode' table has data 'user'.

Instead of a predefined user code, the query will use the user code of whomever is logged into the application. The data returned will be the data associated with the user who made the request. And the records returned are limited to the single record referenced in the request, thus saving the user both time and easing the load on the server since only a single record will be returned.

Custom Table Scroll

It is possible to create custom tables in Offline, just as it is possible in GuiXT. Offline tables are constructed using edit fields and read and write functionality is implemented by using arrays to hold the values of the fields that constitute the tables.

Scroll functionality can also be implemented in custom tables. To do this, the following is necessary:

- An 'Up' pushbutton that will display the next record in the table
- A 'Down' pushbutton that displays the previous record in the table

Table Control Scroll

Most table controls in Offline can have scroll functionality added by using a For Loop in the code. To add scrolling functionality, we would add the following code to the existing script file:

```
for(var cRow = 0; cRow < objApprTableControl.GetRowNumber( ); cRow++)
{
// logic goes here
}
```

The preceding code specifies that Offline will begin scrolling from row 0 until the end of the table is reached. However, to do this, Offline needs to know how many

rows actually exist in the table. There is a special method that can do exactly this, for which the code is as follows:

```
GetRowNumber( );
```

To determine if a given row is or is not selected, we would use a second special method, as follows:

```
IsRowSelected( );
```

This second special method is actually a Boolean that will return either true or false depending on whether or not the row in questions has been selected by the user. In the following example, we show the code for our table:

```
var strAppTCode = "appr";
var objAppr = new SR3TransactionObject(strApprTCode);
var cArray = [ ];
var cCount = 0;
// ***********************************************
objAppr.addScreen([
{type:SCR_CAPTION,label:"Employee Approval"},
{type:SCR_GROUPBOX,labe
l:"Custom Table",position:{
row:0,col:0,height:9,width:33},fieldname:"g_scroll_tb"},
{type:SCR_EDIT,label:"Field 1",p
osition:{row:2,col:1,edcol:17,edlen:12},
fieldname:"g_op_scond_0",style:EDIT_STYLE_READONLY},
{type:SCR_EDIT,label:"Field 2",p
osition:{row:3,col:1,edcol:17,edlen:12},
fieldname:"g_op_scond_1",style:EDIT_STYLE_READONLY},
{type:SCR_EDIT,label:"Field 3",p
osition:{row:4,col:1,edcol:17,edlen:12},
fieldname:"g_op_scond_2",style:EDIT_STYLE_READONLY},
{type:SCR_EDIT,label:"Field 4",p
osition:{row:5,col:1,edcol:17,edlen:12},
fieldname:"g_op_scond_3",style:EDIT_STYLE_READONLY},
{type:SCR_EDIT,label:"Field 5",p
osition:{row:3,col:1,edcol:17,edlen:12},
fieldname:"g_op_scond_4",style:EDIT_STYLE_READONLY},
{type:SCR_PUSHBUTTON,label:"@
0H!",position:{row:6,col:30,width:4},
fieldname:"DOWN", fcode:"=FN_DOWN",
callback:function()
{
cCount = cCount +1;
```

```
this.m_CurrentTransaction.sendScreen(1);
}
},
{type:SCR_FUNCTION,funcname:FUNCTION_PBO,callback:function()
{
for(var i=0;i<13;i++) {
cArray[i] = i + i;
}
this.m_RealTimeScreen.SetElementValue("g_op_scond_0", SCR_EDIT, cArray
[cCount]);
this.m_RealTimeScreen.SetElementValue("g_op_scond_1", SCR_EDIT, cArray
[cCount+1 ]);
this.m_RealTimeScreen.SetElementValue("g_op_scond_2", SCR_EDIT, cArray
[cCount+2]);
this.m_RealTimeScreen.SetElementValue("g_op_scond_3", SCR_EDIT, cArray
[cCount+3]);
this.m_RealTimeScreen.SetElementValue("g_op_scond_4", SCR_EDIT, cArray
[cCount+4]);
}
}
]);
```

Refresh Table Control

It is possible that users may desire to display certain values based on pre-specified conditions in an Offline table control. To do this, the user will add the following line to the table:

```
vva01TableDefinition.linkedtableinfo = data_table + "^" + data_table +
 ".column = ' " + value + " ' ";
```

The complete table code utilizing this feature is shown below. In the example, we are calling the above function for the current screen. The data displayed in the table control will change depending on the entered conditions:

```
function fva01_get_materialdata( ) {
System.TraceOutput("INSIDE THE FUNCTION")
vva01TableDefinition = this.findTableDefinition( );
var objva01TableControl =
this.m_RealTimeScreen.findTableListControl(vva01TableDefinition.positi
on.top, vva01TableDefinition.position.left); // Find the table control
if(objva01TableControl != null) { // If the table control is not empty
objva01TableControl.RemoveControl( ); // Remove the object
```

```
objva01TableControl = null; // Null the object
vva01TableDefinition.linkedtableinfo = strva01TCode + "^" + strva01TCo
de + ".g_mat_sno = ' " + sordChg[0] + " ' ";
System.TraceOutput("vva01TableDefinition.linkedtableinfo = " + vva01Ta
bleDefinition.linkedtableinfo + "\n");
if(this.createTableListControl(vva01TableDefinition) == false)
{
return;
}
}
}
```

Unique solution ID: #2088
Author: Poojitha Reddy
Last update: 2021-06-03 13:57