Framework for WinDev (v25)

Created By:
Glenn Rathke
561.317.2270 – West Palm Beach, FL
www.wxtraining.net
grathke@sdcdev.net

A word of thanks to those of you who have purchased the Framework for WinDev, known as the Framework. We make every effort to keep the Framework and manual in sync. However, the Framework is our first priority to which we tweak the code and create or modify functionality, sometimes multiple releases are distributed weekly. From time to time you may notice the manual lags behind the updated Framework, rest assured we will try to make the manual follow as closely to the Framework as possible.

We consider the Framework a part of our training materials. As for our other training materials, we use the manuals and examples in every class we teach, whether it be here in West Palm Beach, FL or at your site.

Disclaimer:

Depending on the version of the Framework and manual we may make reference to a product called ThinfinityUI. We make no endorsements or recommendations concerning this or any other product. The code that is specific is ThinfinityUI is only to be used as an example in how to implement an Automation Object.

Soft Design Consulting, LLC
7803 Red River Rd.
West Palm Beach, FL 33411
www.sdcdev.net

Table of Contents

GETTING STARTED

Pre Requisites for Implementing the Framework

1. You are familiar with WinDev.
2. You know how to create windows, queries, and query parameters.
3. You know how to use the debugger for stepping through code.
4. A familiarity with indirection would be help but not necessary.

The Framework Purpose

The Framework arose out of a stated need from developers looking for an easy way to jump-start their application development. We have taken some of the major points we teach in our WinDev class as well as what we use in our own application development and wrapped them into a Framework.

The features such as login, CRUD (Create, Read, Update, and Delete) control template, business rules, and role based security come directly from our everyday development.

Why We Wrote the Framework

Besides using it in our own application development, we wanted to make it easier for the average procedural programmer to develop applications. Loosely speaking, the extensive use of the CRUD control template gives to us procedural programmers something similar to what OOP gives to object oriented programmers. Reusability is at the heart of the control template and the Framework.

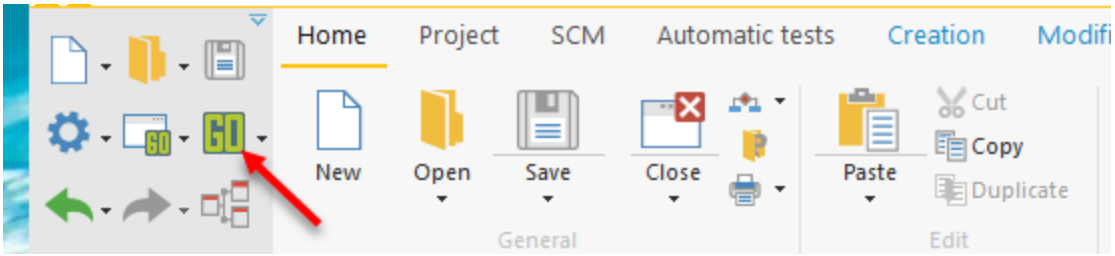
While we are going to cover some of the inner workings of the Framework let us get to the heart of the matter, how to create browse and maintenance windows without having to write the CRUD code. Also, for those windows where you implement a QBE (Query by Example), not having to write the QBE interface.

Four Very Important Points to Remember

1. Before we go further, it is important to know AND remember that anything you need to know about the user after they Login is in the QRY_CurrentUser_Get. The query has one row of data and is always current throughout the life of the running

exe. Naturally, it contains the user information, the company information, and the roles they belong to. The role field contains a comma delimited list of all roles.

2. As you test the windows you have created, you should test from the 'Run project test (Ctrl + 59)' not the 'Test Window (F9)'. As mentioned in #1, you need to proceed through the Login window and code.



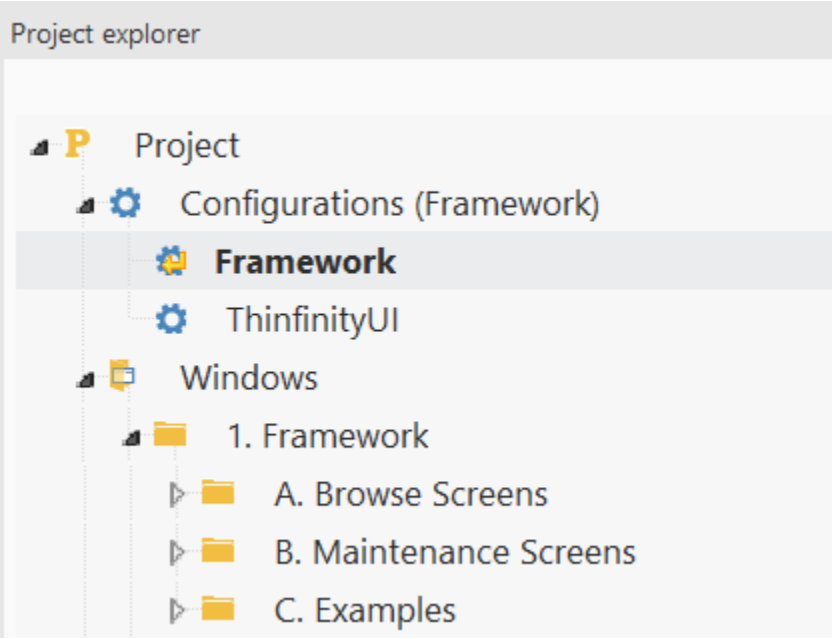
3. If you want to have any tables take advantage of the SaaS feature of the Framework, be sure to include a field named CompanyID (int). The value will automatically be populated.

4. Take advantage of the Examples as they are fully functional. They can be found on the Examples tab of the ribbon bar in the Main Menu

The Framework Layout

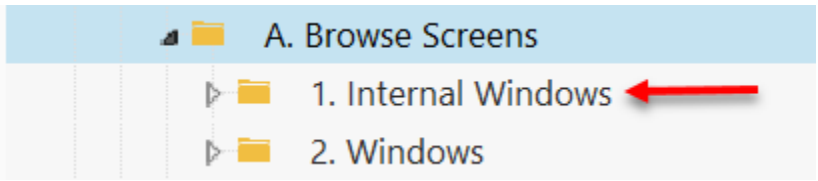
The Framework Layout

The Framework layout consists of 3 separate areas, Browse Screens, Maintenance Screens, and Examples.



Browse Screens

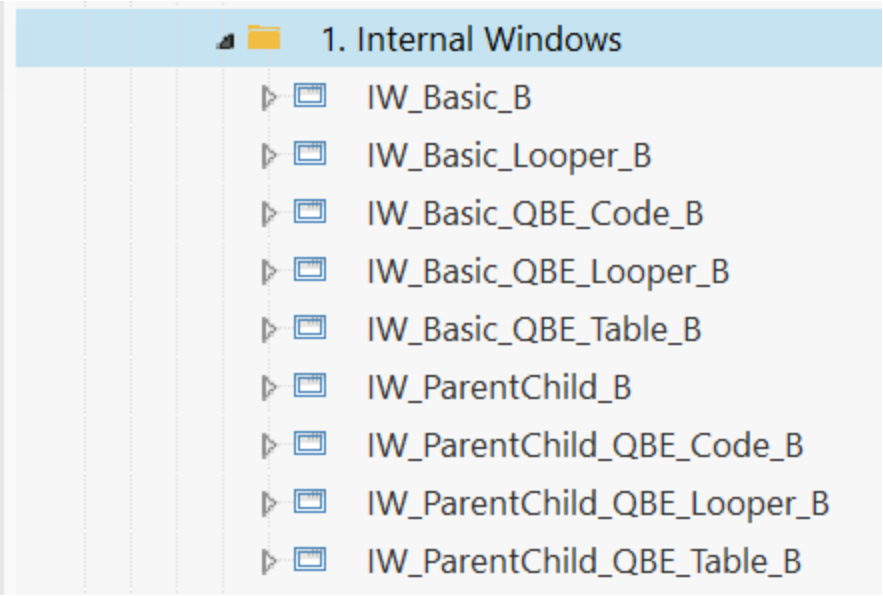
Opening the *A. Browse Screens* reveals two different types of Smart Windows, Internal Windows and modal Windows.



Internal Windows are those windows that are contained in an internal window control and switched out at runtime ChangeSourceWindow(). Modal windows are windows that are opened by using the Open() command. You either or both types if windows in you application based on the Framework.

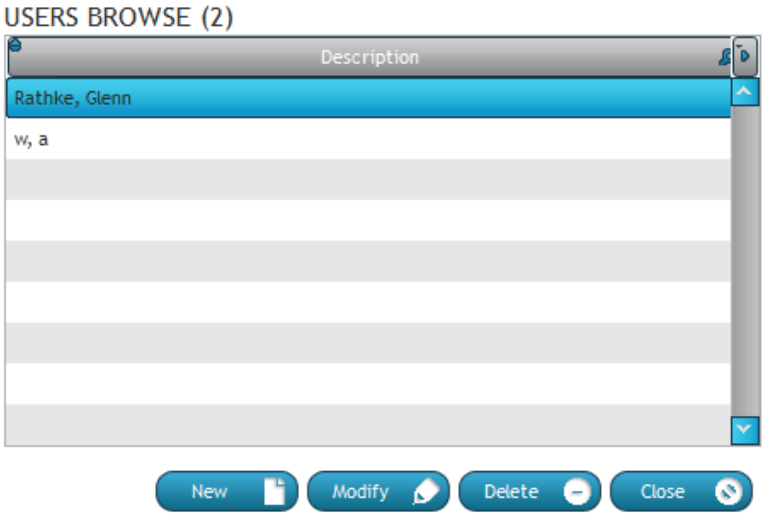
Basic Windows

Opening *1. Internal Windows* reveals *Basic*, and *Parent Child* windows as displayed next.



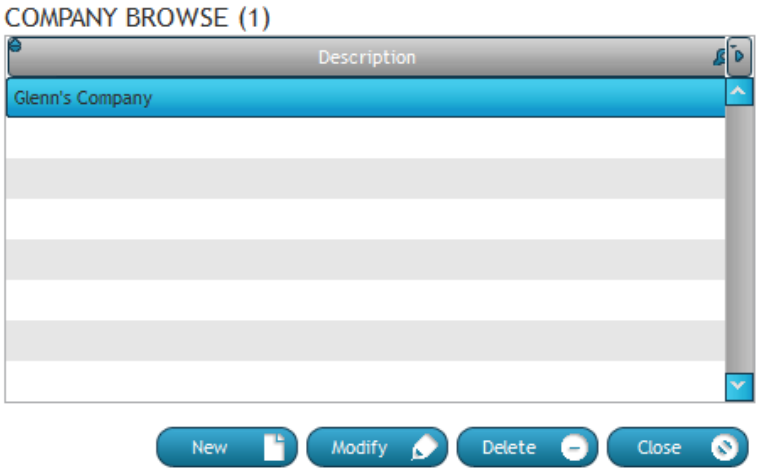
Basic windows also referred to as modal windows are those windows with a single table control or looper on the window for displaying the data. By assigning the values to a few variables the data will automatically be displayed without having to write any code. Of course, if you want to write code to extend or change the functionality you can.

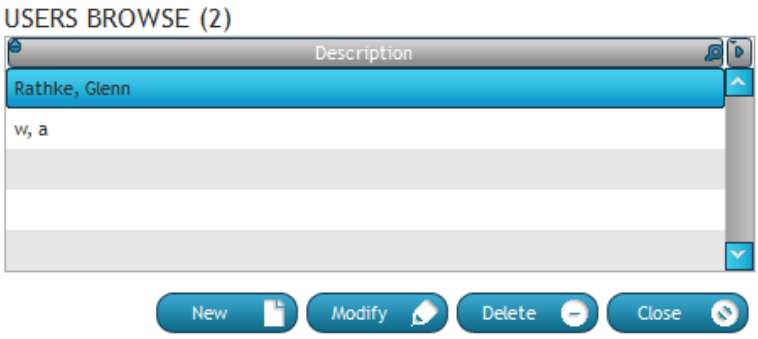
As a side note, both Basic and Parent Child windows are also available with a QBE. The QBE can either be the traditional type where you place controls and code them to use the parameters of a query. Or, they can be codeless if you select a table control or a looper to prompt the user for values. Simply select the functionality you want. The Examples section (covered later) makes it easy to see the different types of Smart Windows available.



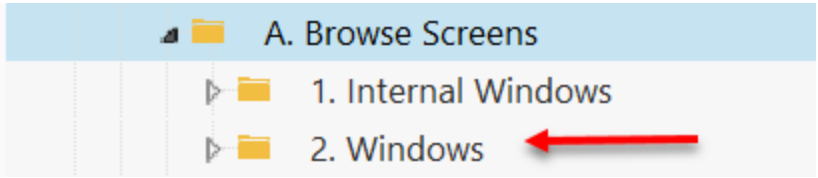
Parent Child Windows

Parent Child windows, also known as master detail windows are those windows with two table controls where you select a parent record from the first table control, and the children records are displayed in the second table control automatically.

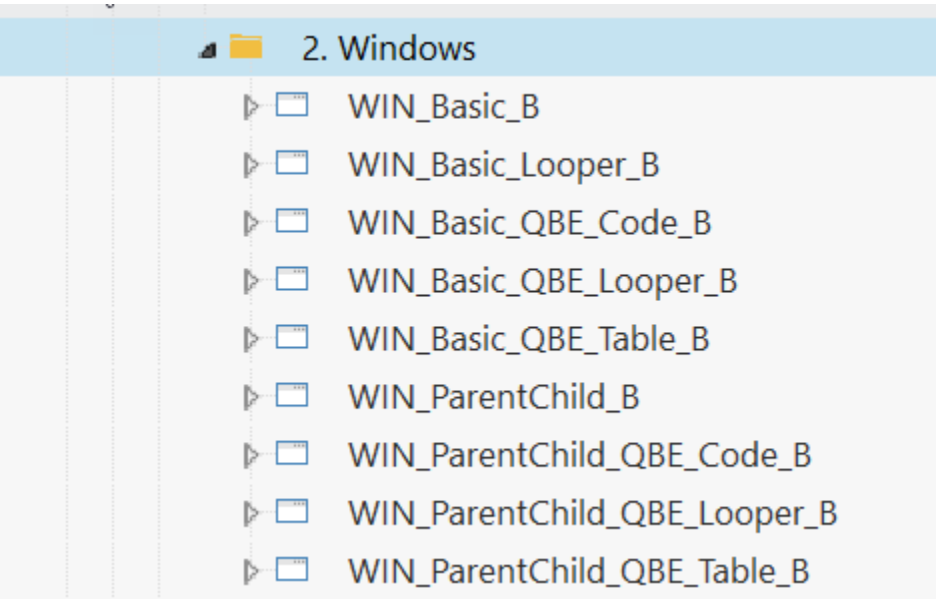




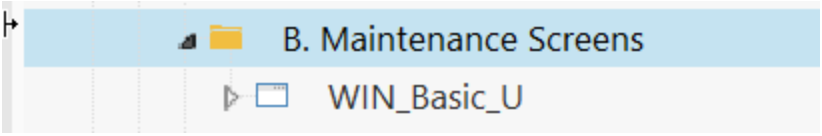
As previously mentioned, if you want to write code to extend or override this functionality you can.



Opening *2. Windows* reveals windows with the same Basic and Parent Child functionality as above, but instead of being internal windows, they are modal windows.



Maintenance Screens

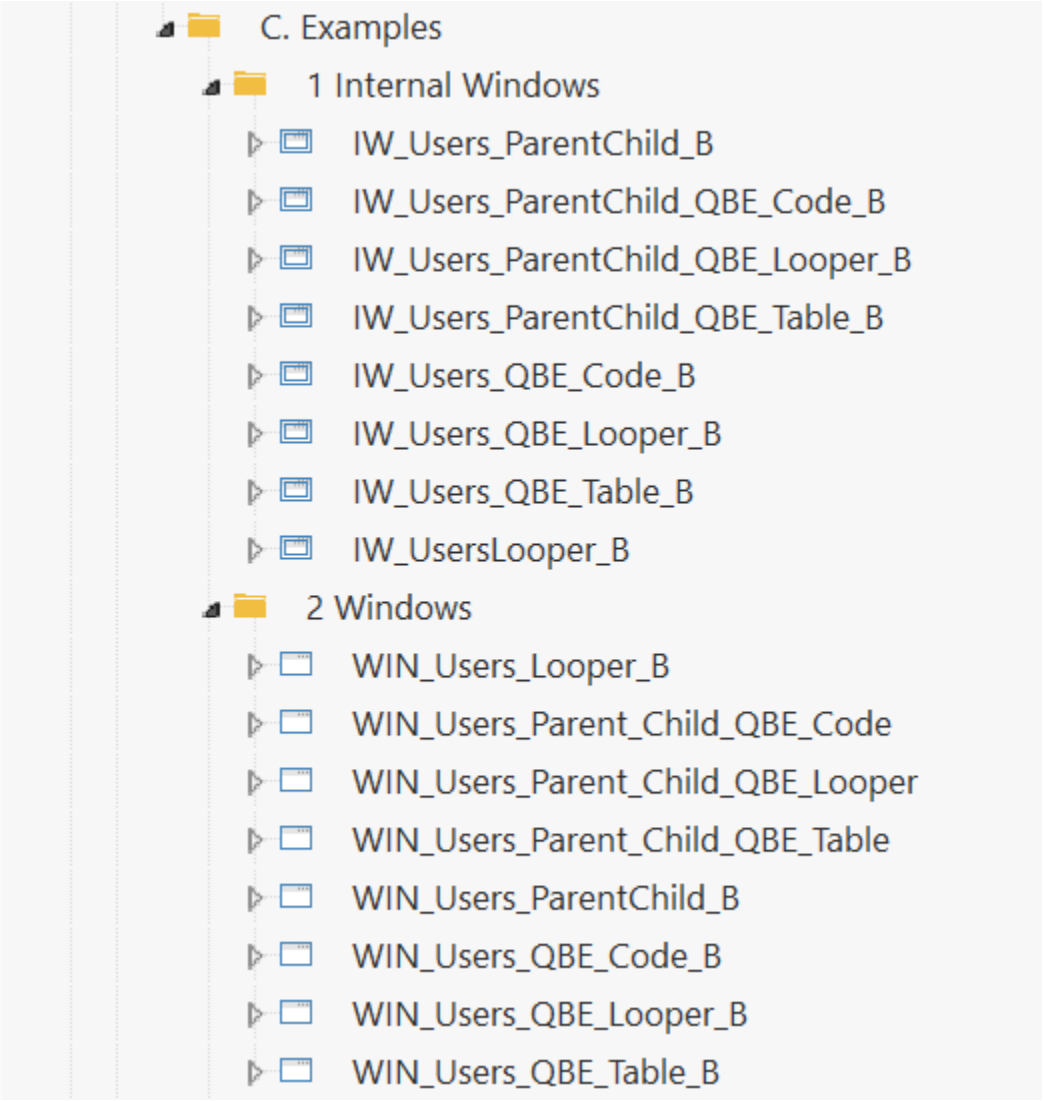


The Smart Window, WIN_Basic_U is the purely generic window where you place any controls needed for creating a maintenance window. You may make this window as simple or as complex as needed for your app. The validation code for saving the record is already coded for you in the template.

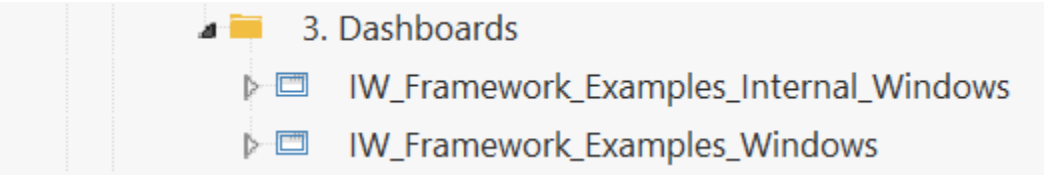
Examples

The Examples section contains examples of all the different types of windows. As you run through the examples, remember that each window is functional because of the variables that have been assigned values. There is absolutely no hand code written for the windows. But, and this is a big BUT, you are free to extend or override anything in the window. In fact, once you get proficient with the Framework you can modify the Framework to your needs. As long as you are not using the Framework as a teaching aide, a teaching example, you are free to use the Framework in any of your apps. There are no royalty fees.

Each example below started by taking a window from section *A. Browse Screens*, doing a 'File Save As', connecting the table control and columns to a Query, and assigning a few variables.



Dashboards

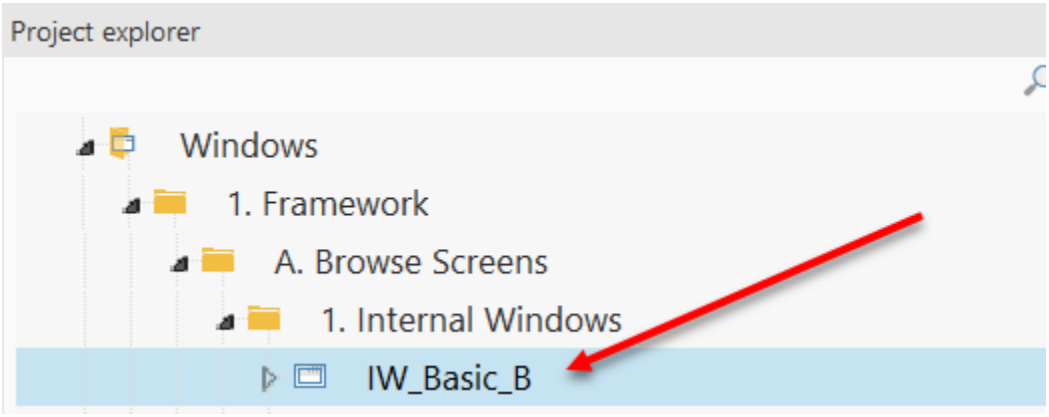


The dashboards are merely used to run the examples.

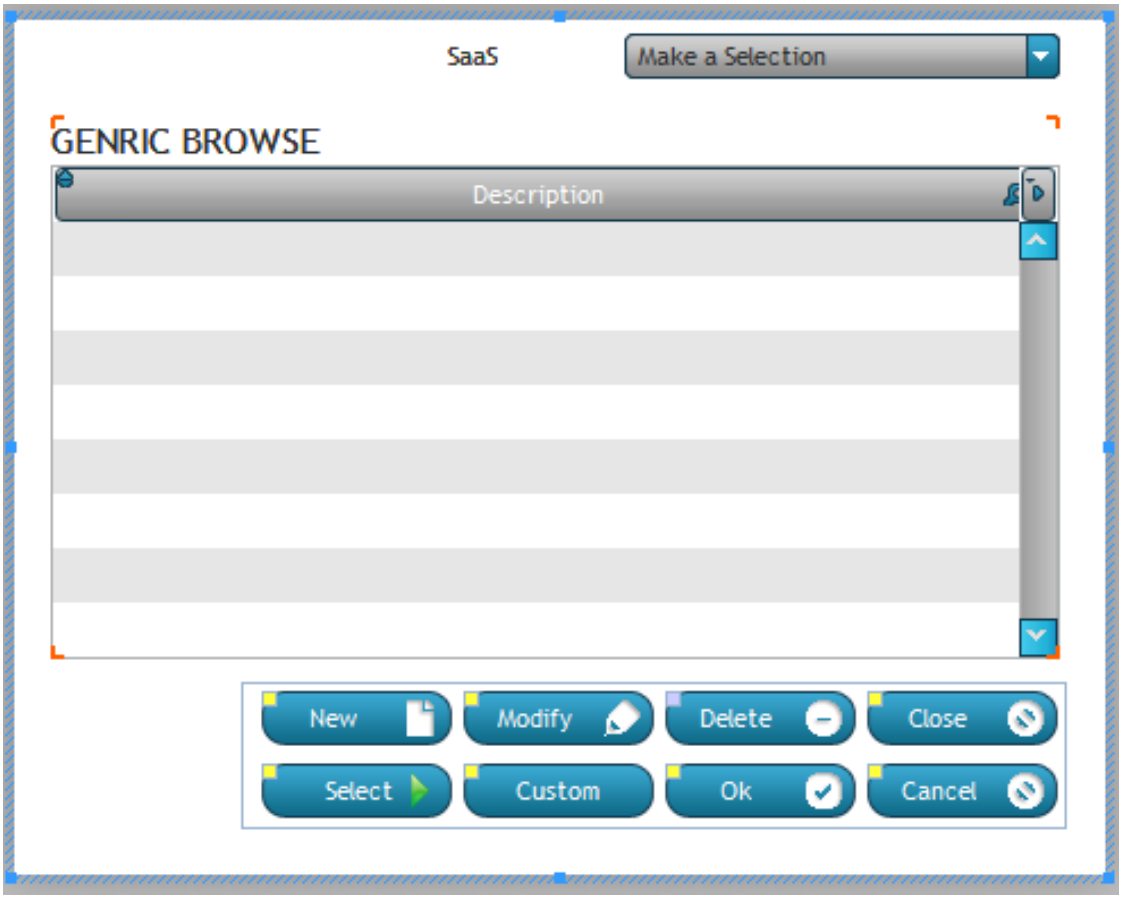
Ready to get started creating some functionality?

CREATING THE WINDOWS

If you have not done so, load the Framework project into WinDev. From the Project explorer select the 'IW_Basic_B' window as displayed next:



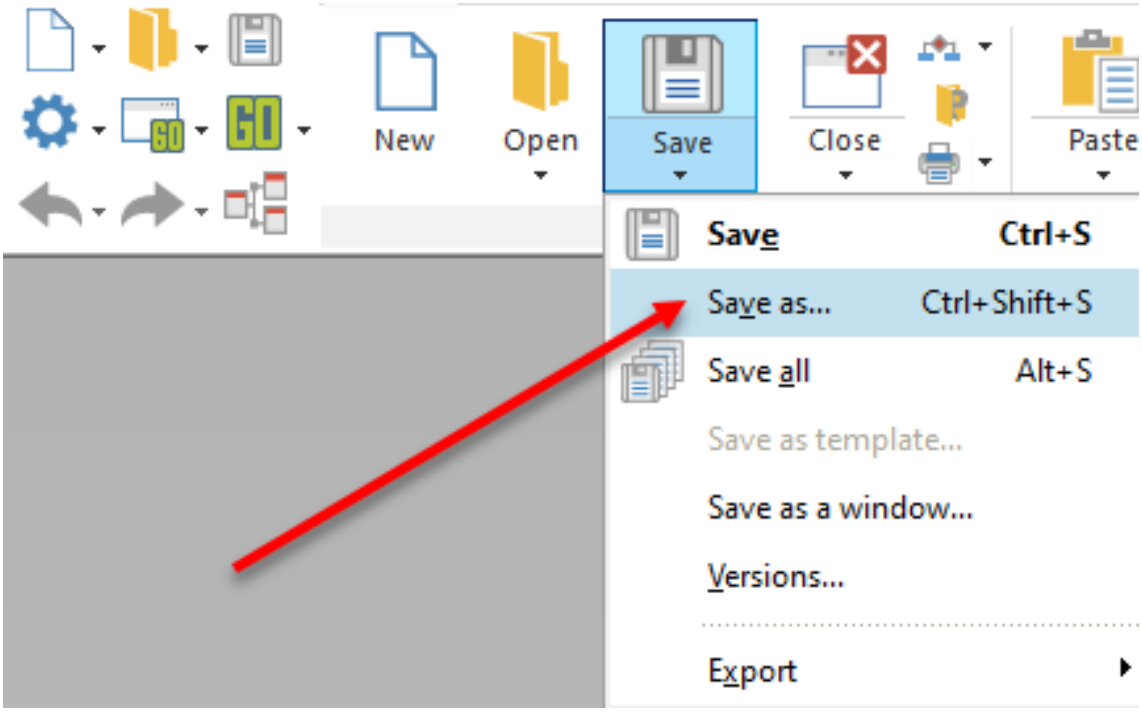
Open the window so it appears similar to the next graphic.



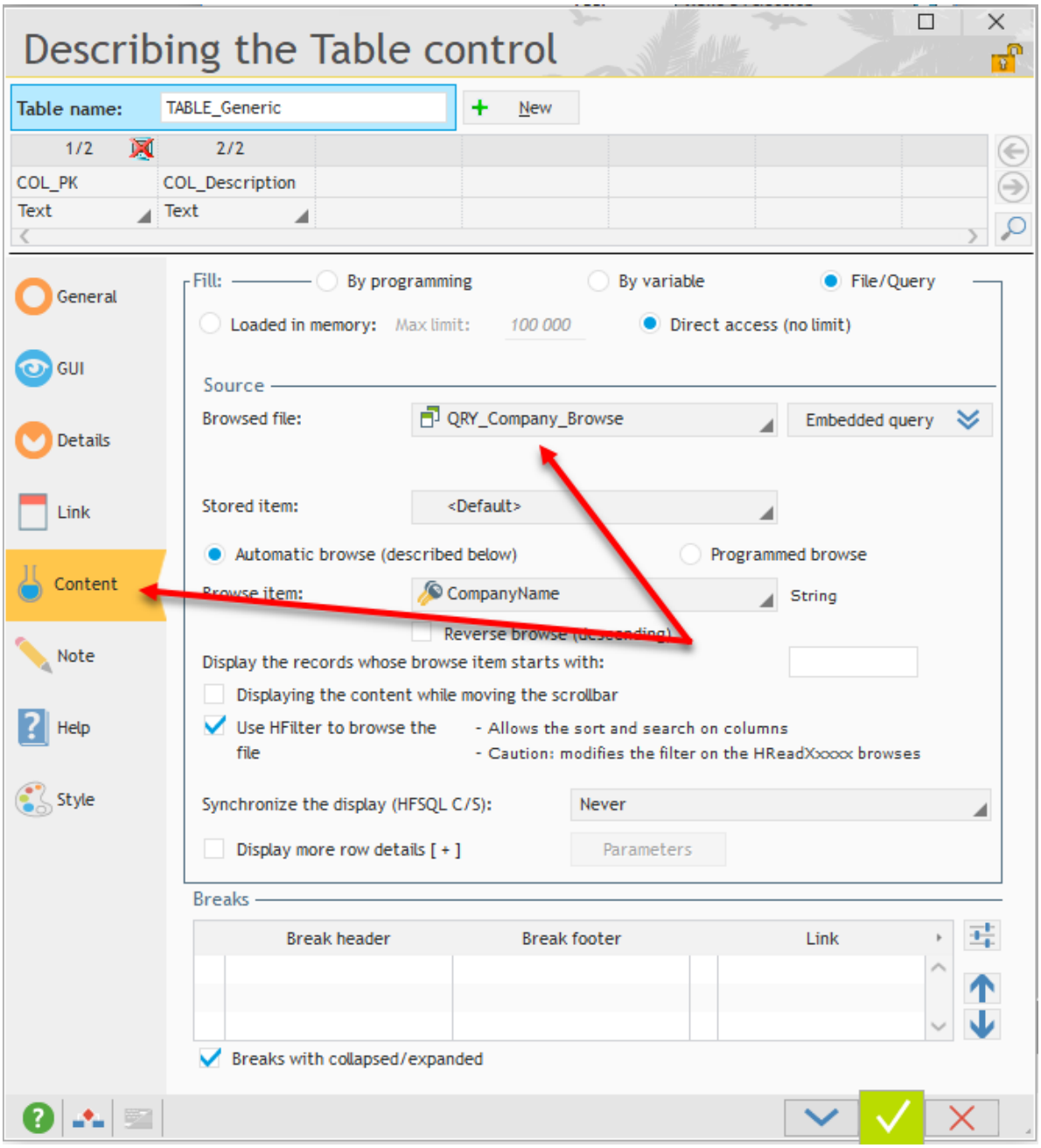
There are three main areas of the browse window. In fact, all three generic table control windows share the same main areas. The dropdown combo box at the top known as the Saas combo, the generic browse table control. And the CRUD template at the bottom. The CRUD template is where all the 'magic' happens. Through the heavy use of

indirection, the 5 or so variables that you will assign values to will direct the control template to handle all the CRUD activity.

To save the window as the starting point for the development of your new window, select ‘Save as’ as displayed next and give the window a meaningful name.



Next, open up the dialog describing the table control by double clicking on the table control. Select the ‘Content’ tab. The dialog appears similar to the next graphic.

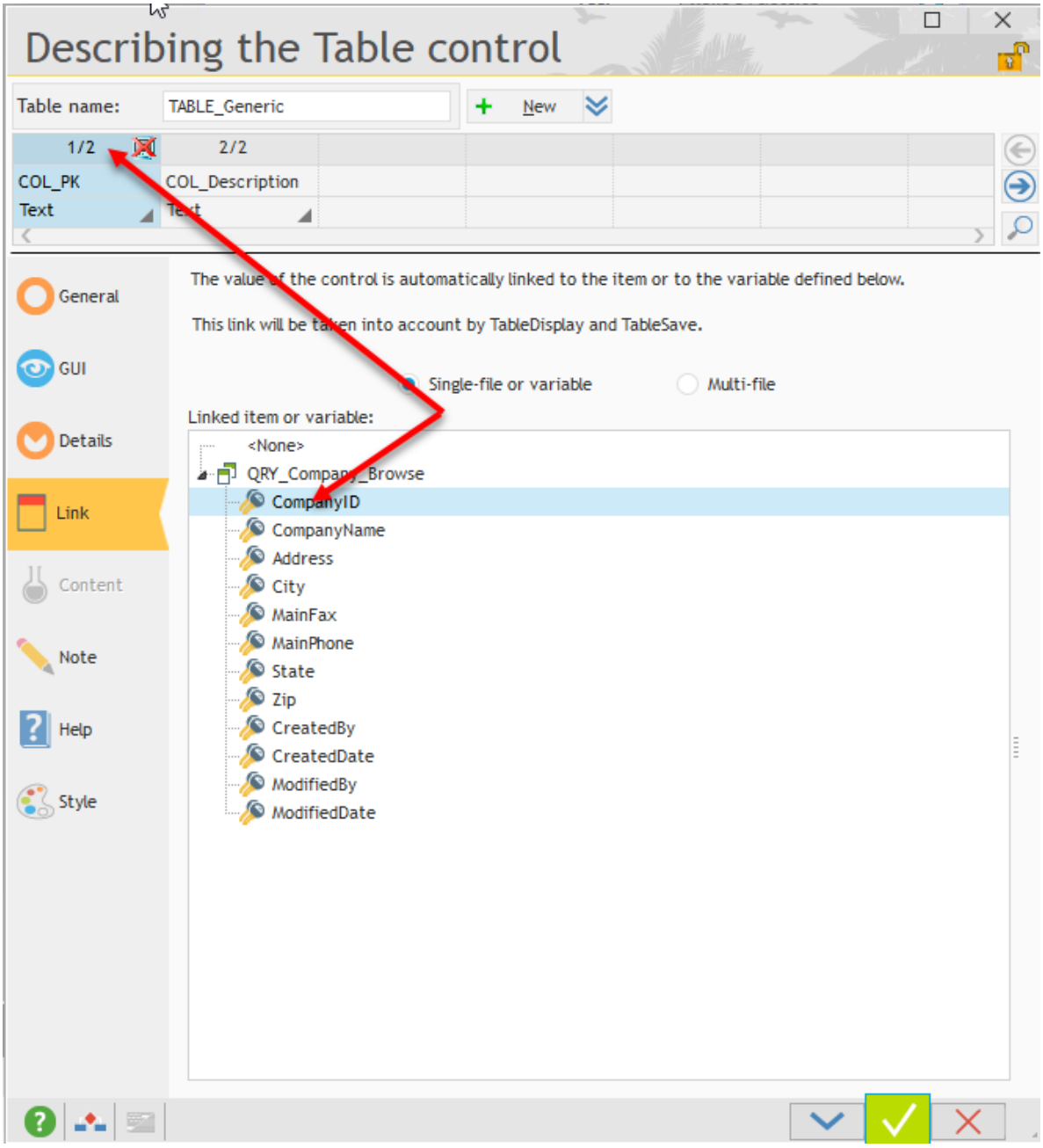


The secret to returning data as quickly as possible is to use a SQL statement. However, you would have to write a bit of code and the object of the Framework is to write as little code as possible. Or, use a query developed in the Query Editor. The Framework leverages the advantages of using queries created in the Query Editor; you can use the query elsewhere in the application and you can have as many Query Parameters as needed to take advantage of implementing a QBE with little to no additional coding. Remember though, this is a project and not a black box, you are free to write additional code or make any changes to the existing code.

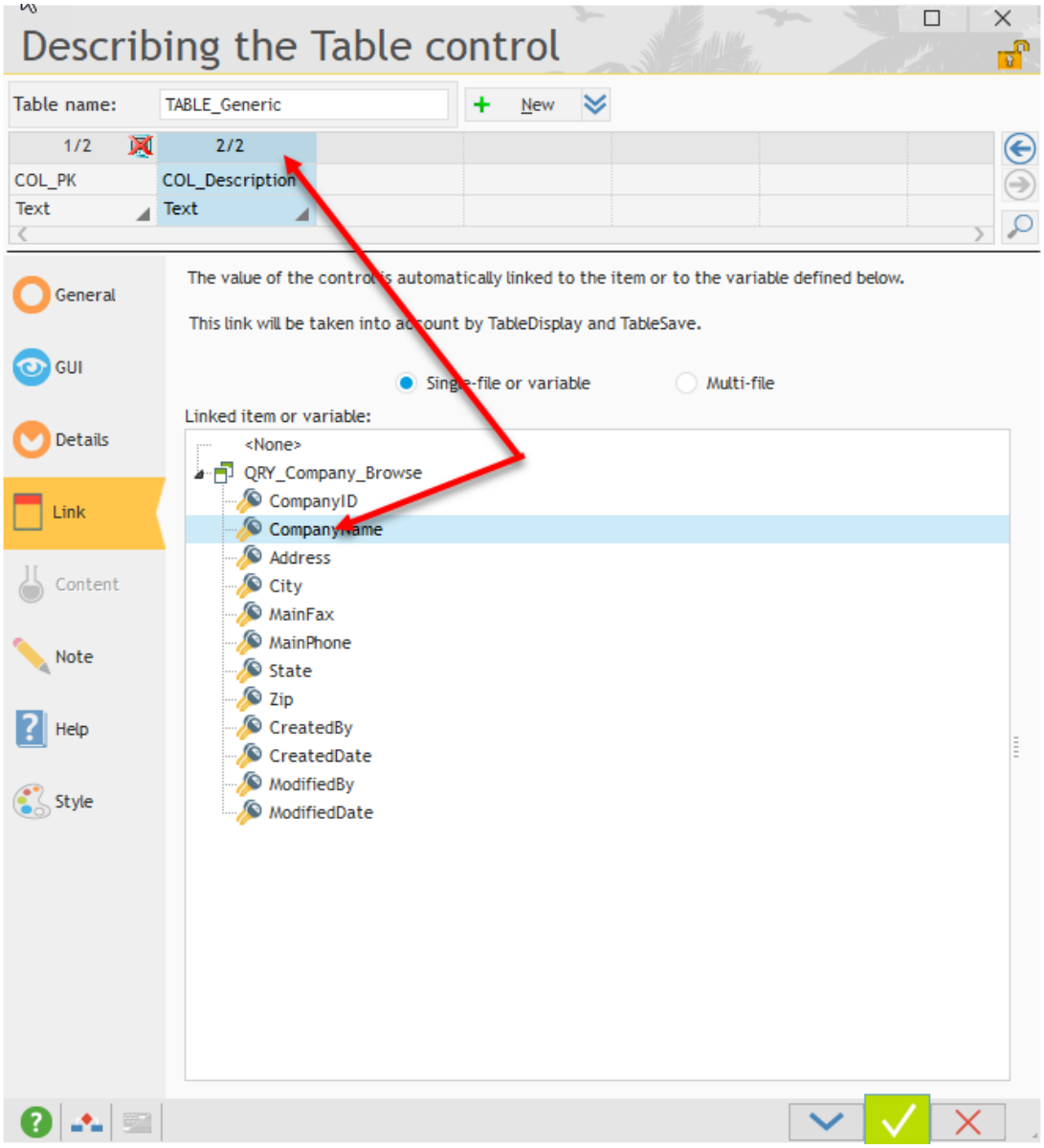
At the Browsed file dropdown, select a query that will return the records to be displayed in the table control. In this example we are using the 'QRY_Company_Browse' query. It is required to have at least 1 parameter in the query. This parameter is used to return

records in case the dropdown combo at the top of the window is not being used. There are 2 reason why the dropdown combo would not be used. One, you removed it from the window, or two, the role for the logged in user is not defined so they can 'View all Records'. We will cover roles later. For now, keep dropdown.

Next select the 'Link' tab as displayed.



And connect the table columns to the appropriate fields in the query.



Save the table information and close the window dialog. This is only our example window. In your case, you will select the query, query fields, and table columns as needed.

Next, we are going to assign values to the variables discussed earlier. Double click anywhere in the gray area outside of the windows or press the function key F2.

The code appears below:

```
Global declarations of IW_Users_B                    If Error: by
PROCEDURE MyWindow(LOCAL gsIW_Name is string = "")

NextTitle(csDialogTitle)

gbFirstRun      is boolean = True

IF InTestMode() THEN // Display the primary key column in test mode
    COL_PK..Visible = True
    TABLE_Generic..Width += COL_PK..Width
END

End of initialization of IW_Users_B

Do_Housekeeping() // Do any custom coding here

Closing IW_Users_B

sQry is string = {WD_CRUD.CRUD_BrowseTable, indControl}..BrowsedFile
HDeactivateFilter(sQry)

Request for refreshing the display of IW_Users_B
```

Place your cursor on the Do_Housekeeping procedure and bring up the code by pressing F2.

Change the code displayed above so it appears as below.

```
//IF InTestMode() THEN
//  Info ("DO NOT FORGET TO INITIALIZE THE CRUD ACCESS.") // Delete this when the
//END

WD_CRUD.CRUD_DbTable                        = Company..Name // e.g. Generic..Name
WD_CRUD.CRUD_DbPrimaryKey                   = Company.CompanyID..Name // e.g. Gene
WD_CRUD.CRUD_BrowseTable                    = TABLE_Generic..FullName
WD_CRUD.CRUD_BrowsePKColumn                 = TABLE_Generic.COL_PK..FullName
WD_CRUD.CRUD_DefaultQryParamAndValue        = "QRY_Company_Browse.Param_Company_ID
//WD_CRUD.CRUD_SaaSComboName                = "" // If you have not changed the na
WD_CRUD.CRUD_SaaSQryParameterAndValue       = "QRY_Saas_Company_DropDown.Param_Saa
WD_CRUD.CRUD_MaintenancePage                = "WIN_Company_U" // e.g. "WIN_Basic_U
//WD_CRUD.CRUD_ReturnWindowOverride         = "" If you are going back to the prev
WD_CRUD.CRUD_PreviousBrowseWindow           = gsIW_Name
WD_CRUD.CRUD_UsedForReporting               = False // Hides New, Modify, and Dele

WD_CRUD.CRUD_Apply() // Apply the values from above as well as the security level
```

WD_CRUD.CRUD_DbTable is the table in the analysis that you will read and write the data. It is most likely the same as the table in the From Clause of the query on the Content tab of the table control.

WD_CRUD.CRUD_DbPrimaryKey is the primary key of the table you used in the above variable.

WD_CRUD.CRUD_BrowseTable is the name of the table control in the window. If you are going to use the default name of the control, you do not need to change it.

WD_CRUD.CRUD_BrowsePKColumn is the name of the table column that you have linked to the primary key of the table. If you are going to use the default name of the control, you do not need to change it.

You will notice the values of the first 4 variables are in the format xxxx..name, using the name property alerts you to any problems with the names not being recognized by the compiler.

WD_CRUD.CRUD_DefaultQryParamAndValue is used as a default query parameter and value to be used when the Saas combo does not come into play.

WD_CRUD.CRUD_SaaSComboName is used as the name of the SaaS_Combo. The global value sCOMBO_Saas is used globally. Hopefully, the name of the control and the value in the global variable are the same. If not, you can override it here.

WD_CRUD.CRUD_SaaSQryParameter is used to limit the records in the Saas combo.

WD_CRUD.CRUD_MaintenancePage is used to call the maintenance window.

WD_CRUD.CRUD_ReturnWindowOverride is used to call a different page to return to.

WD_CRUD.CRUD_PreviousBrowseWindow is the page you came from. No need to change the value.

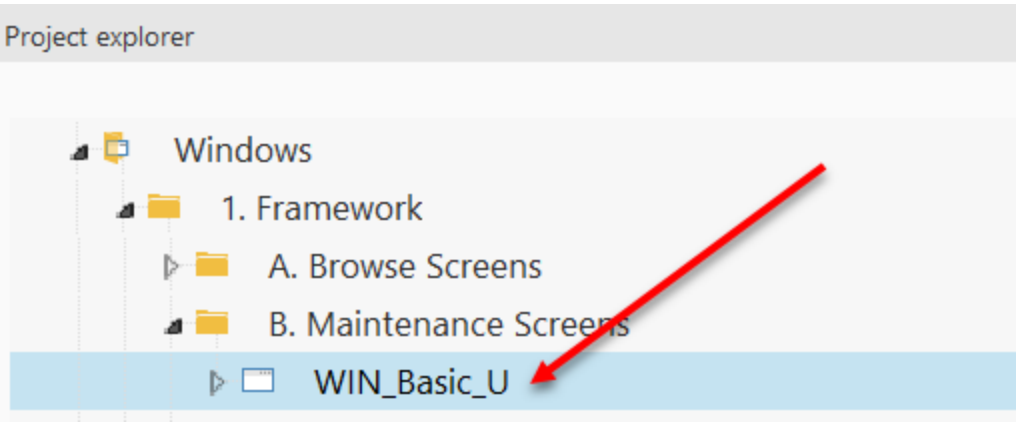
WD_CRUD.CRUD_UsedForReporting is set to True if the window is only used to call Reports

That is all it takes to create a fully functional browse window.

To recap:

5. We opened the 'IW_Generics_B' browse window and did a 'File Save As'
6. Connected the table control to a query on the Content and Link tabs.
7. Assigned values to 5 variables.
8. Saved the completed window.

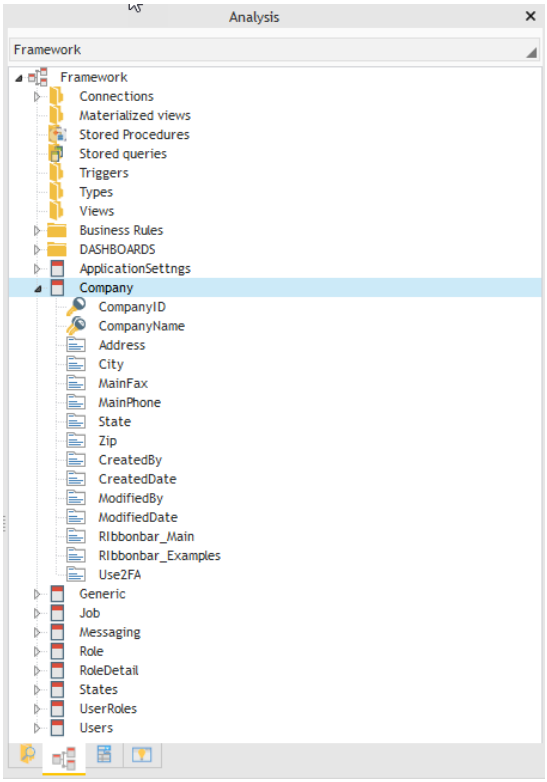
Similar to what we had done with the browse window, select the 'WIN_Basic_U' window from the Project explorer as displayed next and do a 'File Save As'.



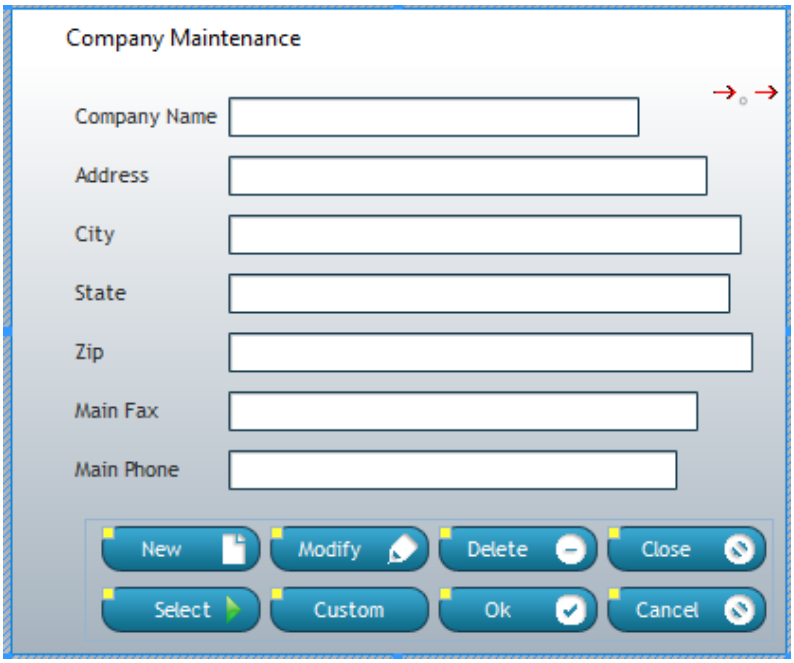
When saving the window, this is the name of the window that should be used when you assigned the window name to the variable `WD_CRUD.CRUD_MaintenancePage` above.

When saved, the window will appear similar to this:

Next drag the fields from the Project explorer as displayed below



The window should look similar to this, you may use fewer fields.



Save the window.

The maintenance window is now complete.

To recap:

1. You opened the ‘WIN_Basic_U’ maintenance window and did a ‘File Save As’
2. Dragged or created the fields that are to be displayed on the window.
3. Saved the completed window.

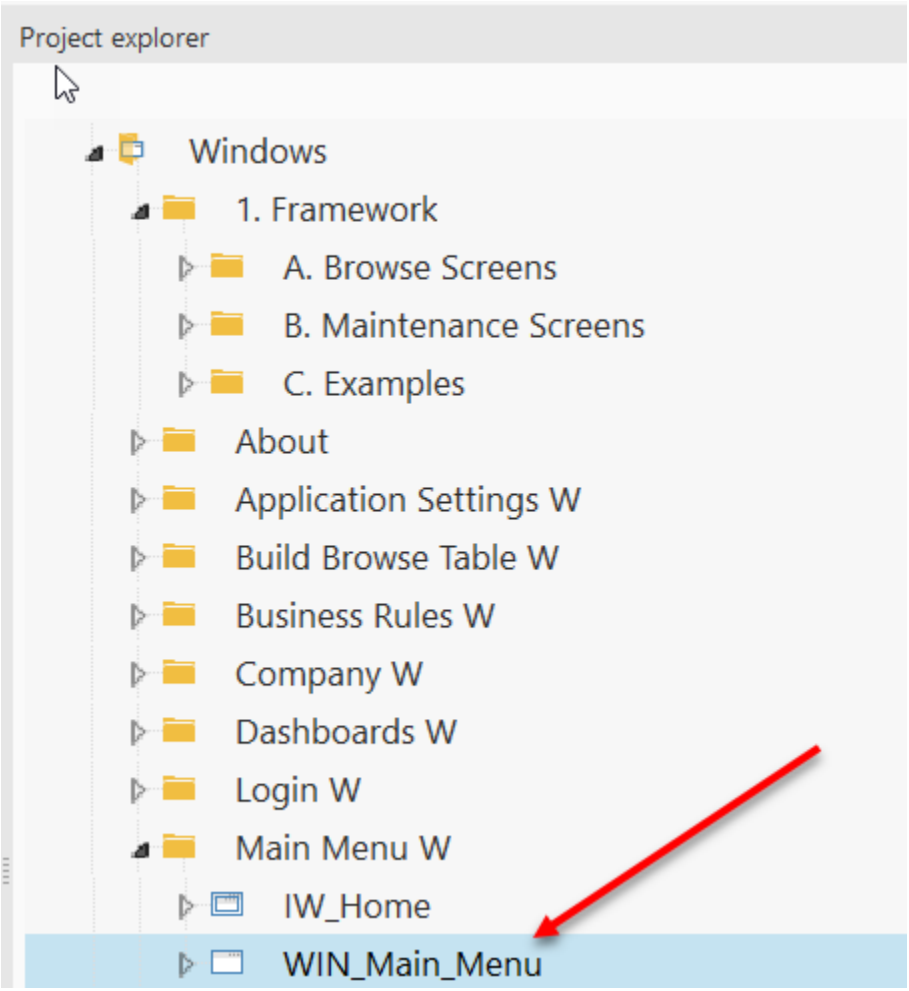
That is all it takes to create a pair of browse and maintenance windows. Your windows will be different, but the steps remain the same.

USING THE RIBBON BAR

The ribbon bar is used in place of a traditional main menu. It offers several advantages, it has a more visual appeal. When implemented with dashboard windows, you are never more than a few clicks away from display additional windows.

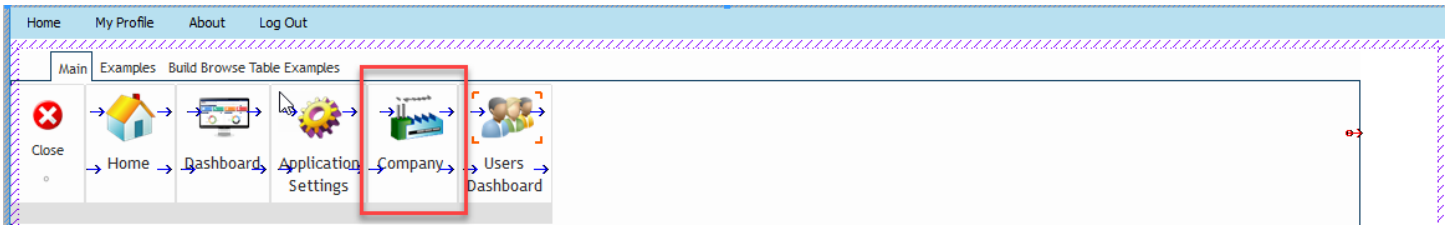
The example windows you created were similar to the Company windows already in the project. Since the Company browse window is already being called by a menu choice, really a ribbon bar choice, let us look to see how it is called.

Open the Main Menu from the Project explorer

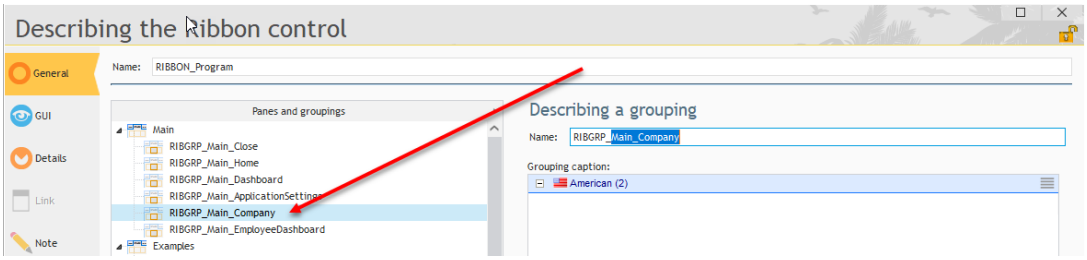


Let us look at how the ribbon bar is used to call the windows and subsequent dashboards. Verify the 'Main' tab is selected. We will concentrate on the highlighted area.

There are 3 components that make up the area. A ribbon bar group that is the container for the graphic and static text.



Double click the ribbon bar in order to display the tabbed dialog below.



The RIBGRP_Main_Company is the ribbon bar group that is mentioned above as a container.

Based upon the user role when logged in, the group can either be made to hide/display or make gray/active.

Select the Company image and look at the code.

When the control is initialized the following code is executed.

```
Initializing IMG_Main_Company

sVisibleOrGray is string = "Visible"
//sVisibleOrGray is string = "Gray"

RoleAccess_Get(RIBGRP_Main_Company..Name, Company..Name, sVisibleOrGray)
```

Behind the scenes the role or roles for the logged user is validated against the Company..Name table. If the user has access to the table, the ribbon bar group will be visible. If they do not have access, the group will be hidden. If you only wanted the group to be grayed, simply use that line of code.

To recap the ribbon bar group, the `RIBGRP_Main_Company..Name` group is acted upon in some fashion based upon the users role of the `Company..Name` table. The `sVisibleOrGray` determines if it is Visible or Gray.

Looking at the click event of the image we see:

Click **IMG_Main_Company**

```
// True if called from the Ribbon Bar, False if from a SubMenu Dashboard
gbCameFromMainMenu = True

gsCurrentDashboard = "IW_Companies_B"
Window_Display(gsCurrentDashboard)
```

The variable `gbCameFromMainMenu` directs window we are about to call to the main menu. The next lines call and open the window.

Look at the code behind the User Dashboard image. Rather than directly open the User browse window, it calls another window with additional choices. We call this a dashboard.

Even though it is not calling a browse window, the code is similar to above.

Click **IMG_Main_UserDashboard**

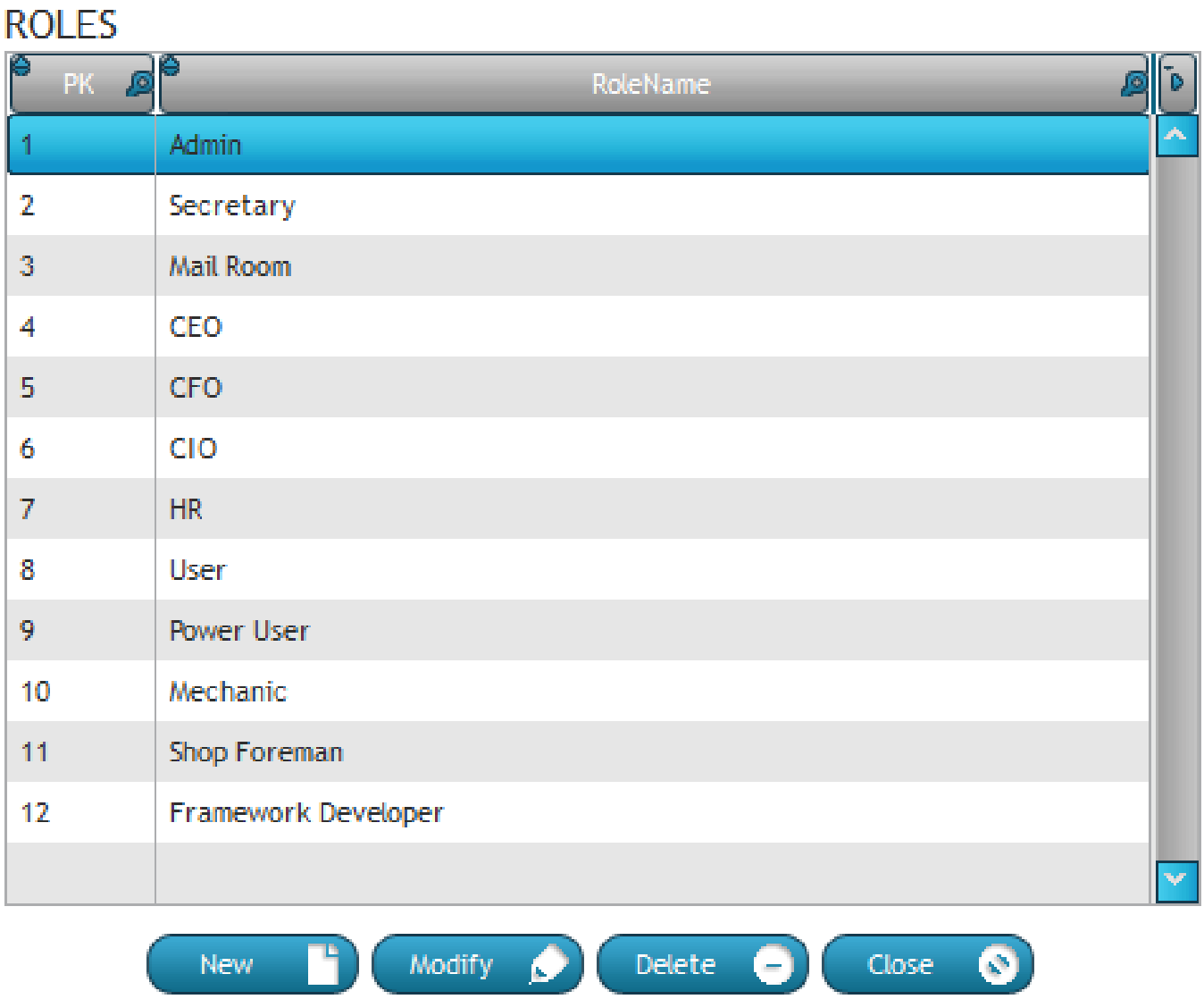
```
// True if called from the Ribbon Bar, False if from a SubMenu Dashboard
gbCameFromMainMenu = True

gsCurrentDashboard = "IW_Dashboard_Users"
Window_Display(gsCurrentDashboard)
```

I am sure you noticed multiple tabs on the ribbon bar. Each tab can be used however you see fit. In our production app, each tab represents a product of suit of products. Each tab can be turned on or off at the Company level on the WIN_Company_U maintenance form.

ROLE SECURITY

Roles and Their Scope of Access

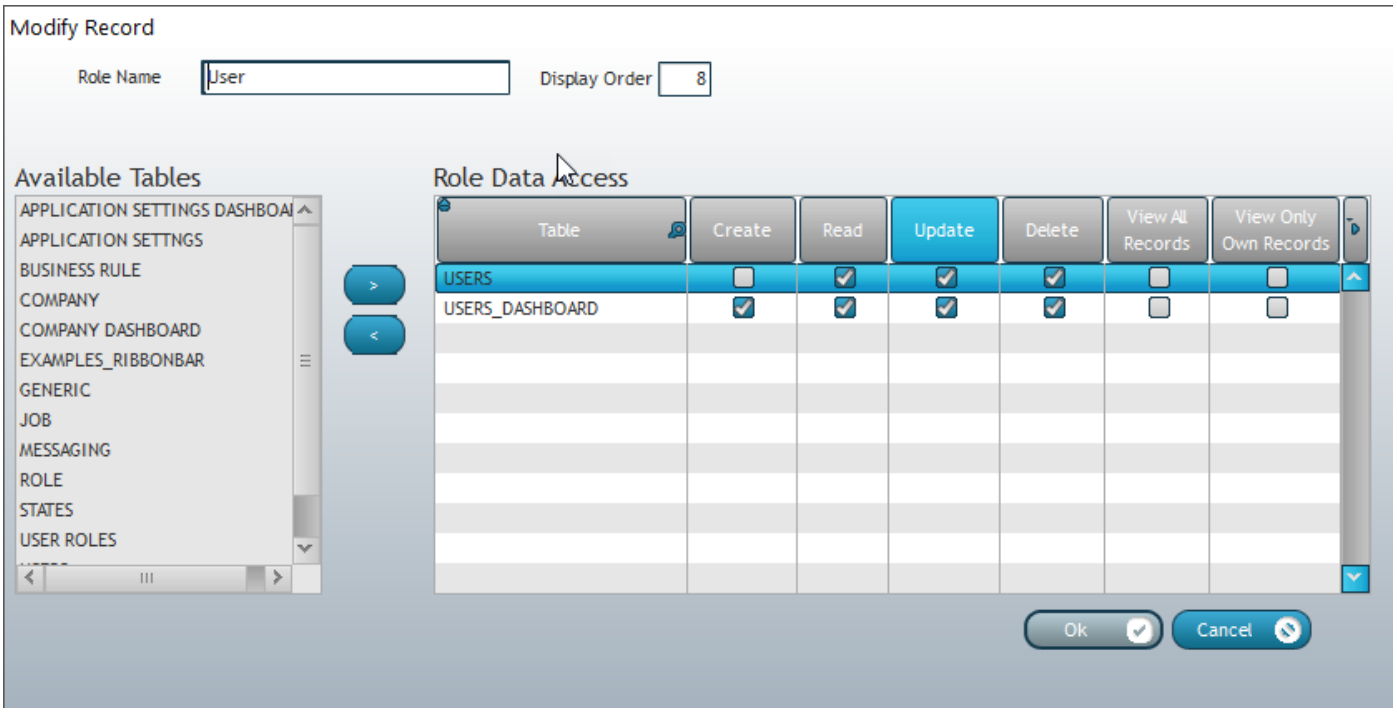
Security is based on Roles assigned to each user. Roles at their highest level are used as the parent in a parent / child relationship. Examples of roles could be User, Admin, and Framework Developer. There could be more or less roles based upon your needs. They are up to you to define. Each role can have access to one or several tables.



The definition of roles and how they are assigned to users is also a Role. That way only specific people can assign roles.

Let us look at an example of the roles mentioned above where User is the most restrictive. Or another way of looking at it, has access to the least amount of information. Perhaps the User only has access to the User's table. The level of access

they could have with regards to the User's table is Create, Write, Read, Delete, and View all Records. Each of these levels is activated or deactivated by selecting the appropriate checkbox.



Notice how the User has access to the User Dashboard as well as the User table. The reason they have access to the dashboard is so they can have access to the User browse on the dashboard. The access to the User table comes into play when the users go to the dashboard. The same role validation code we saw in the ribbon bar is used whenever a menu choice is chosen.

In the above graphic, notice that the 'Create' and 'View All Records' attributes are turned off. When the user logs in, on the Main Menu / Ribbon Bar they will see that they have access to the User Dashboard. Upon selecting the User Dashboard they will be taken to the dashboard and because they have access to the User table they will also see that they can select either of the choices relevant to the User.

When the User choice is selected they will be taken to the browse window for Users and there they will find that the Saas combo is not available to them to select different companies, and the 'Create' button is not available.

Naturally if you want to further restrict their access, you can uncheck the 'Write' and 'Delete' access. Leaving the 'Read' attribute checked will only let them select a record for viewing, but unable to save any changes. Not having any items selected results in the item not being available in the Main Window or subsequent dashboards.

Assigning Roles to Users

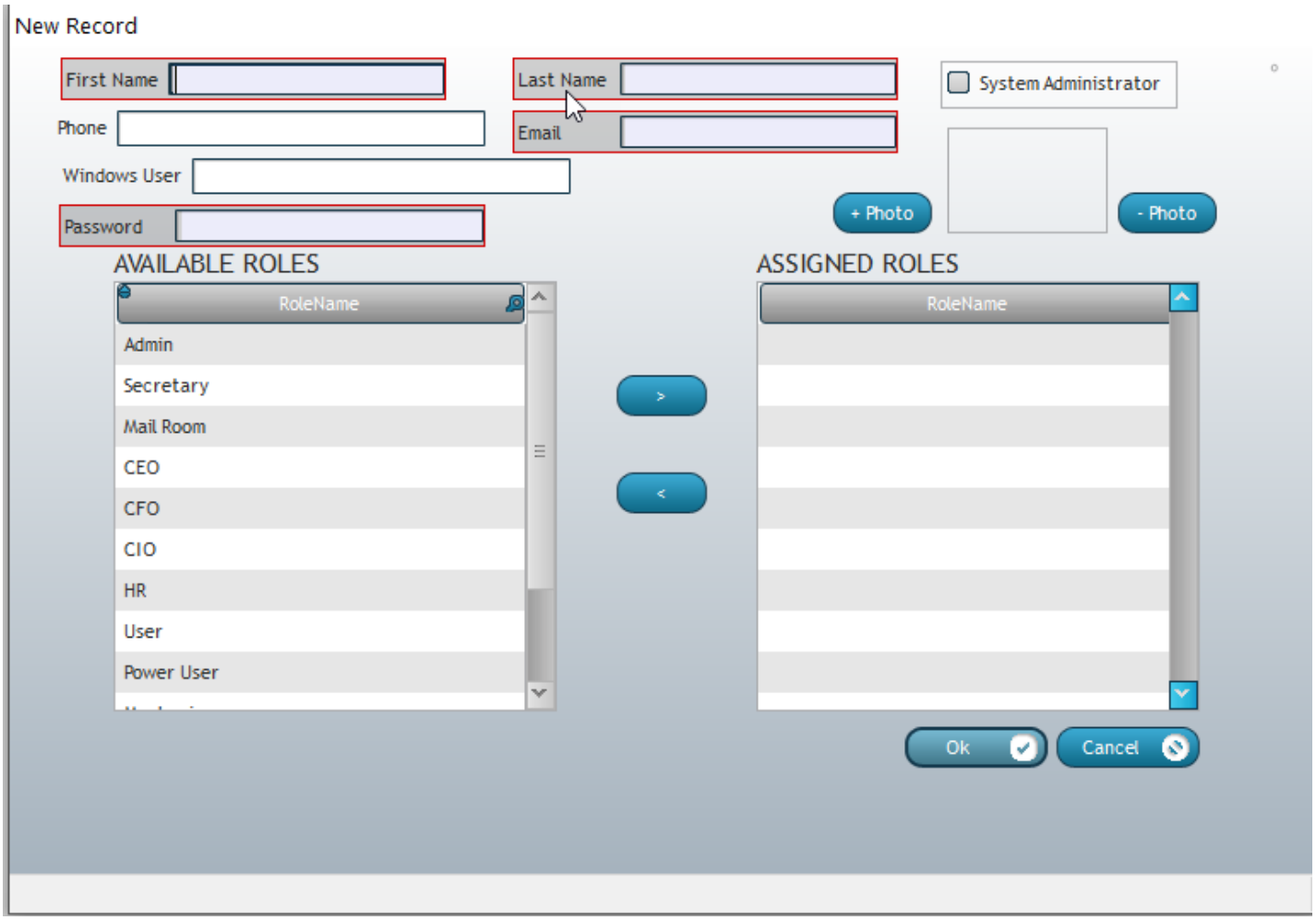
Assigning roles to users is easy. At the User maintenance form you will see an 'AVAILABLE ROLES' table and an 'ASSIGNED ROLES' table. To assign a role, select a role from the available roles and select the button with the right arrow. To remove a role, select the role from the assigned roles and select the button with the left arrow.

This User maintenance form is a special window in the application. It was originally created using a sample window and following the steps we have already covered. However, as you can tell there is additional functionality. It is highly recommended not to delete this window as user security is defined here. If additional or different functionality is required, I suggest adding fields here or make a separate maintenance form to handle your needs.

BUSINESS RULES

Field Level Business Rules

In their simplest form business rules pertain to some form of validation or action. We offer validation at the field level and at the window level. In the User maintenance form we just mentioned, there are some rather rudimentary field level business rules. The window below shows the required fields upon creating a new User record.



When saving a record, our implementation of business rules does not prohibit you from tabbing off a required field as is the default action of required fields by PCSoft. Instead, we offer what we call Soft Validation. This soft validation uses our version of an eYe Magnet but allows you to move around the form freely. As soon as you enter a value into a required field, the eYe Magnet disappears. If you clear out a value in a required field the eYe Magent re-appears. Upon saving a record a message is displayed.

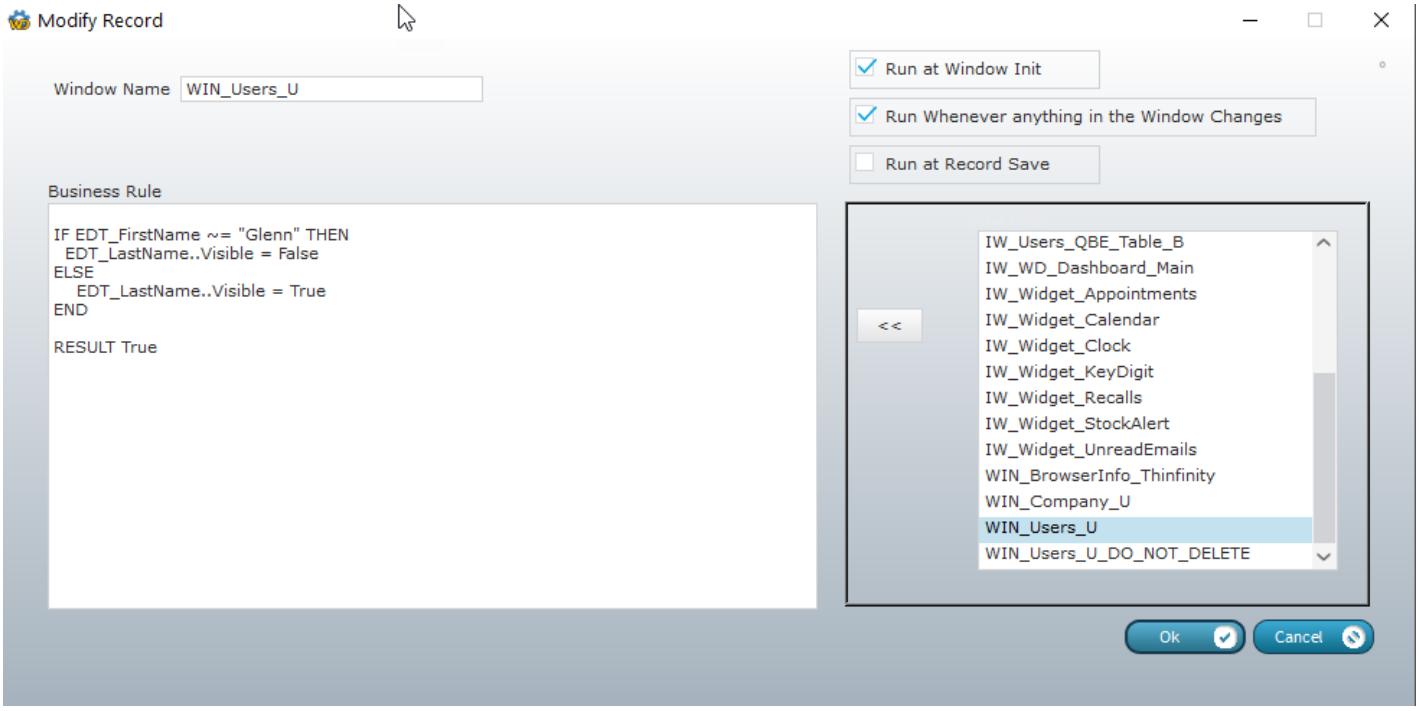
Field level business rules are linked to a field in a table in the Analysis. Any control on any maintenance form that has a filelink to a field with a field level business rule, that rule is automatically run. The advantage to this type of rule is that it works anywhere in the application without you having to define it again.

Here is the Field Level Business Rule maintenance form.

You can see that the business rule will make any control with a FileLink to the Users.LastName field required. That's all there is to making a field required.

Window Level Business Rules

As opposed to having a business rule tied to a specific field, this type rule is specific to one window. It is not tied to any control in the window. It is up to you to write the business rule using WL language.



While this is a silly business rule, it does drive home the point that you can code anything. This rule is only for the WIN_Users_U window and will run when the window is initialized and whenever anything in the window changes. You can see that if the control EDT_FirstName approximately equals Glenn the EDT_LastName control is made invisible. If it is not approximately equal to Glenn then the control is visible.

Window level business rules can be extremely powerful. You can write rules to make calculations based on other controls. You can uppercase, lowercase or even ProperCase the contents of controls if you have a procedure to do that. Again, whatever you can write in WL Language you can put in a business rule.

BE CAREFUL TO WHOM YOU GIVE ACCESS TO CREATING BUSINESS RULES. ANYTHING YOU WRITE IN BUSINESS RULES WILL BE EXECUTED.

THE CRUD CONTROL TEMPLATE

The heart of the Framework is the CRUD Control Template.

It appears on every browse window and maintenance window. It allows us to place a copy of the template on each window. The code associated with the template is inherited and can be overridden or extended pretty much similar to oop programming. I like the use the term visual oop for the CRUD control template.

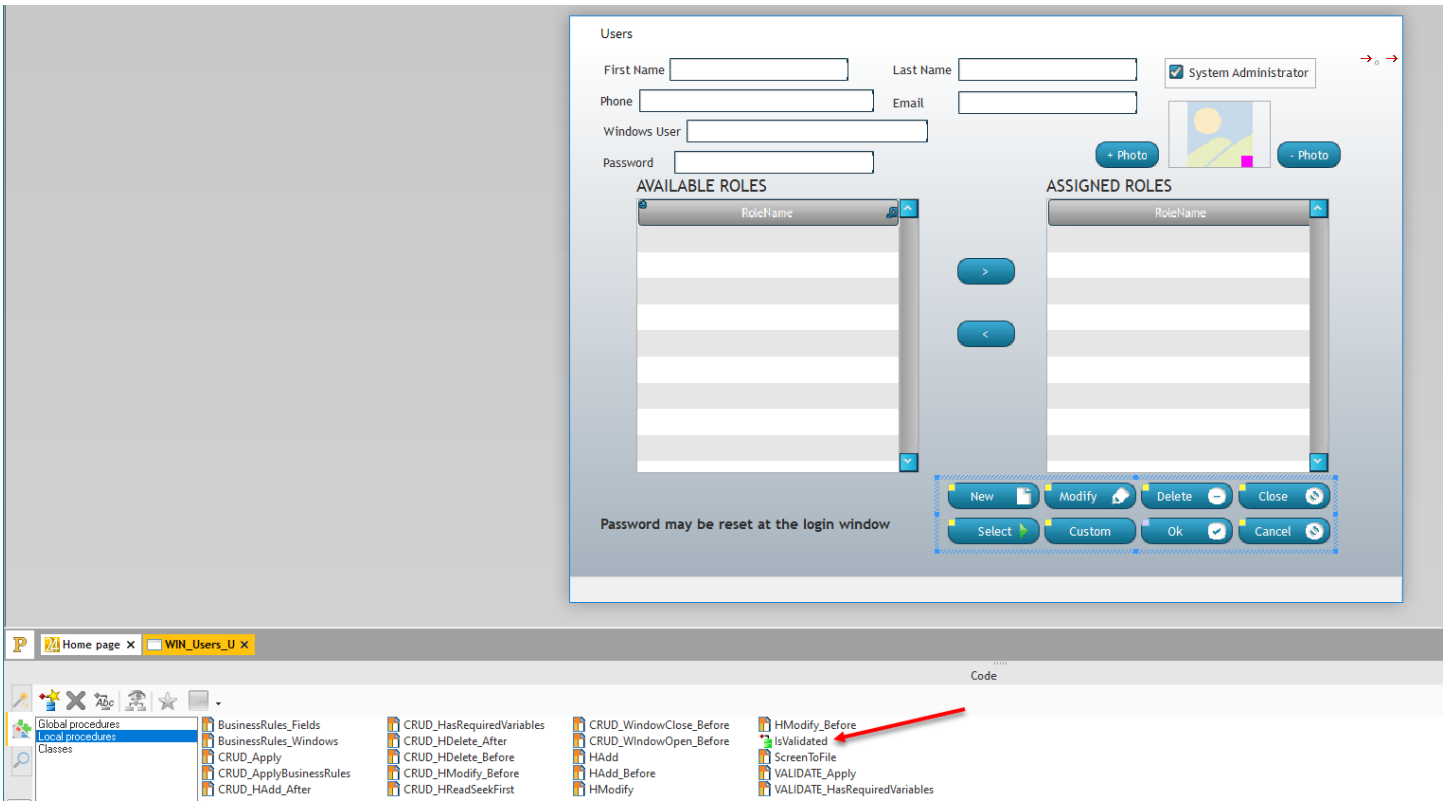
There is nothing magical about this set of controls. They are just WD controls grouped together to performs some action. We use buttons for 'Add', 'Modify', and 'Delete'. We also use the same template for the 'Ok' and 'Cancel' buttons. The template of course through code is smart enough to know which buttons to display on what forms.

Assigning values as we did earlier to the variables associated with the control template enables all the CRUD functionality through indirection. Every place you see one of the variables in your mind substitute that value for the variable. That's it.

Sample of Indirection used in the Business Rules called from the CRUD Template:

```
IF {sControlName,indControl}..FileLink > "" THEN
 sLink = {sControlName,indControl}..FileLink
 QRY_BusinessRules_Browse.Param_FieldName = {sControlName,indControl}..FileLink
 QRY_BusinessRules_Browse.Param_IsActive = True
 HExecuteQuery(QRY_BusinessRules_Browse, hNoBind + hQueryDefault)
 FOR EACH QRY_BusinessRules_Browse
  SWITCH sDisplayOrValidate
   CASE ~= csDisplay // Required have a blue halo and a red border
    IF QRY_BusinessRules_Browse.Required THEN
     IF {sControlName, indControl}..Visible THEN
      IF {sControlName, indControl} ~= "" THEN
       {sControlName, indControl}..VisualEffect = veDdw + veBlueShade
       {sControlName, indControl}..Border        = RequiredBorder
```

We mentioned the various features of the CRUD Control Template can be overridden or extended. The window WIN_Users_U is a perfect example of how to achieve this.



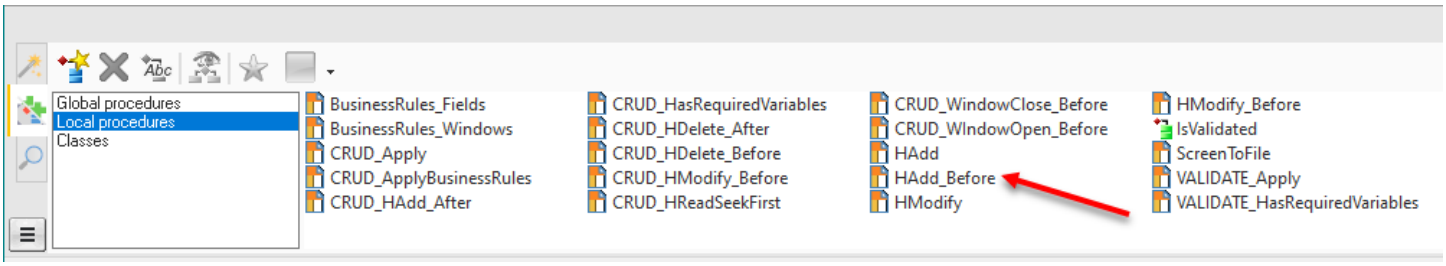
In this example we wanted to prompt the user if the record should be saved if there are not any roles assigned. First, let us look at the code. Select the control template and then select Local Procedures. You will see a procedure named 'IsValidated' has a green icon in front of the procedure name. (at the present time version 25 is not displaying the green icon.) This means the procedure has been over overridden or extended. Open the procedure and you will find:

```
Local procedure IsValidated ( WD_VALIDATE ) (TPLC_FRAMEWORK_CRUD template)
PROCEDURE IsValidated(bRunAdditionalValidation is boolean <useful> = True)

// Example of how to code the validation routine
// Special Code can be made to run when overriding the IsValidated
// on a special occasion

bIsValidated    is boolean

ScreenToFile()

// User the framework BusinessRules but you are free to use your own.

bIsValidated = BusinessRules_Windows(csRecordSave, MyWindow..Name)
bIsValidated = BusinessRules_Fields(csRecordSave, MyWindow..Name)

RESULT bIsValidated
```

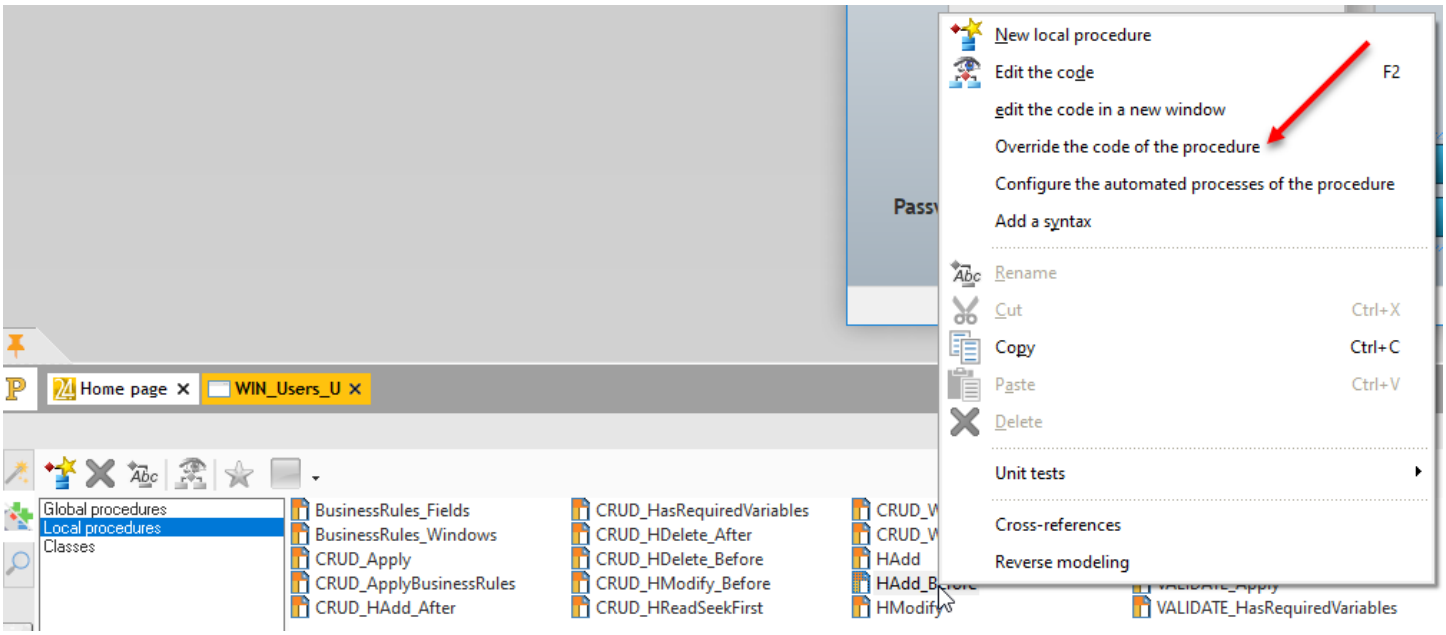
```
Local procedure IsValidated ( WD_VALIDATE ) *                    If Error: by
PROCEDURE IsValidated(bRunAdditionalValidation is boolean <useful> = True)

IF bRunAdditionalValidation = True THEN
    IF TABLE_TargetRoles..Occurrence = 0 THEN
        IF YesNo(No, "Ok to save the Employee without any assigned Roles?") = Yes THEN
            RESULT ExecuteAncestor
        ELSE
            RESULT False
        END
    ELSE
        RESULT True
    END
ELSE
    RESULT True
END
```

The code we are interested in is the area not grayed. This is standard WL Language, again nothing fancy or even hard. The RESULT ExecuteAncestor command will execute the code in the grayed out area. Because we are continuing to use the ExecuteAncestor and run the code, we have extended the functionality.

Let us look at another local procedure of the control template, the HAdd_Before.



Notice we have not overridden or extended the procedure. Select the procedure and with a right mouse click select 'Override the code of the procedure'.



Now open that procedure to see the code:

```
Local procedure HAdd_Before ( WD_VALIDATE ) (TPLC_FRAMEWORK_CRUD template)
PROCEDURE HAdd_Before()

bRes is boolean = True

sCompanyCombo   is string
sCompanyId is string

sCompanyCombo   = VALIDATE_PreviousBrowseWindow + ".COMBO_Saas"
sCompanyId      = VALIDATE_DbTable + ".CompanyId"

sCompanyCombo = "WIN_Main_Menu.IWC_Parent.[%sCompanyCombo%]"

IF CRUD_DbTable NOT ~= "Company" THEN
    IF ControlExist(sCompanyCombo) THEN
        IF {sCompanyCombo, indControl}..Visible THEN
            IF Position (HListItem(VALIDATE_DbTable, hLstDetail), "CompanyId", 1, IgnoreCase + WholeWord) THEN
                {sCompanyId, indItem} = {sCompanyCombo, indControl}
            END
        END
    END
END

RESULT bRes

Local procedure HAdd_Before ( WD_VALIDATE ) *                    If Error: by program When Exceptio
PROCEDURE HAdd_Before()
//Runs the process defined in the template
RESULT ExecuteAncestor
```

As previously mentioned, the ExecuteAncestor will run the code in the grayed out area. If you want to completely stop the code in the grayed area from running, comment out the RESULT ExecuteAncestor and replace it with your own. Commenting out the ExecuteAncestor line has not overridden the procedure.

We consider all the local procedures in the control template to be embed points, meaning that you can pretty much write your own language code to extend or override almost any code of the CRUD Control Template.

SUMMARY

The Framework for WinDev was developed to give your application development a jump-start by allowing you to create browse and maintenance windows with little to no coding for the CRUD process. Also included in that functionality is to give you a means to make this as a SaaS type of application by separating out the data according to a CompanyID or whatever you chose. This was implemented with the Saas dropfown combo. (The caption of the Saas dropdown combo is Saas. You are free to change that on each window or use the global variable `sCOMBO_Saas_Caption`)

The Framework also provides two different types of QBE. One, where you can completely design your own by adding controls to filter on as well as your own code. Or two, where all different ways you can filter the data are based on parameters of the query and presented in an easy to use table format. If you have a query parameter beginning with ParamND_ it will not be displayed in the QBE. The ND is short for 'No Display'. This brings up a naming convention of the query parameters. We typically name our parameters as such: Param_First_Name or Param_ND_First_Name. This enables the QBE table to remove the Param label at the beginning and also replace the '_' with a ' '. So Param_First_Name is displayed as 'First Name' in the QBE table.

We've also implemented Business Rules that you can design, code and call from several places of the maintenance window.

PCSoft has provided a fantastic family of products collectively known as WX. Since the products share a core language, you can most likely create your own framework, or take some of the functionality or code of this Framework and implement in WB or WM. By far the most powerful features of the WL Language is Indirection and Dynamic Compilation. If you would like to learn more about Indirection follow through the code of the CRUD Control Template. If you want to know more about Dynamic Compilation follow the code through the implementation of the Business Rules as it is executed within the CRUD Control Template.

If you have an idea for new functionality in the Framework, let us know. If the idea has broad apeal we may implement it.

Remember, the Framework is pure code. You are free to add to, change, or delete the code according to your needs.

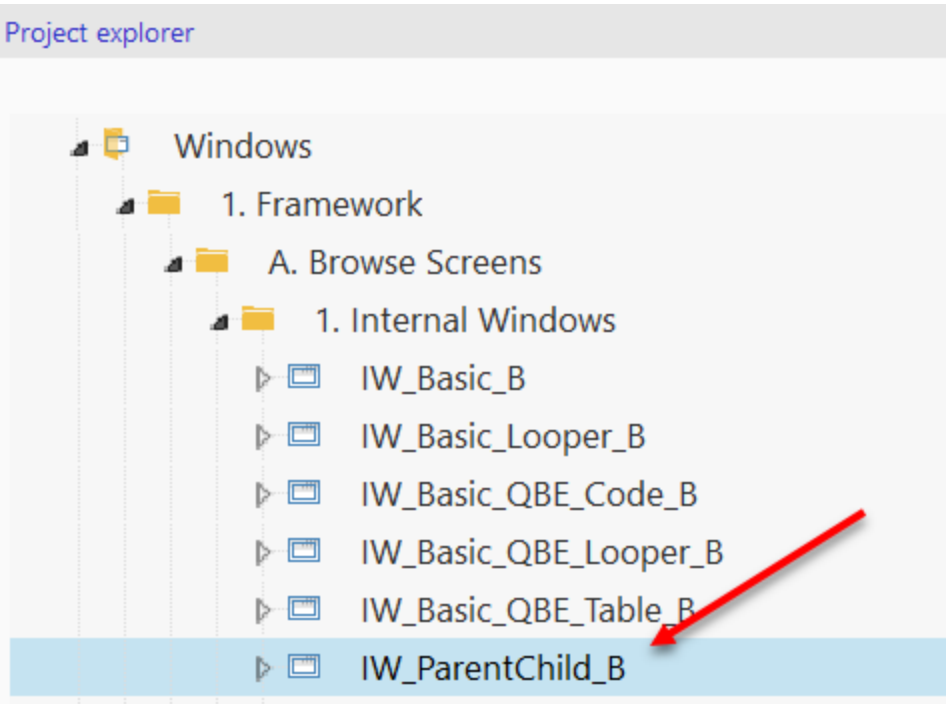
Advanced Features

Parent / Child Relationships

The Parent / Child relationship is accomplished with two table controls on a window. The synchronizing code is already done. While there is not any code to write, you do have to assign values to the CRUD Control Template as before. However, this time you have to assign variables to the Child CRUD Control Template.

Below is a working model of the Company / User relationship:

To create your own window, select the IW_ParentChild_B



As with creating any window, do a 'File Save As'.

Instead of connecting a table control and columns to a query, you will do this for the parent and child table controls. You will also need to assign values to the CRUD Control Templates for the Parent and Child.

Use the Examples as a resource to see how any of the windows areimplemented.

APPENDIX

The CRUD Control Template

IF you want to implement the Framework in a different database than HF, you can easily do so. Create a new connection in the Analysis and use that for all the tables. If you want to keep the core tables of the FRAMEWORK in HF and use a different database for the other tables, simply verify the connection for each of the tables and adjust as necessary. The Framework uses two different connections so you can separate out the tables if desired.

How to save an image or for that matter any kind of blob in a record

The WIN_Users_U has an image control to dislay a photo. The code for retrieving the image and saving in the table is under the BTN_GetPhoto. The code to save the image is easier than you think.