XbWEB

Development Tool

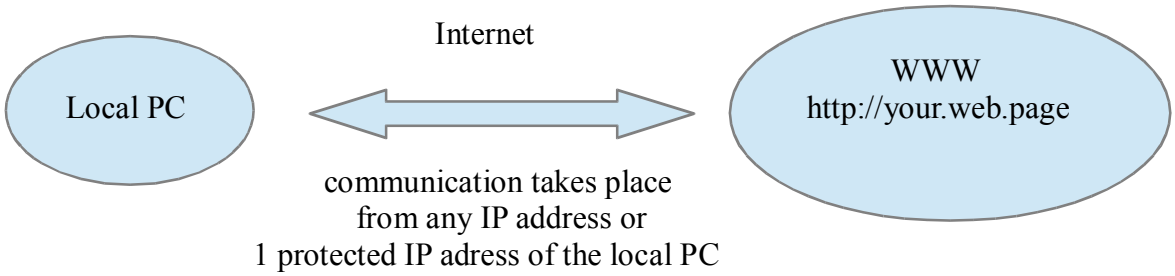
Author: Dragan Pavlović

Content

XbWEB is a development tool that uses xBase code.
Create a simple application whose source code and database are on the WEB.
The program integrates a powerful Alaska xBase++ code interpreter with additional enhancements to help you easily make your application.

You can also use XbWEB. LIB to integrate through the Alaska xBase + + 1.90 compiler.

How does the XBWEB application works:



Local PC:
- Execution of source code
- Data control over the object
  MySql or file-array
- Show menu and buttons

It's on WEB:
- Source code
- Your data in MySQL
  database or file (.arr)
- Menu, button bar file ..

- The communication protocol is password protected by 16 characters that you set
- Server logon enabled through user levels with full access or Read only. Login is possible from any IP address (any PC) o r predefined IP addresses
- Access menus organized from user level position
- MultiLanguage interface Supported (simple Add new languages enabled)
- Executing source code on local PC. The contents of the source code are located on the WEB server. Source code is performed through an embedded interpret that supports the language of Alaska xBase + + V 1.90, and additional features available through XbWEB tools.
- The data control located on the WEB server is supported by using the T_Alias () object that gives full control of the data through a working area whose source is:
  - Table from MySQL database from WEB server
  - The table resides in a file with an extension .ARR on a WEB server
  - Table located in file (.ARR) on local computer
  - A table from an Access database located on a local computer
  (extension: .MDB or .ACCDB)
  - Table located in file (.DBF) on the local computer
- Data source control. Add/delete a data source, determine the structure of the tables.
- Tabular display/edit data from any object T_Alias ()
- Generate a report that has data in any T_Alias() object
- Generate a report without a work area (line view). Display text, primitive graphics, bar codes, charts.
- Easily generate data entry dialog
- A lot of extra useful functions

Prerequisite for using XbWEB application:

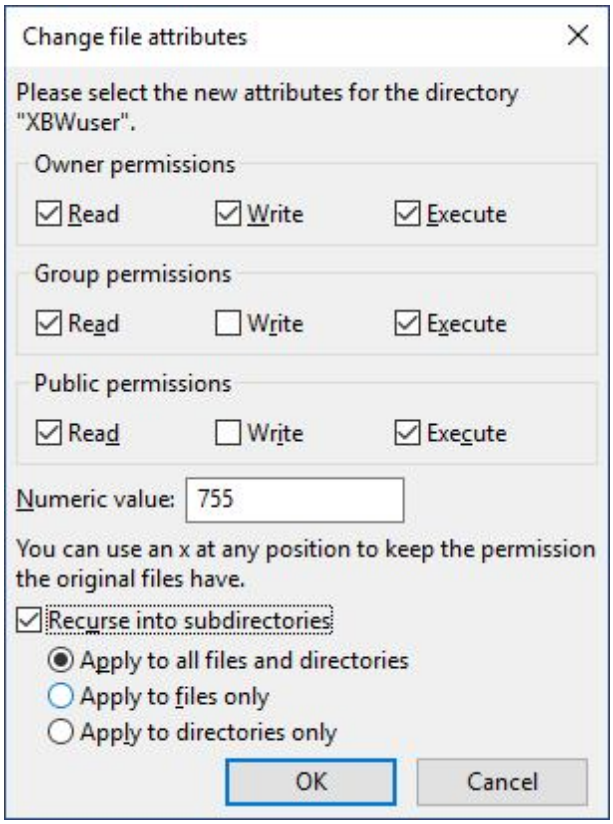
1. You must have the option of using MySQL database on your WEB portal
2. Set up on your WEB host all PHP scripts and additional files obtained with XbWEB
3. You need a key (encrypted file) that you create after you register and purchase a certain product from a distributor of the XbWEB program
4. On the development PC , install the Setup_XBWEB.exe application
5. Install Setup_XBWEB_client. exe on the local PC of your users

How to set an application on a WEB server

On your server, using some of FTP tools setup next parameters:

Specify the folder where your application will be located. Letid by for example, XbDemo

Folder XbDemo Specify attributes:



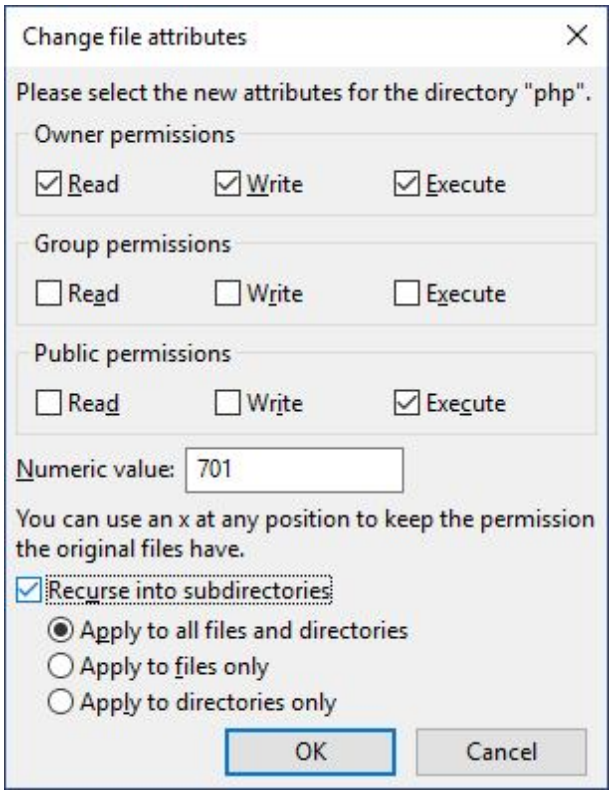
Only the main user has the right to write Data

Under folders XbDemo create subfolders:

APL
SYS
DLL
php

Note: The php folder is in lowercase letters and the others are capital as shown above.

NOTE: You must have access to the MySQL database (at least one available database) on your host
Assign the following attributes to a php folder:

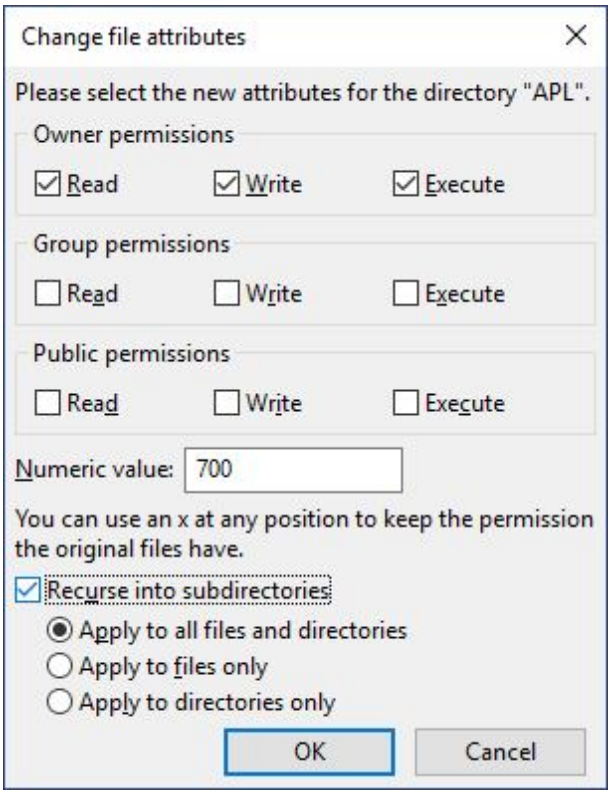


Only the main user can read, write and perform

Everyone else is forbidden

Only "Public permisions" is checked for "Execute"

Set the attributes for the APL, SYS, and DLL folders as on the following illustration:



Only the main user can read, write and execute

Everyone else is forbidden

The **Setup_XBWEB. exe** installation program is located in the folder **\XbWEB** (standard name of the folder). Below this folder are folders: APL, SYS, DLL, php, Manual

An application from a client PC from an IP address to access your WEB server using the routines stored in the XbWEB. DLL on one side and on the other through PHP scripts mounted on your WEB server.

You need to create a folder (for example, root: XBW_App) on your WEB server below root.
Then, below this folder, create the folders: APL, SYS, DLL, php
Specifically:

```
/XBW_App/APL
         /SYS
         /DLL
         /php
```

Note: Php folder must be entered in lowercase letters, and other folders are capital.
In PHP folder, copy any php files that came with the installation of the xbweb application.
It is not important to reedit the contents of the **xset.php** file on the local disk.
It looks like this:

```
<? php
Define ("SECURE_KEY", "KEY16_char");
Define ("_ HOST", "localhost");
Define ("_user _", "username");
Define ("_pass _", "password");
Define ("_TIME_OUT _",600);
Define ("_mdb _", "xbweb_database");
Define ("_PATH_BIN _", '/mnt/ext/opt/mysql/bin/');
Define ("_DB_NAME _", ' xbweb_database ');
Define ("_DB_PBY _", ' ');
Define ("_DEF_NAME _", ' admin ');
Define ("_DEF_PASS _", ' admin ');
>
```

Line:	Replace:	New content
define ("SECURE_KEY", "KEY16")	key16	Enter 16 characters representing the communication protocol code. This password must by the password defined in the KEY file of your application. Example: 0123456789#XbWEB
define ("_USER _", "username")	username	User name for the CPANEL
define ("_PASS_", "password")	passowrd	Password to access the CPANEL
define ("_TIME_OUT _",600)	600	Time in seconds after which you have to log back if you were inactive
define ("_MDB _", "database")	database	MySQL database name
define ("_DB_NAME _", ' database ')	database	MySQL database name
define ("_DEF_NAME_",'admin')	admin	Login supervisor standard name to your application
define ("_DEF_PASS_",'admin')	admin	Standard password for sign-in to your application

Your extension files are stored in folder /XBW_App/APL :

.CMD - Source at your application
.MNU - Menus
.FRP - files used by report generator

In folder /XBW_App/SYS should be:

MENU. INI - File that contains a list of the languages you use in the application
MENU_xx. INI - For each language individually (tag xx) .INI file contains messages

Files with extension .ARR representing table-array (table placed in file)

In folder /XBW_App/DLL may stand:

XBWSETUP.DLL - Program for the xbweb Application Management Program by the user - supervisor
The XBEDIT.DLL - Editor with CMD, FRP, MNU
.DLL - Other DLL files that you can develop in Alaska Xbase + + developing environment and use in your application

Upgrading programs.

If you want your client computer to automatically update the software, you need to do the following:
In the Root folder on your WEB server there should be files:

UPDATE.INI -
XBUPDF.EXE - With the help program used for the upgrade process
XBWEB.DLL - XbWeb communication Library
XBWEB.EXE - Executable XbWEB file

If you do not use the standard executable XBWEB.EXE file, but it is your file, then replace it with XBWEB.EXE. In addition to the listed files can be other DLL and EXE files that you have developed.

When you sign in to a WEB server, the program checks for the file UPDATE.INI in Root folder. If present , it is downloaded immediately into the Root folder of the local PC after downloading the files specified in it.
Standard content UPDATE.INI file is:
/XBWEB.DLL,3955200,10.12.2018,18:19
/XBWEB.EXE,288768,10.12.2018,18:19

The files are listed in the lines individually: the filename with the indicating folder or Root (/) Followed by the comma and length of the file, comma and date, and at the end of the comma and time. If there is a change of these files and UPDATE.INI (on the Root folder on the WEB) content is different from the content on the local PC to an automatic update processes and downloads of these files. After the end of the process, the program will run again.

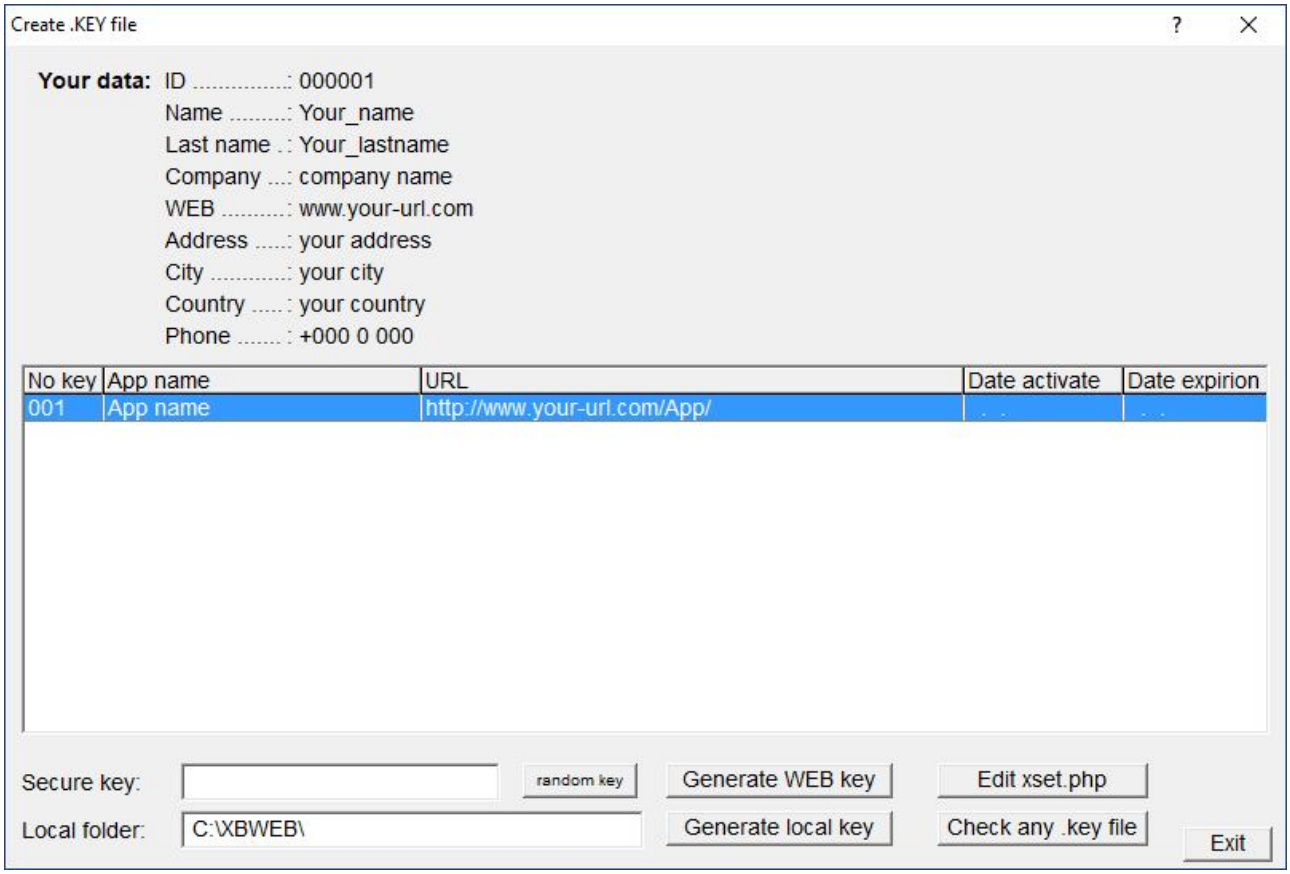
Note: The file names used in the UPDATE.INI file must not contain the name of the symbol "-" (bottom line for highlighting)

Creation of .KEY file

Assuming that you registered as a user on the website www.xbweb.rs and you have purchased at least one product (the bench) follow the instructions.

The XbWEB folder contains the file **XBW_KeyGen.exe** Run it.

You need to log on to the XbWEB portal with your email and password provided during registration.



In the offered window, choose one of the purchased keys. At least one must exist.
Then enter the password (16 characters) in the secure key box or click the random key button. Allow the program to generate key by itself. Then click on "Generate WEB key" to form the file with extension of a .Key and save it on a local disk. You can also use the local “Generate local key”. You have also have to specify a local folder to work on the local disk.

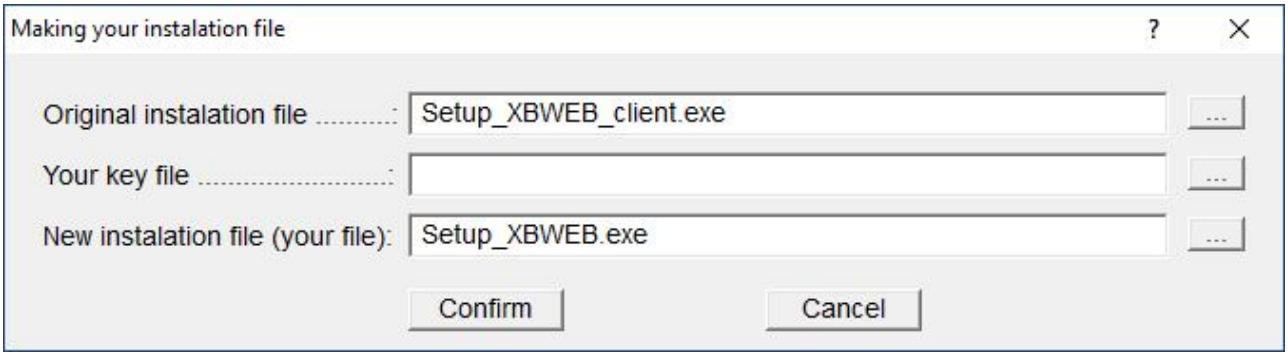
The “Edit xset.php" button serves you to edit this file (it was mentioned above), and the key in the secure key field is automatically placed in the corresponding xset.php file line.
To check any .key, file button is used to verify some previously generated key file (which you have generated) and the display its content on the screen.

The .Key file is copied to ..\SYS folder of each user who uses this application.

Note:
Creating a key file will also have an activated key. In fact, you can activate the key purchased later and after purchasing. Since the key is triggered, the date of his duration is determined in accordance with the selection during purchase. You can always create a key again and change the communication password (Secure key) that opportunity. You cannot modify the URL link. It was appointed in purchasing a key.

Installation file for clients

The client is in the Create_Setup.eke program
Run it and get the following window:



The original file Setup_XBWEB_client.exe is already in the same folder, so skip the first field.
In the second line, enter the key that you have created (.key file) or find it by clicking the "..."
Set the new file name Setup_XBWEB.exe (*) to your liking.
Clicking the "Confirm" button will create a file (*)
You can place this file on your WEB site and give it to your users. Once installed, they will automatically receive the .key file that points to your WEB link.
If you were to use the original Setup_XBWEB_client.exe file, the user would have to get you the .key file that should be copied to the \ XBWEB \ SYS folder which is certainly a more complicated procedure.

Local host mode

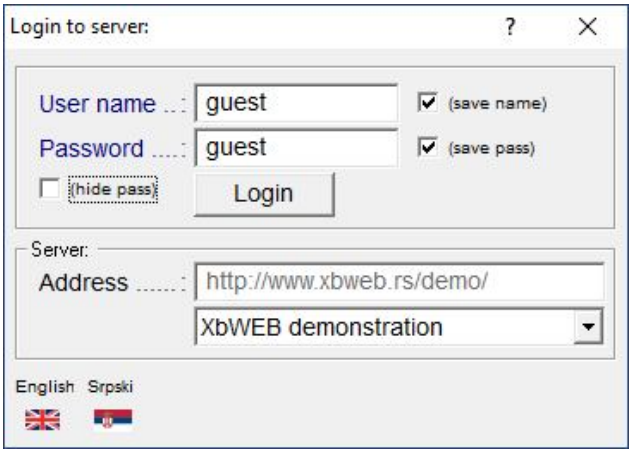
The XbWEB can also work on the local PC (in local network) without the Internet.
If the application URL is listed as a local guideline (for example: \xbweb\) the program will work the same way that Access's database is (with. mdb or. accdb).

During forming a key (.key file) you should also form a local key next to the WEB key.
By using the local mode, you can develop the application faster and then copy the required files from the APL and SYS folder to the WEB folders.

The simplest way to run program from the client PC:

After you install Setup_XbWEB_client.exe on the destination folder (standard C:\XbWEB) it is necessary to copy the keys (.KEY files)
To connect programm to a server, there must be at least one key.

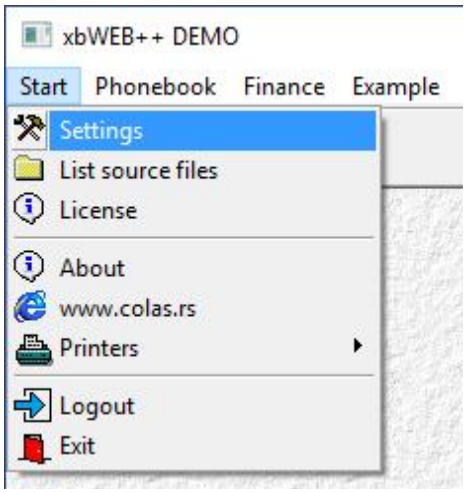
Run program C: \xbweb\XbWEB.EXE



Log on to the selected address.

in the .KEY file is the name of the application, the server address, and the communication key.
When you log on to the server, as user you are assigned the level of privilege, menus, and button of the program's basic menu buttons:

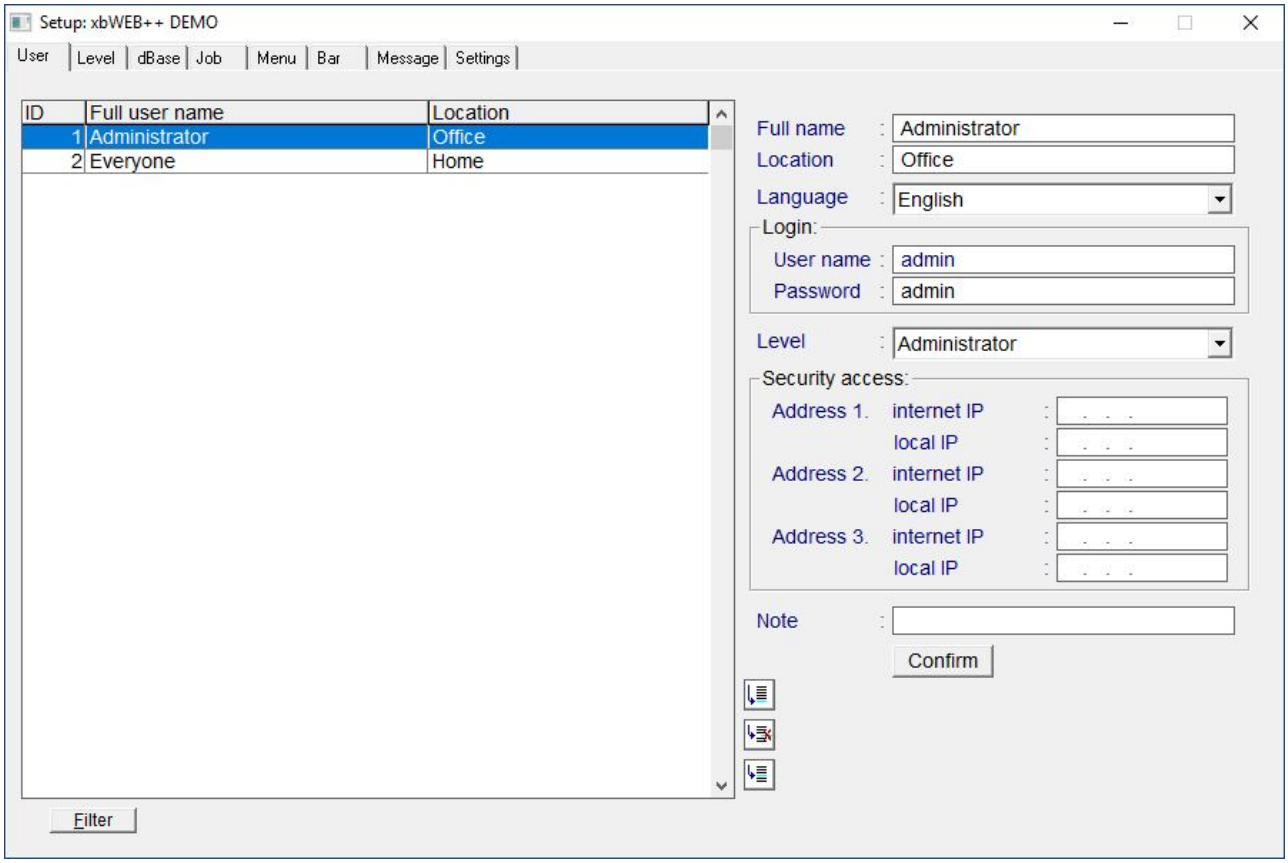
If you are logged on as a supervisor, in the basic Start menu you have two additional options:



Supervisor receives additional options:

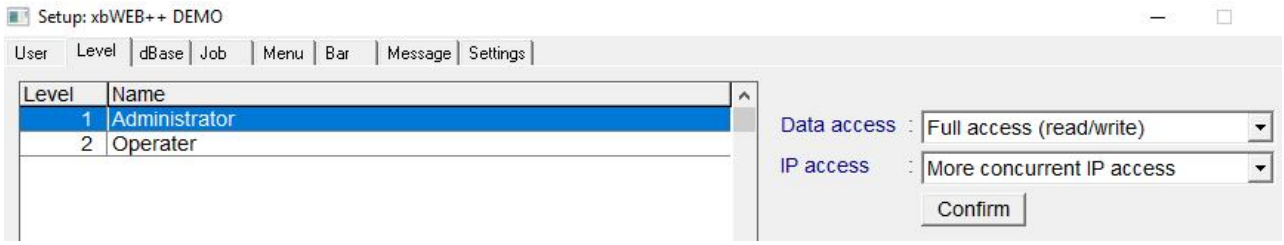
Select Settings option among others
View the list of source files

Supervisor can, using the option "Settings," perform various software settings and program specific options.



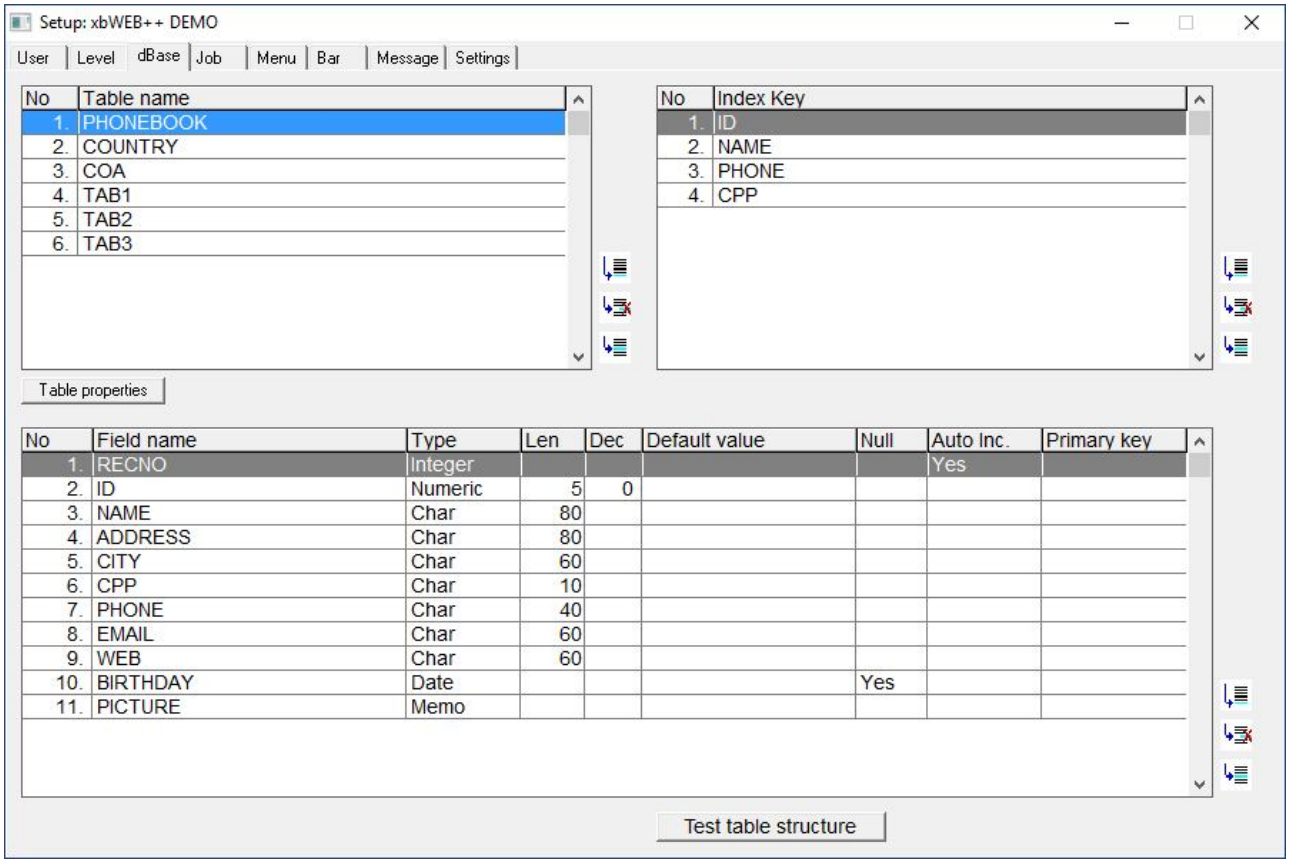
The first “User” language enables the user to determine the program, its name, language, name, and password, which is logged on to the WEB server, access level.
If you want to limit user access only from a specific IP address (or to three IP addresses)
You can do this by entering IP address 1.2 and 3. The first IP is an Internet address and the other is the IP obtained from the local network that the Internet is accessing.

Tab “Level”



In this section, the levels are set as the data type of the file: Full access (Full access) or read-only. In addition to this, a specific level user may have access to one or more IP addresses. Specifically, multiple users can log on with multiple IP addresses at the same time with a user name and password.

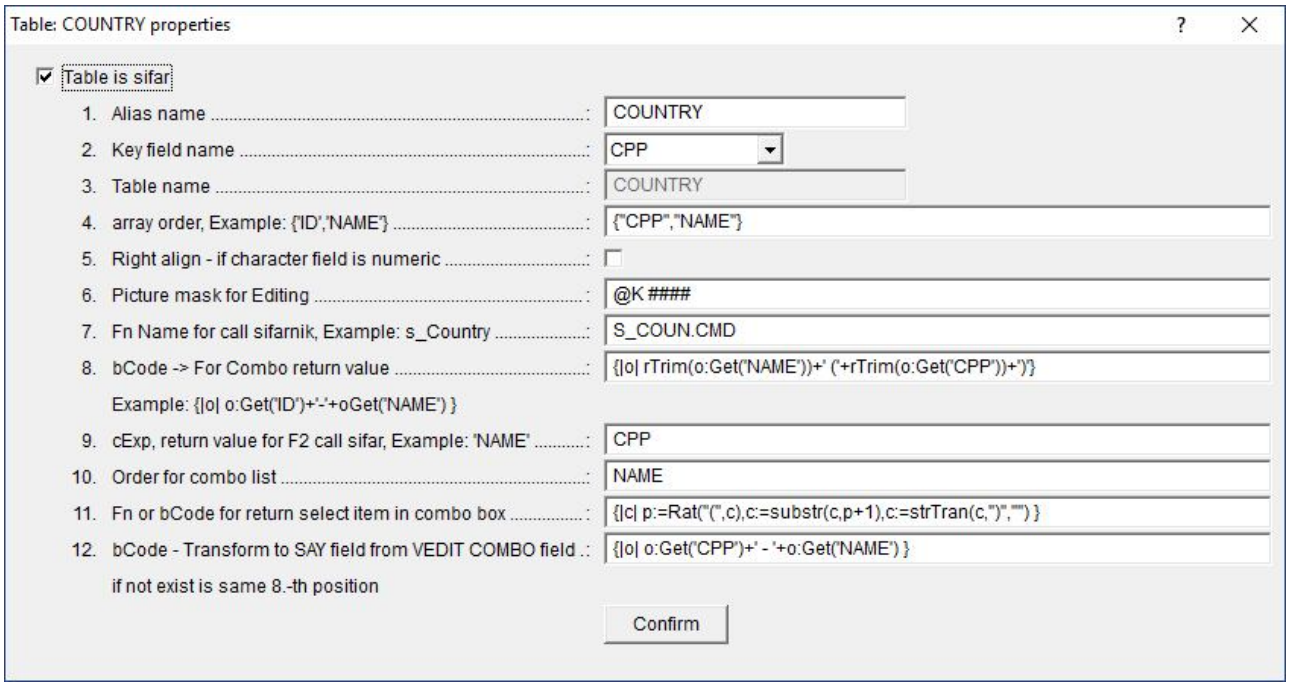
Tab "dBase"



This option defines the structure of the tables in the database and the index keys.

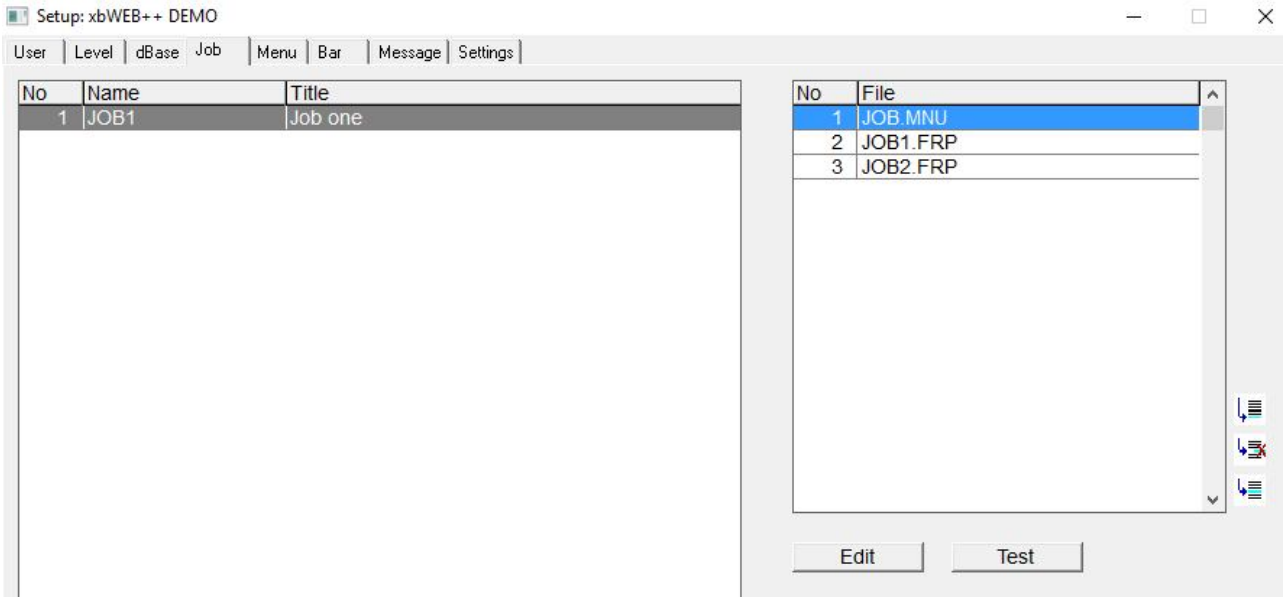
The "Test table Structure" button serves to comply with the outline between this table and the actual database . This way, if a new field or table is added, the same changes will really be reflected in the database.

The Table Properties button is for further setup of the table set to be a coder.
Let's look at the COUNTRY table



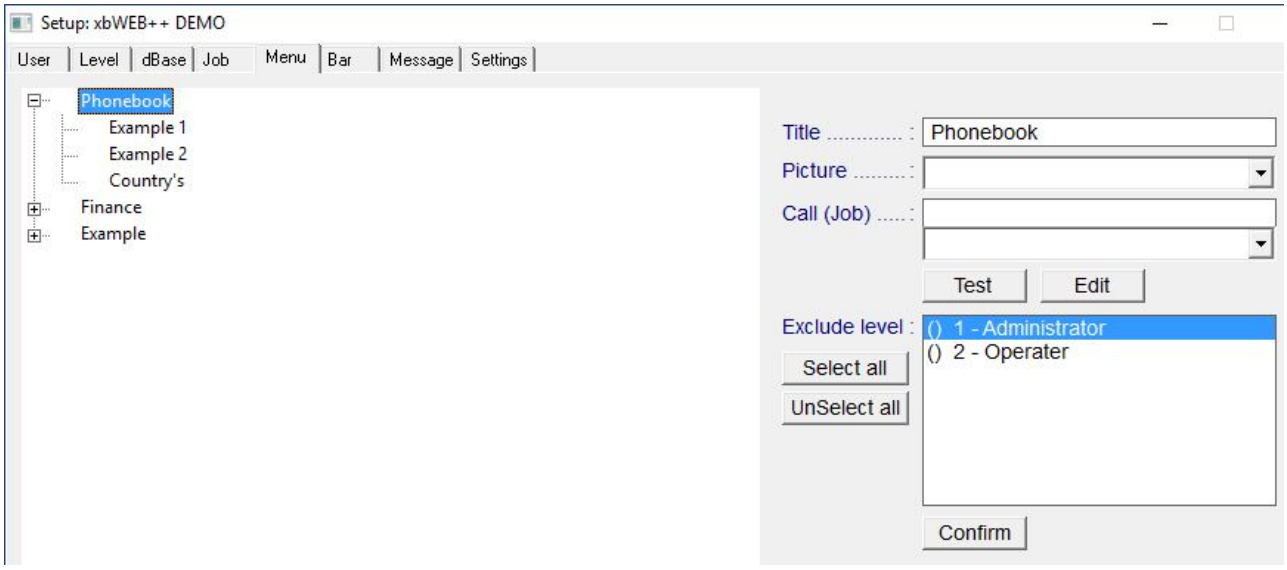
The COUNTRY table is named Codor. A key field in the table is a CPP country label.
After these settings when we want the field from another table and the variables in the dialog box, which are the label of the country, we connect with the COUNTRY table enough to give him the LINK to COUNTRY table.

Tab "Job"



In this window define specific jobs (file group)
For each job, it is possible to assign a file fish with an MNU extension, FRP, CMD.
This group makes it easy to access the specific job files that are used.
When you select a job, the first specified file is always called within its group
(in the example above, it is job .MNU)

Language "Menu"



This window defines menus.

Field:	Description
Title	Contains the menu name
Picture	You can choose one of the thumbnails that is displayed in the menu in front of the name
Call (Job)	Is assigned the name of the file that is running or simply chooses already defined job Assigned to the menu at its choice. From the job chosen is always Calls the first file in the list Call file can be: CMD, MNU, FRP
Exclude level	Can turn it off by certain user menus (access levels)
Confirm the	Button you confirm settings for
Edit the	Specified Call file
Test of	That file (execution)

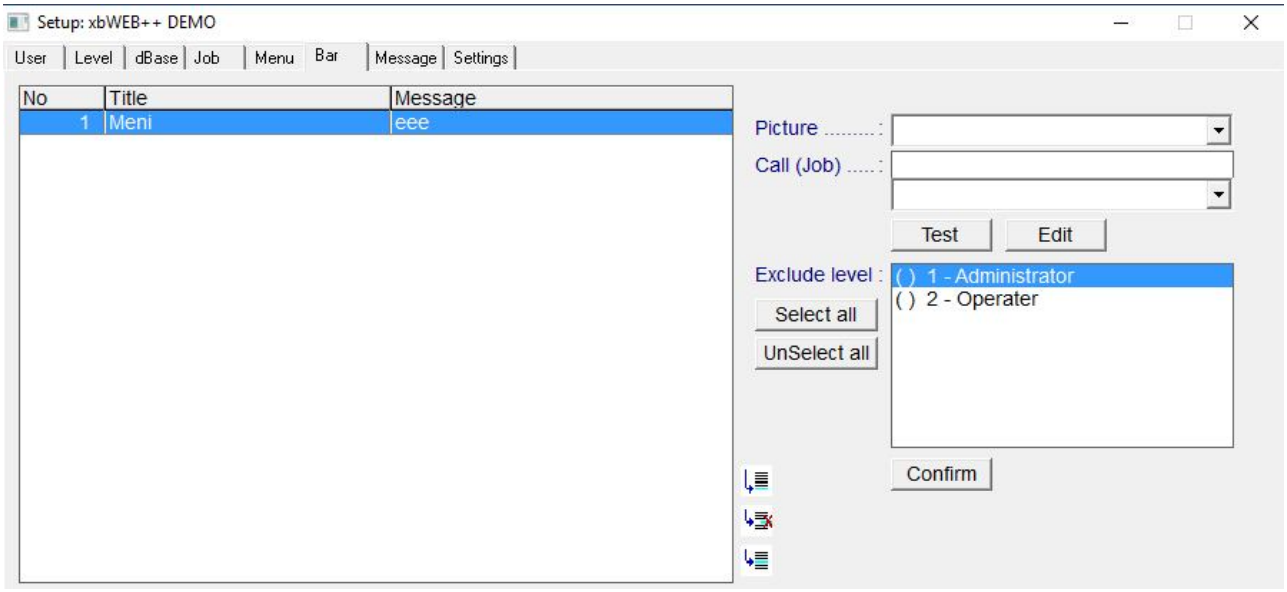
Adds a new menu (next in sequence)

Deletes the selected menu

Insert Menu

Insert Submenu

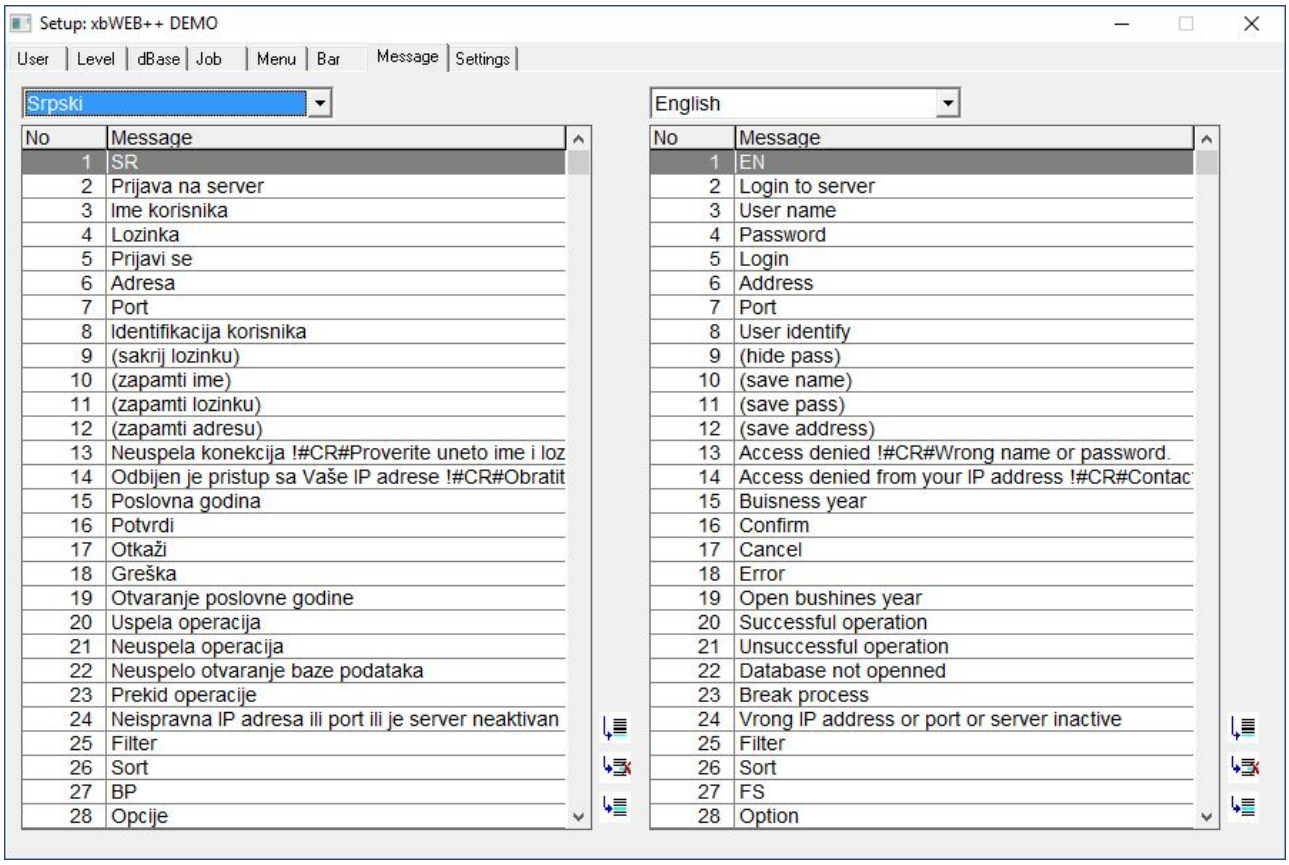
Tab "Bar"



Analog defining menus in this window defines the buttons listed below the menu in the underlying program window.

The language "Message". Add new messages that are later used through your app.

They are displayed in parallel two tables in two languages selected.



No	Message
1	SR
2	Prijava na server
3	Ime korisnika
4	Lozinka
5	Prijavi se
6	Adresa
7	Port
8	Identifikacija korisnika
9	(sakrij lozinku)
10	(zapamti ime)
11	(zapamti lozinku)
12	(zapamti adresu)
13	Neuspela konekcija !#CR#Proverite uneto ime i loz
14	Odbijen je pristup sa Vaše IP adrese !#CR#Obratit
15	Poslovna godina
16	Potvrdi
17	Otkaži
18	Greška
19	Otvaranje poslovne godine
20	Uspela operacija
21	Neuspela operacija
22	Neuspelo otvaranje baze podataka
23	Prekid operacije
24	Neispravna IP adresa ili port ili je server neaktivan
25	Filter
26	Sort
27	BP
28	Opcije

No	Message
1	EN
2	Login to server
3	User name
4	Password
5	Login
6	Address
7	Port
8	User identify
9	(hide pass)
10	(save name)
11	(save pass)
12	(save address)
13	Access denied !#CR#Wrong name or password.
14	Access denied from your IP address !#CR#Contac
15	Buisness year
16	Confirm
17	Cancel
18	Error
19	Open bushines year
20	Successful operation
21	Unsuccessful operation
22	Database not openned
23	Break process
24	Vrong IP address or port or server inactive
25	Filter
26	Sort
27	FS
28	Option

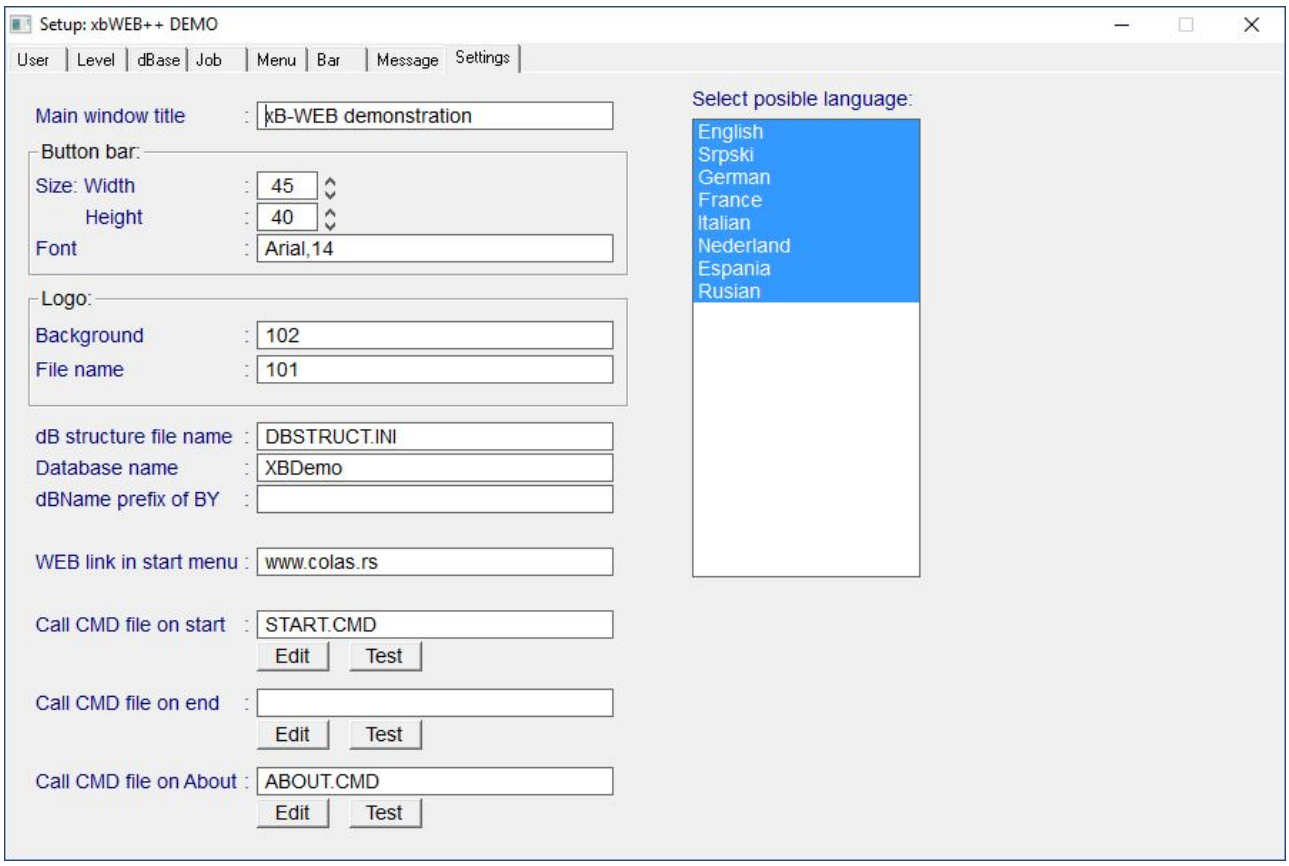
In your code you can get call functions : XBW_Msg (No)
Where the message parameter is No. This function returns a string (message)
In the language you logged on to.

An example in. CMD file:

Info (XBW_MSG (25)) -> writes the text "Filter"

Note: When using a language that has a specific code layout, it must be set to Windows Regional setup for Unicode applications to that language

Label "Settings"



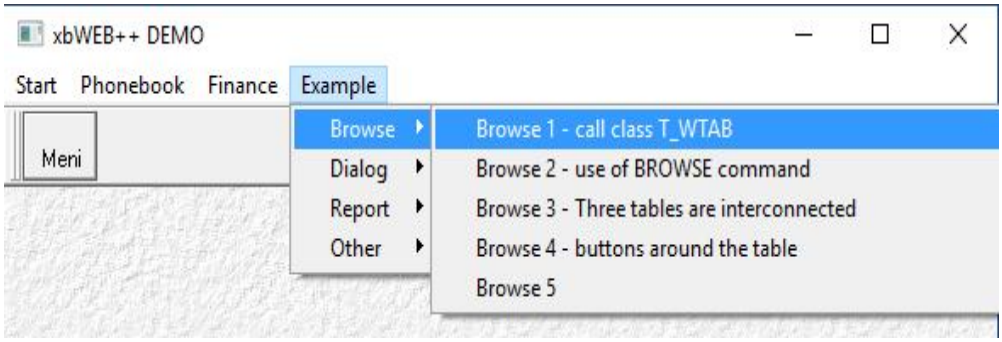
Explanation:

Main window Title	- Default window name
Button Bar	- Set the buttons: width, height, and font
Logo	- Window background and file name are the numbers of images integrated in the XbWEB Tool
DB Structure file name	- The name of the file that contains the structure of the tables
Database name	- Database name
dbname Prefilx of BY	- Prefix in database name
WEB link in Start menu	- WEB link of your site that appears on the root menu
Call CMD file on start	- call cmd file when starting a program
Call CMD file on end	- call cmd file when exiting
Call CMD file on about	- call cmd file by selecting options about on the basic Menu
Select posible language	- Selecting offered languages This option changes the LANG file directly. Ini

Editing files with CMD, FRP and MNU extension

If you are logged on as a supervisor, you have the option of setting up and editing program files.

For example, on the menu tab from the bottom image before right-click, press CTRL (means CTRL + ENTER or CTRL + Right_click)



Then, instead of calling a specific part of the program, the editor is called with the contents of the corresponding file:

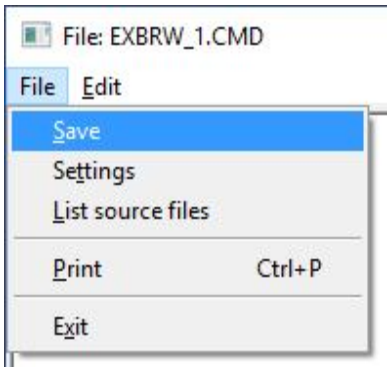
File: EXBRW_1.CMD

File Edit

```
PARAM o

// Direct call the class T_WTAB

oTab := oXbWEB:XClass("T_WTAB","40","20",800,400,;
        { oXbWEB:oWnd ,;
         'Phone Book'} ,;
        'Arial.14',;
        oXbWEB:oDB:oAlias('PHONEBOOK') ,;
        {'ID','NAME','ADDRESS','CITY','CPP','PHONE','EMAIL','WEB'}  ,;
        {'ID','Name','Address','City','CPP','Phone','eMail','WEB link'} ,;
        ,;
        {60,400,300,300,300,300,300,300} ,;
        {'R','L'} ,;
        {"IF(Empty(NAME),'#####','.')",'@S100'} ,;
        {'!Empty(ID)','!Empty(NAME)'} ,;
        {,,,'COUNTRY',,,} ,;
        ,,3,360,;
        {"ID","NAME","PHONE","CPP"} ,;
        {'BAR BOTTOM ALIGN RIGHT','VTAB',,'ADD','DEL','INS',;
                {'FONT','Arial.12'},;
                {'MENU',"&Option",;
                        {{'&Report',"PhoneBook.MNU"},;
                         {'&Option A',{|| Info("Hello !",,"Option A ..")}}},;
                  ,22},;
                'BAR BOTTOM','FS:','SORT:','FILTER'} ,;
        , 'AD E',,,,,,,2)

oTab:Show(.T.) // Dialog Wait is .T.

RETURN NIL
```

Cursor: row 1, col 1 Num Caps Ins

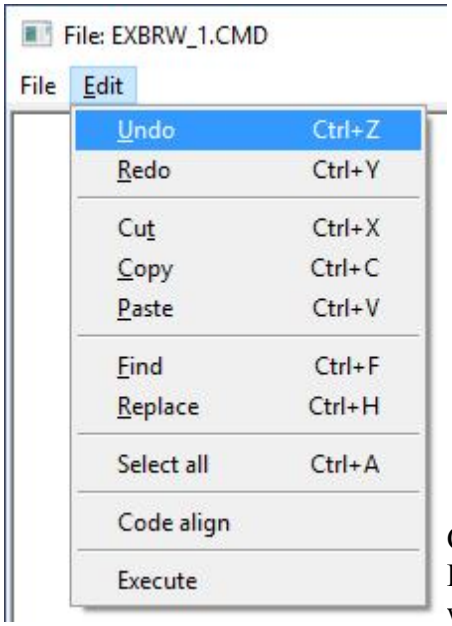
Some options:



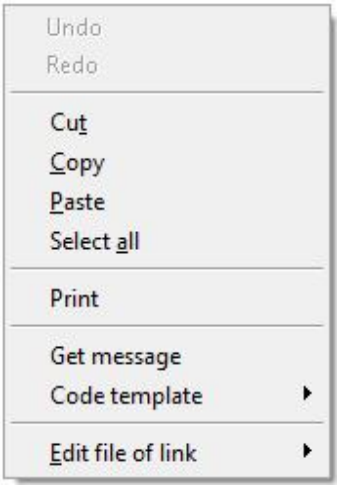
Save- Download file
Settings- Call dialog to set up the editor
List source Files- program file list

Print- Press code

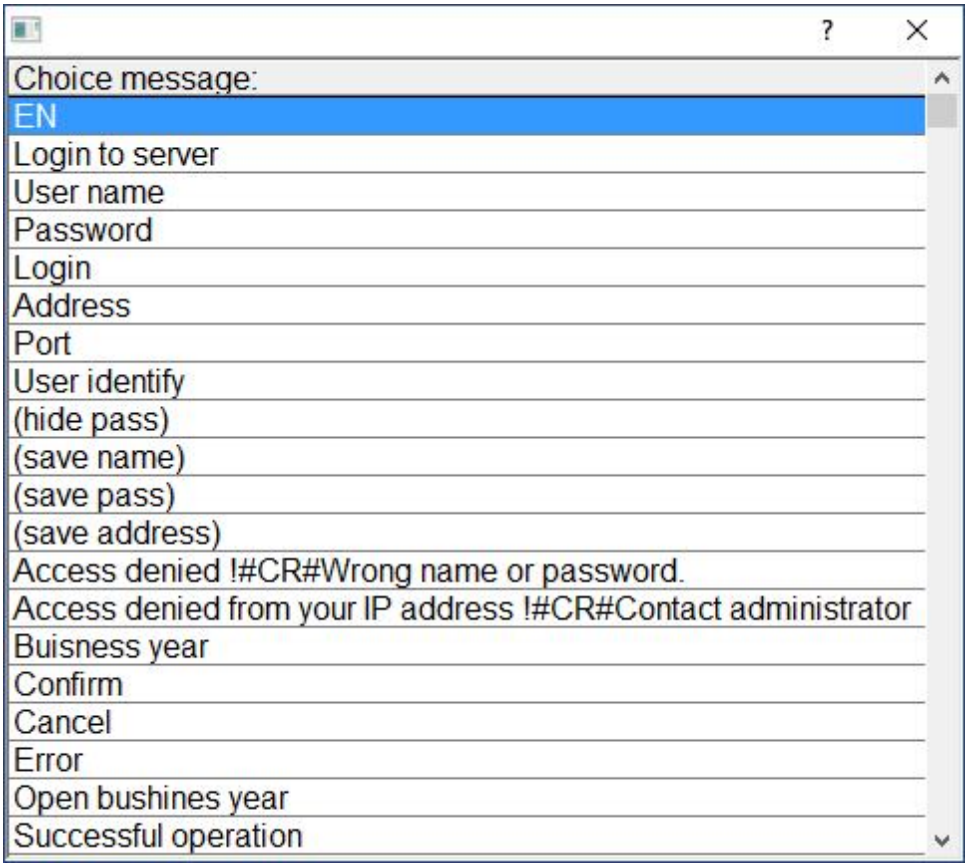
Exit- Exit Editor



Code alignment
Perform a programming Code (you can always try how it works when you enter the code)

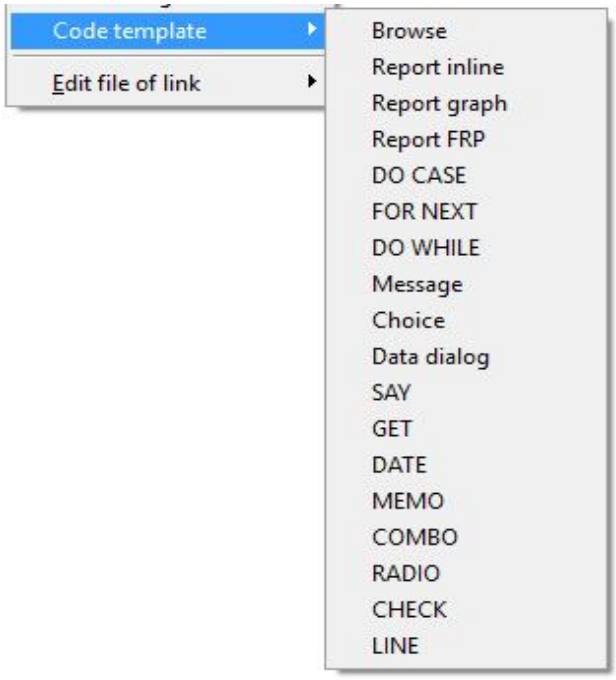


The option: "Get Message" will open the following table:



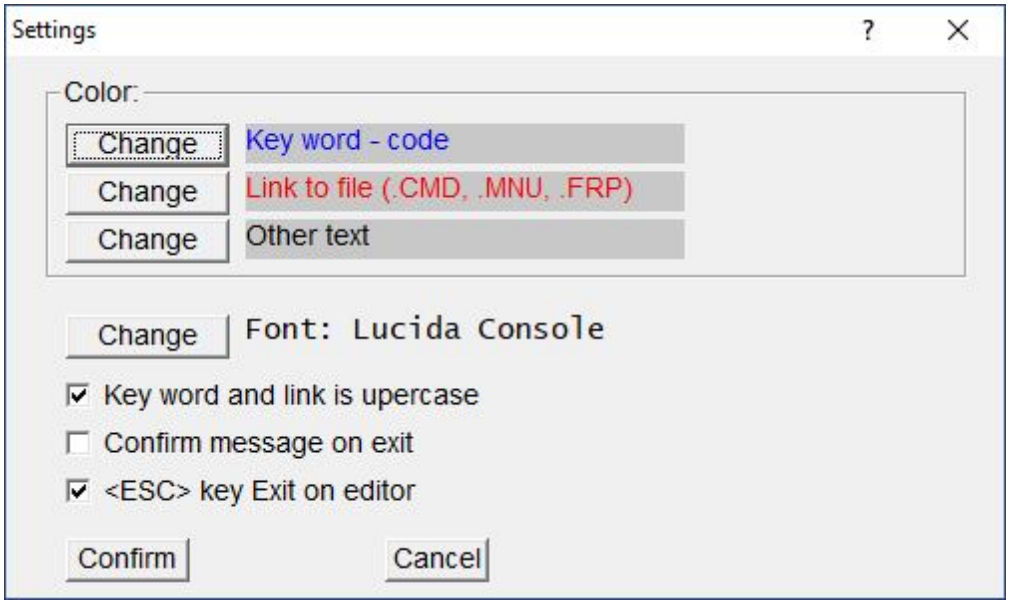
Selecting one of the offered messages on the cursor position will be returned text with the selected message number in parentheses: XBW_Msg (1)

Code Template provides a list of Code examples that can be indent in an editor

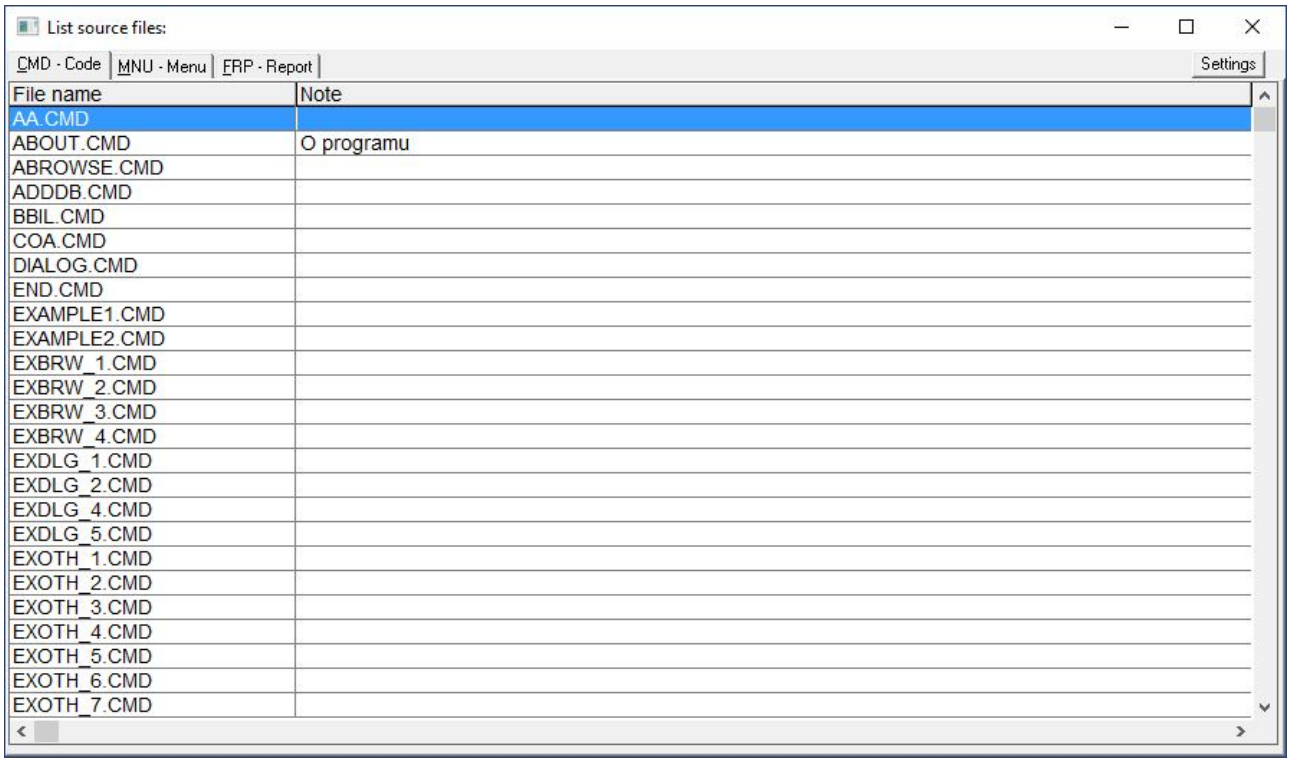


If there are specific text in the text (code) that is associated with a CMD, FRP, or MNU extension (see the first picture and it stands for "Phonebook. Mnu") then a list of those files is offered to be arranged in a separate window (editor). Example:

"**Settings**" makes the adjustment to the editor.



The "**List source files**" option provides a table with a preview of all program FILES with a CMD, mnu, and FRP extension. Each file can be attributed to a description (a comment) or called that file to the left by clicking the editor's editor.



Files with extension. CMD contains source code and special Inerlines are performed.
The call and execution of such files is done in the following way:
Call ("TEST. CMD ", p1, p2, p3)
Parameters P1, p2, p3 or more will be forwarded during the call and any other Fn

Commands that can be used in the CMD file interpret:

PARAM Cvar [, CVar2..]

In the beginning of the CMD file, input parameters can be received

Example (a CMD file call): Call ("TEST. CMD ",7,2018)
In the contents of the file TEST. The CMD start line is:

```
PARAM nMonth, nYear          In The nMonth And nYear variables will be received
                             Input parameters: 7 and 2018
```

```
DO  WHILE  1Condition        // Logical  lCondition of execution DO WHILE loop
      EXIT                   // Exit from do while loop
      LOOP                   // Rollback of code execution on line below up DO WHILE
ENDDO                        //
```

```
IF  1Condition                        // If a satisfied condition will be
   .. Code
ELSE                                  // Otherwise it will be
   .. Code
ENDIF
```

```
DO CASE                      // DO CASE quotation marks
     CASE 1Condition         // For fulfillment of the conditions of the first case will be
        .. Code
     OTHERWISE               // Otherwise
        .. Code
ENDCASE
```

```
FOR   nVar := nFirst TO nLast     // FOR NEXT loop
        LOOP
        EXIT
NEXT
```

```
// or REM                    // At the beginning of a line is a comment
/*  ..   */                  // These symbols indicate the beginning and end of the
                             // lines (exclusion of the part of the code or comment)
FUNCTION cFn_Name            // Defining F is under CMD file
PARAM cVar [, cVar2..]       // Accept input Parameters
RETURN xValue                //
```

Example:

```
Z: = ABC (2.4)                    // call of a Function
Info (z)                          // displays result: 6

return NIL

FUNCTION ABC
  PARAM  p1, p2                   // Inserting input parametars
RETURN p1+p2
```

Note: Function that has parameters cannot be declared as follows:

```
        FUNCTION ABC (p1, p2)     // this type of syntax, interpreter does not understand
```

Already has to be:

```
        FUNCTION ABC
        PARAM p1, p2
```

VAR cFile **DEFAULT** aField [LOCAL] // Sets variable and their initial value
VAR cFile **SAVE** [**LOCAL**] // Preserve the values of previously defined variables
// in file

Note: If the LOCAL file is specified, it will be preserved on the local disk. Otherwise, if the server is on WEB and cFile, it will be on the web.

If the file name cFile does not contain an extension, it will be the default **.var**
The file is being kept in the APL folder

Example:

```
VAR varfile DEFAULT {{' cName: C30 ', ' xBWEB '}, {' nYear: N4 ',2018}}
// A PRIVATE variable Cname and Nyear are set and a specific type and initial value.
// The first call is to assign initial values because they are not defined in the file "varfile"
// Next call (load variable) will SET PRIVATE variable with saved values

.. Code

VAR varfile SAVE
// The "varfile" file will be saved by the value of the variable cName and nYear that are predefined.
```

QUIT // Call QUIT procedure (exit program)

DIALOG [WIDTH nCols] // Width of the windows in characters
[HEIGHT nRows] // Window height in lines
[TITLE cTitle] // Window name
[FONT cFont] // Font (font name or font object)

// Calling method T_MSG

PAUSE nSec // Generise Pause for nsec seconds
COUNTDOWN // Display the countdown
[TEXT cText] // Display text in the last line

DIALOG OFF | CLOSE // Close dialog

@ nCol, nRow say xValue [.., xValue] // Display specified one or more values on
// selected column and row
[COLOR cColor] // Select color (option)

@ CLEAR // Delete all lines written in the window

Example:

// Open a dialup with a font selection
```
DIALOG WIDTH 40 HEIGHT 10 FONT '18.Lucida Console'

//Program code…
FOR nCnt :=1 TO nMax
    .. code
    // Show  process in percent
    @ 1,1 SAY 'Procces: ',str(100*nCnt/nMax ,3),'%'
NEXT
//End of processing

// Pause of  5 seconds with Countdown and display of ' Finish ' message
PAUSE 5 COUNTDOWN TEXT ' Finish '

//Close window
DIALOG OFF
```

Working with DBF files

DBF_OPEN {cFile, lCreate, aStruct, aIndex, lStay_DBF_Alias, cAlias, lShare}
  or
DBF_OPEN
        FILE cFile
        NEW   // default is .T.
        STRUCT aStruct
        INDEX aIndex
        ALIAS cAlias
        SHARE   // default is .T.

DBF_CLOSE nDBF_Order

DBF_ADDREC {{cField, xValue},..}

DBF_REPL {{cField, xValue},..}

DBF_SELECT nDBF_Order

DBF_INDEX | The cKey

Example:
oDBF: = NIL
/*
  Create a DBF file (a specified NEW parameter).
  Note: If the NEW parameter is not specified and the file test. DBF already exists, the file opens
        (does not delete)
*/

```
DBF_OPEN ;
    OBJ oDBF ;
    FILE 'test' ;
    NEW ;
    STRUCT {{'ID','C',3,0},{'NAME','C',20,0},{'ADDRESS','C',30,0}} ;
    INDEX 'ID'

//Add 10 syllables to a file
FOR v:=1 TO 10
    DBF_ADDREC {{'ID',strzero(v,3)},;
                {'NAME',str(v,3)+':'+Time() }}
NEXT

//Go to the first line
oDBF:Gotop()

// Show in window contents of file from oDBF.
// File:  ABROWSE.CMD is for display the contents of any Table object
Call('ABROWSE.CMD',oDBF)

//Close database
DBF_CLOSE
```

Forming a report

REPORT FILE cFile_Name.frp - Call FRP file (See Special Chapter on that)
These are forms of the Statas that are generated from the specified alias

The second method is step by step by writing on paper (line) side by side.

REPORT
[LANDSCAPE] - Orientation of Paper
[PORTRAIT] Default value is PORTRAIT
[SILENT] - Work without View processing (typically used in combination with EXPORT parameter)
[PAPER] - The FormatPaperA3 | A4 | A5 | Letter | Legal | Executive in The default value is A4
[MARGINE a,b,c,d] - Margins Paper: nLeft, nRight, nTop, nBottom Default values are: 12, 8, 8.8
[FONT] - Set An initial Font. It involves Arial. 12
[ROWS nrow] - Number of Lines on the Paper
[COLS ncol] - Number of Columns on the Paper
[OBJ orep] - Name variables in the which to get allowed what Report Object
[EXPORT cfile] - Name file in the who to get Forms reports
Can have one of the extensions:
PDF, HTML, RTF, CSV, XLS, BMP, JPEG, TXT, GIF, XML, ODS, ODT

Note: ROWS and COLS determine the number of rows and columns on the paper and are used in the designated coordinates on which to display an object by using the commands under the REPORT section. Number of Columns COLS He has default value in the Line with the selected form of a Paper:

Paper Format	Number of rows
A3	66
A4	130
A5	60
Letter	72
Legal	72
Executive	72

REPORT END | SHOW End of the report and view it in the window.

The following commands are listed between REPORT and REPORT END:
(You can certainly use standard code)

Note: the following commands are used:

AT nRow, nCol - Specifying the row and column of the view
WIDTH nWidth - Width
HEIGHT nHeight - Height

All the values above can be:

1. S with tag %. For example, WIDTH ' 50%' knows that the width is 50% of the width of the visible part Page (no margins). Or HEIGHT ' 30% ' denotes 30% of the visible part of the page.
2. Numeric value is expressed in rows or columns by definition in REPORT commands
3. The String in the line quotation begins with the label # represents Pixel.
   Example: ROW '#20 '
4. A String in the line quotation begins with a tag # if nrow isindicated line relative to the last line in which the view was performed.
   For example: ? ' One ' AT 12.0
   ? ' Two ' AT ' $ + 3 ',0
   ? ' Tree ' AT ' $-4 ',0

Thetext ' on ' on the 12th Rowis displayed. Then follows ' Two ' on the 15th Row (3 behindthe previous) and at the end of the ' Tree ' is shown on the 11th Row (-4 behind the last)

The commands:

? | PRINT xValue[, xValue [,..]]	- View Arrays value there was any type
[ALIGN]	- Alignment CENTER | C | RIGHT | R | LEFT | L
[FONT]	- Choice Font
[COLOR]	- Choice Colours
[AT nRow, nCol]	- Coordinates showing it
[WIDTH nWidth]	-
[HEIGHT nHeight]	-
[LINE nRow]	- View on specified nRow line

NEWPAGE [LANDSCAPE] [PORTRAIT] - Get to a new page with eventual change Orientation of paper

FONT nFontWeight.cFontName COLOR cColor - change the font and color
Standard value font: 11.Lucida Console, and colors: black (BLACK)

The above-stated F- execution moves the position to a new line, so the following command performs a view in the next row.

You can use command ?? who it was equivalent to earlier function with that to get Committing The view does not switch to a new line.

In addition to text printing, you can also use a graphic:

GRAPH line	// show line
AT nRow, nCol	- Coordinates
WIDTH nWidth	- Width
HEIGHT nHeight	- Height
PEN nPen	- Line thickness
STYLE cStyle	- Line printing style
COLOR cColor	- Color
ARROW	- Adding arrows at the line end: FILL | LEFT | RIGHT | UP | DOWN

GRAPH RECTANGLE | ROUNDRECTANGLE | ELLIPSE | TRIANGLE

AT nRow, nCol	- Coordinates
WIDTH nWidth	- Width
HEIGHT nHeight	- Height
PEN nPen	- Line thickness
STYLE cStyle	- Line printing style
COLOR cColor	- Frame line color
FILL cColor	- Internal color
BRUSH	-

GRAPH PICTURE

AT nRow, nCol	- Coordinates
WIDTH nWidth	- Width
HEIGHT nHeight	- Height
FILE cFile	- File name
COLOR cColor	-

GRAPH TABLE

OBJ oTab	- Set table object
AT nRow, nCol	- Coordinates
WIDTH nWidth	- Width
HEIGHT nHeight	- Height
DATA aData	- Data as a Two-dimensional array
ROWS nRows	- Number of table rows (arraysize)
SIZE aSize	- array column width. Advantages can also be specified as %
HEADER aName	- array of table column names
NOTROWS	- N

Example:

//Data defined in aData

```
aData := {{'1.','Note 1', 123,2345},;
          {'2.','Note 2',  56, 992},;
          {'3.','Note 3',6123,  38},;
          {'4.','Note 4',  12,1235},;
          {'5.','Note 5', 923,  55},;
          {'6.','Note 6', 323, 555},;
          {'7.','Note 7', 778, 155}}
```

/*

Set table objects to oTab
Table position is on row 10 and column 1
The width and table height are expressed in percentage of 80% and 15% versus the size of the paper //without margins
The raw column is defined in percentage of 10%, 50%, 20%, 20% = 100%
relative to the raw table (80% of the width paper)
HEADER define column names

*/

```
GRAPH TABLE AT 10,1 WIDTH 80% HEIGHT 15% DATA aData ;
      OBJ oTab ;
      SIZE {'10%','50%','20%','20%'} ;
      HEADER {'No','Opis','Dug','Pot'}
```

//Table View
oTab: Show ()

GRAPH BARCODE

AT nRow, nCol	- Coordinates
WIDTH nWidth	- Width
HEIGHT nHeight	- Height
DATA cData	-
TEXT cTitle	-
COLOR cColor	-
FILL cFill	-
TYPE cType	-
FONT cFont	-
ROTATION cMode	-

Notes:

STYLE - line printingstyle can be:

Square	-
Solid	- One line
Dash	- Dashed line
Dot	- Dunder Tray
DashDot	- Dashed dot line
DashDotDot	-
Double	- Double line
AltDot	-
Square	-

BRUSH - Can be:

Solid	-
Clear	-
Horizontal	-
Vertical	-
bsFDiagonal	-
bsBDiagonal	-
bsCross	-
bsDiagCross	-

Chart View***:***

GRAPH #BAR | #HBAR | #PIE | #LINE | #FLINE
- AT nRow, nCol - Coordinates
- WIDTH nWidth - Width
- HEIGHT nHeight - Height
- DATA aData - array of values
- COLOR cColor - Color or a list of colors for each element of an array
- GRID - View grid
- 3D

A series of DATA that is stated from the data can be one or two-dimensional.
Example:

```
GRAPH #BAR AT '0%','5%' WIDTH '75%' HEIGHT '18%' ;
      DATA {200,330,433,502,123,421,278} ;
      TITLE {'A','B','C','D','E','F','G'}   GRID 3D

GRAPH #BAR AT '60%','5%' WIDTH '75%' HEIGHT '18%' ;
      DATA {{2,3,4,5,6},{6,5,8,12,3}} GRID 3D ;
      COLOR GRAY,BLUE
```

BROWSE - Tabular display data of the selected work area
See the W_TAB class in a detailed explanation.

BROWSE
- ROW | Y // 1
- COL | X // 2
- WIDTH // 3
- HEIGHT // 4
- WINDOW | DIALOG | OF // 5
- TITLE // 6
- FONT // 7
- ALIAS // 8
- FIELD // 9
- HEADER | HEADRES // 10
- BODY | TRANS | TRANSFORM // 11
- SIZE // 12
- ALIGN // 13
- PICTURE // 14
- VALID // 15
- LINK // 16
- COPY COLUMNS | COPY // 17
- FILTER // 18
- VTAB // 19
- VTAB_SIZE | VSIZE // 20
- SORT // 21
- BUTTON // 22
- ON KEY | KEYB | KEYBOARD // 23
- ADD // 24
- ADD_CHOICE // 25
- DEL // 26

```
        DEL_CHOICE                  // 27
        INS                         // 28
        INS_CHOICE                  // 29
        MIN EDIT                    // 30
        TOTAL                       // 31
        ON END | ON CLOSE           // 32
        HSCROLL                     // 33
        SEEK                        // 34
         READONLY                   // 35
         TRIGER                     // 36
         LINK_TAB | JOIN TAB        // 37
         RESIZE                     // 38
         USER MENU | RCLICK         // 39
         PIC FIELD                          // 40
         ADI MENU                   // 41
         COMBO ITEMS | ITEMS        // 42

END BROWSE | THE ENDBROWSE | END


 ADD COLUMN | COLUMN
        FIELD
        HEADER | TITLE
        BODY | TRANS
        SIZE | WIDTH
        ALIGN
        PICTURE | PIC
        VALID
        LINK
        COPY
 END COLUMN | ENDCOLUMN | END

 ADD BUTTONS
        FONT cFont
        SEPARATOR

    MENU
        ITEM ACTION
        SEPARATOR
        SIZE
    END

 END BUTTONS | ENDBUTTONS | END

 PIC    FIELD
        ON CLICK
        BORDER
        SIZE nWidth [, nHeight]
```

Dialog and enter the value of specific variables

DATA DLG oDlg - oDlg it is object of data dialog
[TITLE] - Name dialog window
[FONT] - A Font
..
END DLG | ENDDLG

Example:

The variables that are specified in the commands between DATA DLG and END DLG must be predefined:

```
cName    := PadR("TITLE",40)
cCountry := 'Serbia (+381)'
cCtry    := '+381 '
dDate    := date()
dDate2   := date()
lAct     := .F.
nRadio   := 1
cMemo    := space(1024)
nID      := 1
```

```
DATA DLG oDlg TITLE "Edit data" FONT "14. Arial"

//Print text in the middle
SAY "Hello !" CENTER COLOR BLUE FONT 18.Times New Roman Bold Italic

//Color can also be specified as RGB 3xHEX value. In the example below, the bright red #FF0000
LINE COLOR #FF0000

//Name and right of the Get field
GET "Name" VAR cName COLOR_TEXT RED FONT_TEXT "14. Arial"

//ID Numeric Field
GET "ID" VAR nID WIDTH 5 COLOR_TEXT MAGENTA COLOR RED FONT "10. Courier";
PICTURE # # #. # # valid {| | nID > 0}

//ComboBox, possible entry of empty value EMPTY, link -> link to table COUNTRY
COMBO "Country" VAR cCountry LINK COUNTRY EMPTY

//The Get field associated with the COUNTRY table
//Call a table with F2 from the Get field or click the button to the right
GET "Country" VAR cCtry LINK COUNTRY EMPTY

//Date field
DATE "Date of Birth" VAR dDate COLOR RED

//Date field
GET "Date of Birth" VAR dDate2

//Check box
CHECK "Active" VAR lAct
// Memo-Text in 5 lines
MEMO "Text" VAR cMemo HEIGHT 5 WIDTH 30 COLOR RED COLOR_TEXT BLUE
```

```
// Radio buttons
RADIO "Choice" VAR nRadio ITEMS {"One", "Two", "Tree"}

// End of command quote
END DLG

//Views Dialog
IF ! Odlg: Show ()
   // If the entry was interrupted (canceled with ESC or Cancel)
    RETURN NIL
ENDIF
... Code
```

Note: Between DATA DLG and END DLG cannot be cited, declarations of variables, etc.
Only the following commands are listed:

```
LINE [COLOR]                    -  show line
Example: LINE COLOR RED

SAY cText     [COLOR cColor]    - Text color
              [FONT cFont ]     - A Font
              [CENTER           - Center text

Example: SAY ' Hello!!! ' CENTER COLOR RED FONT 14. Arial Bold

GET cText     VAR cVar          - Predefined variable
              [PICTURE]         - The Mask for editing (Example: # # ##. ##)
              [WALID]           - Condition
              [COLOR]           - Colors for a cVar
              [FONT]            - Font for cVar
              [COLOR_TEXT]      - Colors for a cText
              [FONT_TEXT]       - Font for cText
              [LINK         ]           - Connecting cVar variable with the table
              [EMPTY]           - The field can be blank
              [SPINNER]         - If the variable is Numeric you can use this clause

The variable type (cVar) can be: Numeric, Logical, Date, Char

Example: GET 'Number' VAR nVar  PICTURE ### SPINNER
        or
           GET 'Name' VAR cName PICTURE @! VALID !Empty(cName)
        or
           GET 'Country' VAR cCountry LINK sCountry EMPTY

MEMO cText VAR cMemo
              HEIGHT nH
              [ WIDTH ]         - Cited to get field width
              [ COLOR ]         - Colors for a cMemo
              [ FONT ]          - Font for cMemo
              [ COLOR_TEXT ]    - Colors for cText
              [ FONT_TEXT ]     - Font for cText

Example: MEMO ' Text ' VAR cMemo HEIGHT 5
```

DATE cText VAR dDate
[COLOR_TEXT] - Colours for a cText
[FONT_TEXT] - Font for cText

COMBO cText VAR cCombo
[LINK]
[EMPTY]
[ITEMS] - A Series of value (called) for choice
[COLOR_TEXT] - Colors for aCtext
[FONT_TEXT] - Font for cText

Example: COMBO ' Country ' VAR cVar LINK cCountryTable_Name EMPTY
or
COMBO ' Select ' VAR cVar ITEMS {'One ', ' Two ', ' Tree'}

RADIO cText VAR nRadio
[ITEMS] - A Series of value (names)
[COLOR_TEXT] - Colours for a cText
[FONT_TEXT] - Font for cText

Example: RADIO ' Select ' VAR cvar ITEMS {'One ', ' Two ', ' Tree'}

CEHCK cTextVAR nRadio
[COLOR_TEXT] - Colors for a cText
[FONT_TEXT] - Font for cText

Example: CHECK ' Status ' VAR cVar

The Notes:

COLOR cForeground [,cbackground] - Color
or
COLOR_TEXT cForeground [,cbackground] - Pre-text color

Colors can be:
BLACK | WHITE | LINE NO. | BLUE | GREEN | YELLOW | DARK IN SERBIA | BROWN | MAGENTA | GRAY or in HEX format #RRGGBB
Default color is BLACK
FONT cFont - Can be in format:12. Arial, Arial. 12,14. Times New Roman
This can also be specified by BOLD and/or ITALIC
Example: 12. Arial Bold or 14. Times New Roman Bold Italic

VAR cVar - The name of the PRIVATE variable predefined

LINK cTableName - Link to table

EMPTY - Field can be blank or one of the values in the linked table

==========

```
FRP
        TITLE                   //  1
        HEADER                  //  2
        FOOTER                  //  3
        BFOOTER                 //  4
        TEXT                    //  5
        NEWLINE                 //  6
        FIELDS                  //  7
        COLUMNS                 //  8
        COLUMNS2                //  9
        TRANS                   // 10
        SIZE                    // 11
        ALIGN                   // 12
        PROPER                  // 13
        TOTAL                   // 14
        ROW2                    // 15
        FONT_COLUMNS            // 16
        TOTAL_TITLE             // 17
        START_FN                // 18
        START_FN2               // 19
        ROW_IF                      // 20
        ROW_FN                  // 21
        ROW2HEIGHT              // 22
        ROW2LINE                // 23
        FONT_TABLE              // 24
        BODY                    // 25
        TAB_HEADER_COLOR        // 26
        TAB_HEADER2_COLOR       // 27
        TAB_HEADER_ALIGN        // 28
        LINE_SPACING            // 29
        STYLE                   // 30
        FONT                    // 31
        FONTS                   // 32
        FONTMAXCOL              // 33
        COLOR                       // 34
        SQL                     // 35
        DATA                    // 36
        BOTTOM_LINE_PAGE        // 37
        BOTTOM_LINE_TABLE       // 38
        WIDTH_LINE              // 39
        ALIAS                   // 40
        GRAPH                       // 41
        FILTER                  // 42
        LANDSCAPE               // 43
        PAPER                   // 44
        NEXT_REP                // 45
END FRP | ENDFRP | END
```

Instead of the quotation mark of the values in FIELDS, HEADER, SIYE...

Columns can be specified individually one by one.

```
ADD COLUMN
      FIELD
      HEADER | TITLE
      HEADER2 | TITLE2
      BODY | TRANS
      SIZE | WIDTH
      ALIGN
      PROPER | PROPERTIES
      TOTAL
      BODY2 | TRANS2
      FONT
END COLUMN | ENDCOLUMN | END
```

Example:

```
FRP
   SQL "SELECT * FROM..."

   ADD COLUMN
       FIELD cField_1
       HEADER "Field name 1"
   END COLUMN

   ADD COLUMN
       FIELD cField_2
       HEADER "Field name 2"
   END COLUMN

END
```

or original variant:

```
FRP
   SQL "SELECT * FROM ..."
   FIELD { cField_1, cField_2 }
   HEADER { "Field name 1", "Field name 2" }
END
```

Files with extension the .MNU contains menu definitions.

The call and execution of such files is done in the following way:
CALL ("TEST. MNU ")

Every .MNU file contains a native menu (TITLE) and multiple definitions for each menu separated in sections [PROMPT]

Example:

```
TITLE = Choice report from
[PROMPT] = Report-(Form 1)
CALL = {| y, x | Call ("TAB3_1. FRP")}
[PROMPT] = Report-(Form 2)
CALL = {| y, x | Call ("TAB3_2. FRP")}
[PROMPT] = Report-(Form 3)
CALL = {| y, x | Call ("TAB3_3. FRP")}
IF= {| |  oxbweb: nlevel = 1}
```

Explained by:

TITLE - Basic Menu Name
[PROMPT] - For each submenu (menuitem) , The name is named below the [PROMPT] declaration canbe specified:
CALL - Can be a block code or char that specifies an f ' called
IF - The condition of a menu
In the exampleabove , if the user has a level 1 privilege , itwill display the third menu "Report - (Form 3)"

Formation of tabular reports. Files with the .FRP extension contain report definitions.

The call and execution of such files is done in the following way:
CALL ("TEST. FRP ")

Every .FRP file contains a series of definitions:

```
FONTS= {' 10. Arial ', ' 11. Arial ', ' 9. Arial ', ' 12. Arial ', ' 14. Arial ',;
        ' 10 Arial ', ' 10. Arial ', ' 10. Arial ', ' 20. Arial ', ' 10. Arial '}
```
You can specify a list of possible fonts that are used in your expertise
You need to specify a number (one from this series) later when selecting a font.
For example: FONT= 2 - This is determined that the table FONT is 2 (second from the specified array) and that is 11. Arial

FONT= Basic font that is applied in other declarations if the font does not specify
It canbe: Numeric - number from a string of specified fonts
Text - font is specified in the format: "12. Arial" (size and the font name as stated)

FONT_TABLE= Font of the table. Can be: numeric | ctring_Font
If not specified, the base font is taken

FONT_COLUMNS= a number of fonts for each table column individually (the columncanbespecified or be empty when a basic font is applied)

FONTMAXCOL=

COLOR= cColor // according to standard color

cColor can be one of the colors:

BLACK -
MAROON -
GREEN | -
OLIVE -
NAVY -
PURPLE -
TEAL -
GRAY -
SILVER -
Red -
LIME -
YELLOW -
Blue -
AQUA -
White -
SKYBLUE -
BRUSH1 -
BRUSH2 -
BRUSH3 -

If a declaration is specified: TITLE | HEADER | FOOTER | BFOOTER
This means the beginning of one of the sections in which the textis specified.

TITLE - textdisplayed at the beginning of the report (on First page)
HEADER - text that is displayed before the table (on each page)
FOOTER - textdisplayed by the table (every page)
BFOOTER - textdisplayed at the end of a report

Below each section can be specified IF ...ENDIF condition and within it TEXT or past it TEXT
IF= lCondition
NEWLINE
TEXT = { cText,
cAlign,
cFont,
cMacro,
cColor,
cX,
cW,
lVA,
lWW,
cY }

Example:

```
TITLE
Text= {' TITLE NAME ', "L",0}

HEADER
Text= {' Str. # [Page] ', ' R ' 2}

FOOTER
Text= {"Footer text"}

BODY
STYLE = STANDARD
```

Declarations of TEXT and NEWLINE can be specified by IF and ENDIF declarations
Example:

```
...
Start_FN= {|| _L1  := ChoiceYN('Write A or B ? ') }
...
IF= L1
  Text={'Say A'}
  NewLine=
ENDIF
IF= ! L1
  Text={'Say B'}
  NewLine=
ENDIF
```

SQL= cSQL_Query // Can Specify SQL query or blank

ALIAS= oAlias // If no SQL query is specified, thenThis specifies the table object
// (source data)

GRAPH= cCMD // Specifies a CMD file that is performed by calling FRP report
// this way , calling the FRP file indirectlycalls the CMD from whichI can
// the functions

Example:

```
LANDSCAPE=.F.
FONT = 0
TITLE
GRAPH = 'GRAPH. CMD'
DATA= {}
BODY
BSTYLE = STANDARD
COLUMNS= {''}
FIELDS= {''}
TTRANS= {''}
```

GRAPH. CMD file:

```
PARAM oGr
   oGr:Graph ('TEXT',{'XbWEB','C','16.Arial'},'0','0','80','2')
   nPg := oGr:Graph ( 'EJECT' )
RETURN NIL
```

GRAPH. CMD is transmitted by the barring parameter (graphic object) that is further used in CMD file.

LANDSCAPE = block code | CHOICE |. T. | .F.

If specified:

LANDSCAPE = CHOICE
When calling a report, the question is automaticallyreturned:
Landscape orientation? Yes/No

Thiscouldbeas follows:

```
LANDSCAPE= {| |ChoiceYN ("Landscape orientation? ") }
```

or LANDSCAPE= .T.
Now there is no question whether the print will be a Landscape, but it is implicitly Specified

PAPER = ***cPaper***

Or specify a series of seven elements:

{ cpaper - Can be: ***cpaper***
nLeft_Margine, - Marige paper
nRight_Margine, -
nUpper_Margine, -
nDown_Margine, -
nColumns, - Number of thecolumns of the pasted column on the visible part of the
nrows } - Number of imaginary lines (lines)

cPaper canbe: A4 | A3 | A5 | Letter | Legal | Executive in

-

STYLE = STANDARD | DASH | NONE // StilViewTables

Example: STYLE = STANDARD

TITLE NAME

Str. 1

Account	Document	Date	Debit	Credit	Description
0000	00001	01.01.2018	1.00	-1.00	
Total . . . . . :			1.00	-1.00	

Footer text

Example: STYLE = DASH

TITLE NAME

Str. 1

0000	00001	01.01.2018	1.00	-1.00	
Total . . . . . :			1.00	-1.00	

Footer text

Example: STYLE = NONE

TITLE NAME

Str. 1

0000	00001	01.01.2018	1.00	-1.00	
Total . . . . . :			1.00	-1.00	

Footer text

TAB_HEADER_COLOR = Table header color (cColor)
TAB_HEADER2_COLOR=
TAB_HEADER_ALIGN= cAlign (can be stated: 'L' | 'R' | 'C')
Column name Alignment
LINE_SPACING= numeric (default: 0)

BOTTOM_LINE_PAGE=.T. | .F. (default: F.)
Ifspecified , theline at the bottom of the page is displayed

BOTTOM_LINE_TABLE=. T. |.F. (default: .T.)
If not specified,the line at the bottom of the table is displayed:
Example:

TITLE NAME

Str. 1

Account	Document	Date	Debit	Credit	Description
0000	00001	01.01.2018	1.00	-1.00	
Total . . . . . :			1.00	-1.00	

Footer text

WIDTH_LINE = nWidth // line thickness (standard 1).
If 0 is specified , no lines will be displayed
Example: WIDTH_LINE = 3

TITLE NAME

Str. 1

Account	Document	Date	Debit	Credit	Description
0000	00001	01.01.2018	1.00	-1.00	
Total . . . . . :			1.00	-1.00	

Footer text

START_FN = block code which is performed before the execution of reports

NEXT_REP= if specified , then it is the name of the report that is executed in the continuation of the commencement

Declarations related to table columns:

COLUMNS= Array of column names.
Example: COLUMNS= {' ID user ', ' Name user ', ' Date ', ' Debit ', ' Credit'}
COLUMNS2
TRANS= Array of column views (string valued as macro)
FIELDS= Array name of the field from the table.
Example: COLUMNS = {' ID ', ' NAME ', ' DATE ', ' DEBIT ', ' CREDIT '}
SIZE= The size of the column (each individually)
If the size of each column does not specify , it is automatically determined by the length of the declaration from TRANS Array
ALIGN= Array of individual columnalignment labels.MarkingsMugubeing: L-LeftorR-Right or C-center. The standard value is L or N If the field in the numeric
Example: {' L ', ' L ', ' C}
PROPER= Series of additional attributes of each column individually
Each element ifspecified is an array:
{ cAlign, - Text alignment (L | R | C -> horizontally: left, right or center)
cFont, - Font
cColor, - color (ccolor)
lVert, - vertical display of text (standard is horizontal)
cValign, - vertical alignment
lWordWrap } - if the text is too Long is transferred to the subsequent line
(if it is. T.)
TOTAL= A series of columns that are totaled at the end of a table
Example: TOTAL ={,,,' CREDIT ', ' DEBIT '}
The last two columns are totaled
TOTAL_TITLE= cText -the text that appears in the beginning of the line (last) in which the displays the columns. Example: TOTAL_TITLE = Title

ROW2= Series of display columns in the second row in the table.

In addition to the baseline , it is possible to import a second line display for each column individually.
On that occasion , the text thatreturns as a view can contain the symbol chr(10) that exceeds to the next row and displays the text

ROW2HEIGHT= 1 // height in second row rows

Example:

```
...
    Columns = {' Account ', ' Document ', ' Date ', ' Debit ', ' Credit ', ' Description '}
    Fields = {' ACCOUNT ', ' DOC ', ' DATE3 ', ' DEBIT ', ' CREDIT ', ' DESCRIPTION '}
    Trans = {' ACCOUNT ', ' DOC ', ' d2c (DATE3) ', ' Str_z (DEBIT) ', ' Str_z (CREDIT) ',
                ' DESCRIPTION '}


    ROW2HEIGHT = 2
    Row2 = {' D2c (Date ()) + CHR (10) + time () ',,,,,}
    Proper = {{,,,, ' T ',}, {,,,, ' T ', {,,,, ' T ',}, {,,,, ' T ',}, {,,,, ' T ',}, {,,,, ' T ',}}
```

Now we get a view of the current date and time in the second row
Declaration of Proper (.. 'T') determinedis to alignment vertically be T | TOP on the topside

We will get the following view:

TITLE NAME

Str. 1

Account	Document	Date	Debit	Credit	Description
0002 13.01.2019 17.34.46	00002	01.02.2018	1.00	1.00	
0003 13.01.2019 17.34.46	00003	01.02.2018	3.00	3.00	
Total . . . . . :			4.00	4.00	

Footer text

ROW_FN = block Code (Fn-function) which is performed before the execution of each line

ROW_IF= block code (Fn) a value (. T. | .F.) determining the condition of the order of

WTAB -The work area display tabular great possibilities

Using source code .PRG (also a .CMD file)

```
LOCAL Inverw

oBrw: = T_WTAB (): New ( nCol,;          // Upper left column
                          nRow,;         // Upper left corner
                          nWidth,;       // Table width
                          nHeight,;      // Table height
                          oDlg,;         // cTitle | {oDialog, cTitle} | NIL-Dialog
                          ocFont,;       // cFont_Name Or oFont
                          oAlias,;       // Object Alias
                          aFld,;         // Array Of Field_Name
                          aTitle,;       // NIL or array title columns (header)
                          aTrn,;         // NIL or array transform (array string or code block)
                          aSize,;        // NIL or array size each column in pixels
                          aAlign ,;      // NIL or array Tarna columns (L-left or C-Center)
                          aPic,;         // NIL or array picture from edit columns
                          aVal,;         // NIL or array validation (string or block code)
                          aLink,;        // NIL or array Link of codebook table
                          aCopy,;        // NIL or Array nField_Pos or cField_Name
                          aFilter,;
                          nVTab,;        // Begin column position of show vertical table
                          VTab_Size,;    // Width Vertical Table
                          aSort,;        //{Sort_Name,..}
                          aButton,;      // Array buttons
                          aKeyb   ,;
                          cADI_Mnu ,;
                          lAdd    ,;
                          lAdd_Ch  ,;
                          lDel    ,;
                          lDel_Ch  ,;
                          lIns    ,;
                          lIns_Ch  ,;
                          nMin_Edit,;
                          aTotal   ,;
                          bEnd    ,;
                          lHScroll ,;
                          cFSeek   ,;
                          lReadOnly,;
                          aTriger  ,;
                          aJoin_Tab,;
                          lResize  ,;
                          aRClick_UserMenu ,;
                          aPic_Fld ,;
                          aItems )
```

Or translate commands (xbweb.ch from source code. PRG)

```
W_TAB <oWTab> ;
    TO <nRow>, <nCol> ;
    WIDTH <nWidth> ;
    HEIGHT <nHeight>
    TITLE | OF | WINDOW | DIALOG <oDlg> ;
    FONT <ocFont>  ;
    ALIAS <oAlias> ;
    FIELD | FIELDS <aField> ;
    HEADER | HEADERS <aHeader> ;
    TRANS | TRANSFORM <aTrans> ;
    COMBO ITEMS | ITEMS <aItems> ;
    SIZE <aSize> ;
    ALIGN <aAlign> ;
    PICT | PICTURE> <aPict> ;
    VALID <aValid> ;
    LINK <aLink> ;
    COPY | COPY COLUMNS <aCopy> ;
    FILTER <aFilter> ;
    VTAB <nVTab> [SIZE <nVTab_Size>] ;
    SORT <aSort> ;
    BUTTON <aButton> ;
    KEYB | KEYBOARD <aKeyb> ;
    RCLICK MENU | RCLICK <rClick> ;
    USER MENU <rClickUM> ;
    ADD <lAdd> [ CHOICE <lAdd_Ch> ;
    DEL <lDel> [ CHOICE <lDel_Ch> ;
    INS <lIns> [ CHOICE <lIns_Ch> ;
    MIN EDIT | MIN EDIT COLUMNS <nMin_Edit> ;
    TOTAL <aTotal> ;
    ON CLOSE | ON END <bEnd> ;
    HSCROLL <hScroll> ;
    SEEK <cSeek> ;
    READONLY <lbReadOnly> ;
    TRIGER <aTriger> ;
    JOIN_TAB <aJoin_Tab> ;
    RESIZE <lResize> ;
    PIC_FLD cField_Name [SIZE <cPicFld_W>,<cPicFld_H>] [BORDER>] [ON CLICK
                <bClick> ]   ;
    COLOR [ HEADER nForeClr [,nBkgrClr]]    [ FOCUS nForeClr [,nBkgrClr]]
```

A CMD file is possible to create an object of *OBRW: = T_WTAB (): New (*..as specified by the
In the following way (substitute xbweb.ch in source code)

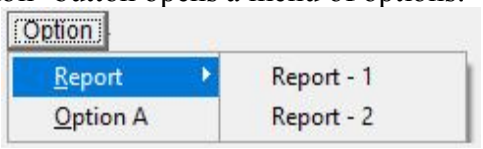
```
BROWSE      oBrw
            WINDOW | DIALOG | OF   oDlg
            ROW  nRow
            COL nCol
            WIDTH   nWidth
            HEIGHT   nHeight
            TITLE   cTitle
            FONT   coFont
            ALIAS   coAlias
            FIELD   aFields
            HEADER | HEADRES   aHeaders
            BODY | TRANS | TRANSFORM   aTrans
            SIZE   aSize_Col
            ALIGN   aAlign
            PICTURE   aPicture
            VALID   aValid
            LINK   aLink_Tab
            COPY COLUMNS | COPY   aCopyCol
            FILTER   aFilter
            VTAB   nVTAB_PosCol
            VTAB_SIZE | VSIZE   nVTAB_Width
            SORT   aSort
            BUTTON   aButtom
            ON KEY | KEYB | KEYBOARD   aKeyb
            ADD    lbAdd
            ADD_CHOICE lbAdd_Choice
            DEL     lbDel
            DEL_CHOICE lbDel_Choice
            INS     lbIns
            INS_CHOICE lbIns_Choice
            MIN EDIT   nMin
            TOTAL    aTotal
            ON END | ON CLOSE    bEnd
            HSCROLL     lHScrol
            SEEK    cSeek
            READONLY   lReadOnly
            TRIGER    baTriger
            LINK_TAB | JOIN TAB   aJoin_Tab
            RESIZE    lResize
            USER MENU | RCLICK   aRClick_UserMenu
            PIC FIELD    aPic_Fields
            ADI MENU   cADI
            COMBO ITEMS | ITEMS  aItems

END BROWSE
```

oDlg - Dialog where the table is called. If not specified, the dialog box that will completely contain the table
nRow - Upper-left corner of table
nCol - Upper left corner of table
nWidth - Table width
nHeight - Table height
Note: nRow, nCol, nWidth, nHeight can be numeric or string with Tag % relative to window size
cTitle - Name of window: cTitle | {oDialog, cTitle} | NIL-Dialog
coFont - Example: ' Arial. 14 ' (cFont_Name or oFont)
coAlias - You can specify an object named or work area name (example: ' TAB1 ')
The work area must be preopened
aFields - Array of fields of each column in a table
aHeaders - Series of column names
aTrans - array string or block code values to display data in the body of a table (table rows)
The String value is executed as a macro
aSize_Col - array of numeric values of the raw column in pixels
aAlign - A series of values for each column that can be:
R | RIGHT or L | LEFT
aPicture - array string or block code values
If an individual column is worth '. ' (The wheel, according to), is skipped
aValid - Array of values: Logical, Block, or String
The String value is executed as a macro
aLink_Tab - A series of values {' LINK_TAB ', cCall} (LINK to key table)
aCopyCol - A series of column values that can be copied
These values can be field names or column numbers.
aFilter - A series of filter values in a table.
nVTAB_PosCol - column of which table data is displayed in vertical view to the right of table (vertical view of columns one of another)
nVTAB_Width - Vertical view width
aSort - A series of possible options for sorting a table (a series of key table fields)
aButton - a series of buttons around a table.Eachvalueistextwhocanbeing
BAR LEFT UP | TOP | DOWN | BOTTOM
- Thisdetermines the reposition of buttons to the left of the table from top to bottom (UP or TOP)orCounter - from belowup
BAR- Right - analog previous buttons form to the left of the table
BAR UP | TOP LEFT | RIGHT - buttons are being placed above the table from left right or right to left
BAR DOWN | BOTTOM LEFT | RIGHT- analog previous buttons are form under the table
SIZE y, x - possible to change the size of the button
FONT - font change
SEPARATOR - blank field
ADD - Add a line in a table
DEL - Delete a row in a table
INS - Insert a line in a table
PRINT or PRINT:file_name - Call ::P rint methods and possible change of name an executive call
MENU- user menu:
FIND - the search Charm
SORT - button to sort
VTAB - Outline of fields vertically
FS - Quick Search (fast search)

Example: BUTTON {' BAR BOTTOM Tarna right ', ' vtab',' ADD ', ' DEL ', ' INS ',;
{'FONT','Arial.12'},;
{'MENU',"&Option",;
{{'&Report',"PhoneBook.MNU"},;
{'&Option A',{|| Info("Hello !",,"Option A ..")}}},;
80,18},;
'BAR BOTTOM','FS:','SORT:','FILTER'} ;

Below the tablewill display buttons like picture:

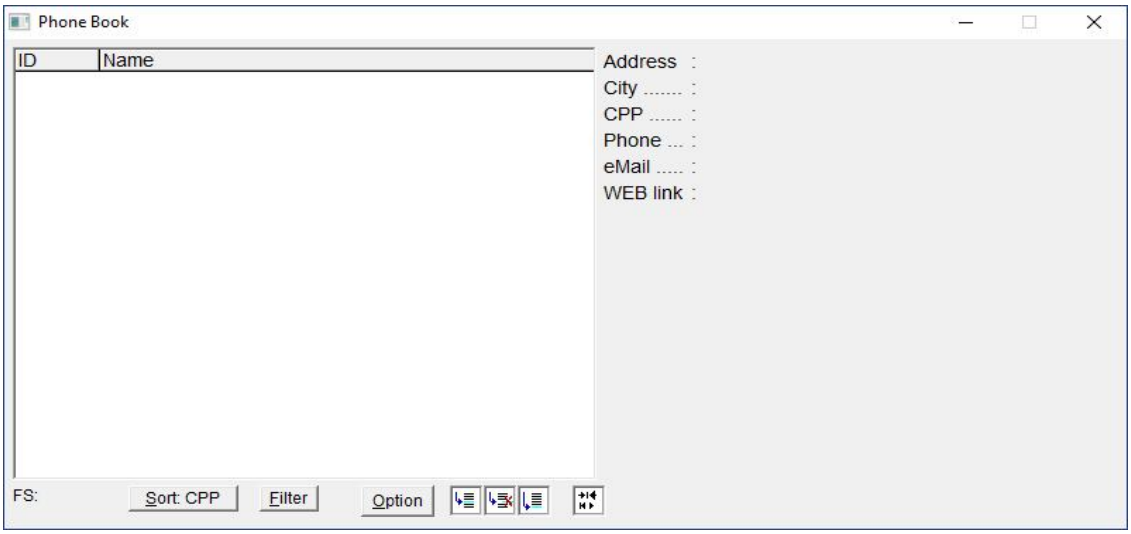
Clicking "Option" button opens a menu of options:



Report is an option that is specified but because it calls . MNU file that
The marked menus are First read from the PhoneBook.MNU file
and add it to the displayed menu.
"Option A" is the second item on the menu. Clicking on theher we'il Get A
message "Hello .. "

Contain the PhoneBook.MNU file:

TITLE = Choice
[PROMPT] = Report-1
CALL = {| Y, x| Msgbox("Report 1")}
[PROMPT] = Report-2
CALL = {| Y, x| Msgbox("Report 2")}



VTAB option is displayed as a button: 

In the parameter tabledefinition: VTAB 3
VSIZE 360

Certain that starting with column 3 displays table fields vertically on the right of the table.
The view is 360 pixels

By Clicking this button , the tableturns into a classic view whena vertical view is blocked.
Picture below:

aKeyb -
lbAdd - If it's. T. You can add a row in the table< INS > or from the menu by clicking right mouse button
The standard value is.T.
lbAdd_Choice- If it is .T. before the operation will follow the question "Add Yes/no"
The standard value is. T.
lbDel - if it is .T. It is possible to delete line < DEL > or from the menu click on the right mouse button
The standard value is. T.
lbDel_Choice - If it is . T. before the operation, the question is "swabYes/no"
The standard value is. T.
lbIns - If it is .T. You can insert a line with < SHIFT > + < INS > or from the menu
Clicking the right mouse button
The standard value is. T. with the condition That The variable is oAlias:lSNo =.T.
Or that the key field of the working area is as a numeric - ordinal number
lbIns_Choice - If it is .T. before the operation will follow the question "Insert Yes/no"
The standard value is. T.
nMin - Column number after which the row in the table is added when you add a new line
For example: if this value is 2 and THE table has fields: ID, NAME, ADDRESS ...
It is sufficient to enter values in the ID and NAME fieldsand if in the next
ADDRESS
Stop editing the ESC row will be memorized
aTotal - Series the column name of the columns to be totaled
bEnd - block called at the output (closing) of the table
lHscroll - Can be a horizontal scroll table (value. F.)
The standard value is.T.
cSeek - Use what table indirect call (with F2)
To perform the initial positioning at the specified value.

lReadonly - if it is a value . F. the editing of the table will be disabled
bTriger - Could be a series of values or Block
The call isin a certain situation in the table
If array is listed, the two cCmd and cCall values
Possible values for cCmd:
* DEL-PRE, DEL-POST,DEL-IF - before, after & condition deleting a row
* INS-PRE, INS-POST,INS-IF - before, after & condition inserting a line
* ADD-PRE, ADD-POST,ADD-IF - before, after & condition adding a row
* EXPORT-PRE, EXPORT-POST - export data in a table before/after
* IMPORT-PRE, IMPORT-POST - import data in a table before/after
* EDITCELL - when editing a column at the end of a value entry
Transfers the value of the column position
* EDIT - Pall the columns in the table
* SORT - after a table is performed to sort
* SEEK - after the overturn of the table
* VEDIT - after entering fields in the vertical table
* FILTER - after execution of the table filter
* PRINT - afterexecution ::P rint methods

The cCall is called the call function with the transfer of parameters: oSelf and nCol
nCol exists as column number at EDITCELL
Example: bTriger := {|cMode, oSelf, nCol|
Call ("TRIGNER.CMD",cMode, oSelf, nCol)}
or
bTriger := {{"PRINT", "TPRINT.CMD"}}
After execution::P Ritn method will be called TPRINT.CMD
With oSelf transmission, nCol
aJoin_tab - A series of values for merging tables
lResize - Enable the change of the raw and height (resize) tables
aRclick_UserMenu - a series of user menu values that are returned by clicking the rightmouse button
aPic_Fields - {cField, nWidth, nHeight, lBorder, bOnclick}
Supported formats: JPG, JIF, GIF, BMP, DIB, RLE, TGA, PCX
cADI - String that can contain "ADIECS MX" markings
A-Add, D-Delete, I-Insert, E-Edith, C-Copy
S or blank - delimiter
M-Import, X-Export
aItems - If the column is ComboBox, then a series of values is specified
For more detailed explanations about the use of individual paramaters, see the following example:

Sample application demo on WEB portal xbweb.rs

PUBLIC Variabla oXbWEB is a basic class object variabla the oDB in the class oXbWEB indicates on the database object that we accessed.
This object can be accessed by an Acces (database on a local disk) or MySQL database on a WEB server.

This example uses the work areas in the Tebel: TAB1, TAB2, and TAB3
TAB1 is a table with financial order groups ID1
The TAB2 table contains the journal numbers within the corresponding journal batch ID1. The ID2 field contains the journal number of the selected journal batch ID1
Table TAB3 contains specific bookings within the selected account (the key is ID1 and ID2)

In the table ID3 is the serial number of the entry in the journal
The table associations (relationships) are the following:
oTab2-> ID1 = = oTab1-> ID1
oTab3-> ID2 = = oTab2-> ID2

Example file: FIN. Cmd

```
// Method: oAlias returns the work area (alias) if it is defined
oA1: = oXbWEB: oDB: oAlias ("TAB1")
// If it's not defined, the value of the NILE returns
IF oA1 = NIL
   // Add a work area called TAB1
   oA1: = oXbWEB: oDB: Alias_Add ("TAB1")

   // Setting SQL Experiment
   oA1: Query (, "TAB1", "*",, {"ID1"})
   // The key field in the table is ID1
   oA1: cKey_Field: = "ID1"
ENDIF

// Follows the same procedure about opening the work area oA2
oA2: = oXbWEB: oDB: oAlias ("TAB2")
IF oA2 = NIL
   oA2: = oXbWEB: oDB: Alias_Add ("TAB2")
   oA2: Query (, "TAB2", "*", {"1 = 2"}, {"ID1, ID2"})
   oA2: cKey_Field: = "ID2"
   oA2: lAutoKCode: =. T.
   oA2: aNewRec: = {{"DATE2", Date ()}}
The ENDIF

// The same applies to the working area oA3
oA3: = oXbWEB: oDB: oAlias ("TAB3")
IF oA3 = NIL
   oA3: = oXbWEB: oDB: Alias_Add ("TAB3")
   oA3: Query (, "TAB3", "*", {"1 = 2"}, {' ID1, ID2, ID3 '})
   oA3: cKey_Field: = "ID3"
   oA3: lAutoKCode: =. T.
   oA3: lSNo: =. T.
   oA3: lIns: =. T.
   oA3: aNewRec: = {| | {{"DATE3", oA2: Get (' DATE2 ')}}}
ENDIF
```

In the opening of Alaiasa oA3 a parameter oA3: lSNo: =. T. and oA3: Lynns: =. T.
This means that the ID3 field declared in cKey_Field has a serial number and is enabled to insert serial numbers. Inserting rows retreats the next sequential numbers by one as the deletion of the row, which pulls down the following serial numbers by one.

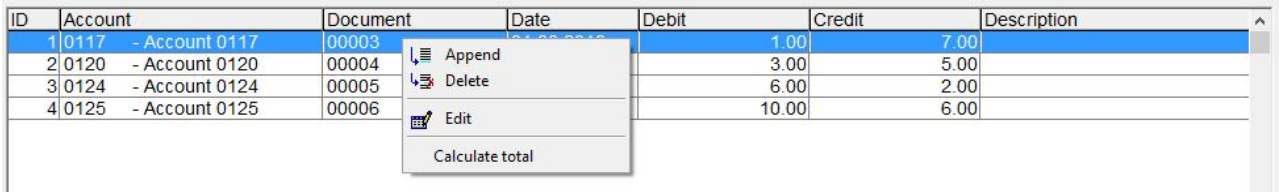
```
// Open dialog
ooDlg: = oXbWEB: XClass ("TDIALOG", 0.0, 650.1020,;
                ' Finance ',,,,,,,,;
               oXbWEB: oWnd,. T.,, oXbWEB: Font (' Arial. 14 '))
```

// Soting PRIVATE variables (need by for settings to JOIN_TAB parameters)
oTab1: = NIL
oTab2: = NIL
oTab3: = NIL

```
// We set the Browse object in the dialog ooDlg

BROWSE oTab1
        OF ooDlg
        ROW 5
        COL 5
        WIDTH 250
        HEIGHT 140
        FONT ' Arial. 14 '

// Work Area object (alias)
ALIAS oA1

// Field names from the current aliases of each column
FIELD {' ID1 ', ' DESCRIPTION '}

// Column names
HEADER {' ID ', ' Description '}

// Size of columns in pixels
SIZE {80.170}
PICTURE {,}

// Horizontal scroltable Disconnected
THE HSCROLL. F.

// Validation can be defined for each column
"WALID"! Empty(ID1)','! Empty (DESCRIPTION) '}

// Clicking the right mouse button opens a menu with A-ADD (add), D-Delete options, E-Edith
ADI MAN ' AD E '

// In the Browse (tables) box, the button at the bottom is added.
BUTTON {' BAR RIGHT TARNA BOTTOM ', ' ADD ', ' DEL '}


// If K_RIGHT is pressed in table 1, the focus switches to table 2
ON KEY {' K_RIGHT ', {| | oTab2: oBrw: SetFocus ()}}}

// Join table oTab1 with table oTab2 (key: ID1)
// Moving through Table 1 oTab1 calls a requery (reQuery) work area oA2 in a table
// oTab2 with key ID1
// So by moving through the table oTab1 the contents of tabele_2 oTab2 are linked, so that the
// selected ID1 table oTab1 equals key ID1 in table 2
JOIN TAB {{| | {oTab2, {{"ID1", oA1: Get ("ID1")}}}} }

END BROWSE

oTab1: = {| nKey | IF (nKey = 39, (oTab2: oBrw: SetFocus ()), NEIL)}
```

```
BROWSE oTab2
     OF ooDlg
     ROW 5
     COL 270
     WIDTH 230
     HEIGHT 140
     FONT 'Arial.14'
     ALIAS oA2
     FIELD {'ID2','DATE2','STATUS'}
     HEADER {'ID','Date','Status'}
     SIZE  {40,90,40}
     PICTURE {,,'.'}
     VALID {,,}
     ADI MENU 'AD E'
     MIN EDIT 2
     BUTTON {'BAR RIGHT ALIGN BOTTOM','ADD','DEL'}

// Linking table 2 to table 3 analog is the preceding Exs in table 1
          JOIN TAB {{| | {oTab3, {{"ID1", oA2: Get ("ID1")}, {"ID2", oA2: Get ("ID2")}}}} }

// If Table 2 is pressed K_RIGHT, it focuses on table 3, and if the K_LEFT is pressed, it focuses
// on table 1.
          ON KEY {{' K_RIGHT ', {| | oTab3: oBrw: SetFocus ()}},;
                         {' K_LEFT ', {| | oTab1: oBrw: SetFocus ()}}}

// Off horizontal Skrol Tables
THE HSCROLL. F.

END BROWSE

// oTab2: = {| nKey | IF (nKey = 39, (oTab3: oBrw: SetFocus ()), NEIL),
// IF (nKey = 37, (oTab1: oBrw: SetFocus ()), NEIL)}

BROWSE oTab3
     OF ooDlg
     ROW 160
     COL 5
     WIDTH 495
     HEIGHT 160
     FONT 'Arial.14'
     ALIAS oA3
     FIELD {'ID3','ACCOUNT','DOC','DATE3','DEBIT','CREDIT','DESCRIPTION'}
     HEADER {'ID','Account','Document','Date','Debit','Credit','Description'}
     SIZE  {40,200,140,100,130,130,100}
     TRANS {'str(ID3,4)',,,,'str_z(DEBIT)','str_z(CREDIT)',}
     PICTURE {'IF(Empty(ACCOUNT),"####",".")',,}
     VALID {,,}
     LINK {,{'COA','COA.CMD'},,,,,}
     ADI MENU 'AD E'
     MIN EDIT 2
     BUTTON {'BAR RIGHT ALIGN BOTTOM','ADD','DEL','INS',,'PRINT:TAB3'}
     HSCROLL .F.
```

// ADI MENU parameter adds options on the menu when the right mouse button is clicked
// In addition to these options, users can also add
// In this case, the "Calculate Total" option, which when the T-key calls the file
// TAB3_TOT. CMD A which calculates the amount owed and subsided and displays on the screen
USER MENU {{' Calculate total ', {| | Call (' TAB3_TOT.CMD ', oA3)}}}



Clicking the right mouse button from the position of the selected table row (TAB3) to form a menu. The ADI MENU declaration defines options for adding a row (Append), deleting a row (Delete), followed by a delimiter, then editing the columns (Edit)
In the end, another menu that was specified through the USER MENU parameter was added, and this option is "Calculaate Total". For USER MENU options, the user determines which function is performed by choosing the option. In our case, a file TAB3_TOT.CMD will be called with the oA3 parameter.

File: TAB3_TOT. Cmd

```
PARAM oA
  _nPos := oA:Recno()
  nD := nC := 0
  oA:Gotop()
  DO WHILE !oA:Eof()
     nD += oA:Get('DEBIT')
     nC += oA:Get('CREDIT')
     oA:Skip()
  ENDDO
  oA:Goto(_nPos)
  Info ("Debit: "+str_z(nD),"Credit: "+str_z(nC),"Balance: "+str_z(nD-nC))
RETURN NIL
```

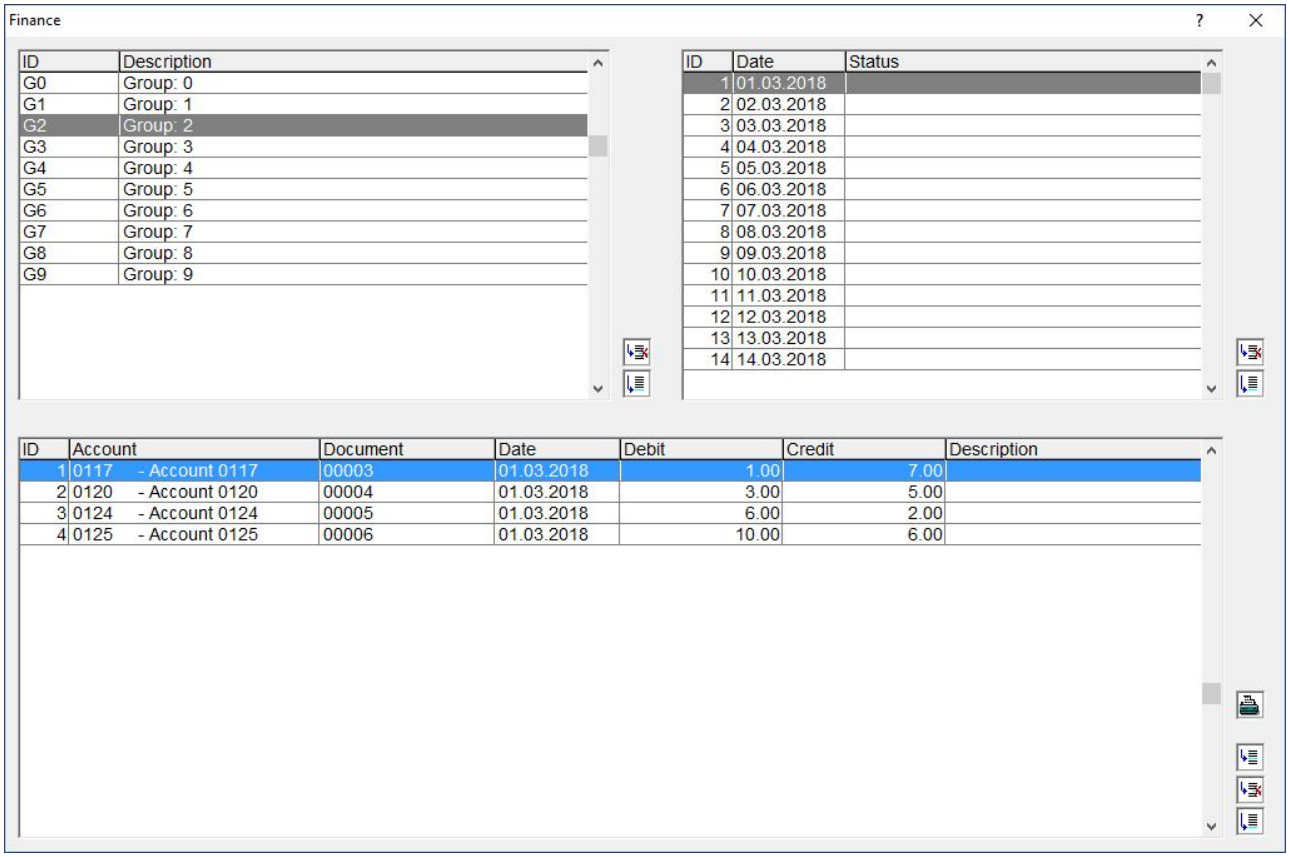
// Keystrokes K_LEFT results in focus tabele_2
// Pressing T will ring TAB3_TOT.CMD file the same as explained in the previous
// When this file is called from a menu.
ON KEY {{'K_LEFT', {|| oTab2:oBrw:SetFocus() } },;
{'Tt',{|| Call('TAB3_TOT.CMD',oA3) }} }

END BROWSE

// Show dialog and tables
ooDlg: Since (,,,. T., {| |. T.}. T., {| | oTab1: Show (), oTab2: Show (), oTab3: Show ()},,,. T.)

RETURN NIL

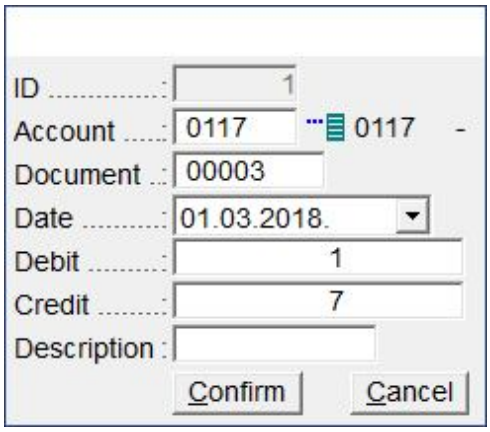
Call FIN.CMD file resulterace the following:



There are three tables that have been linked in one window. Moving around the first table retreats data in another table. Navigating the second table (or data-binding) retreats the third-table data.

Data editing is enabled by ENTER (Editing columns in a table). The other on the; in is selected by the Edit menu or by pressing CTRL + ENTER

Example in table Tab3:



The Account field is specific because it is linked to the COA parameter table LINK
The right of this field shows the button:

By clicking on this button, it is called a COA that we can get to the "ENTER" to the corresponding account. On the other hand, if we type a manual account in the field, it will be checked in the COA table, and if there is no data (account), the COA table is called in order to choose another data or add it in the table.

In addition to this way, the data in the selected row is also possible tabular by pressing ENTER. If the PICTURE f selected Columns returns data. " (point) That column is overturn in editing. Move from a field to a field when you are editing (the next or previous field)
It is possible to DOWN and UP keys. In the Account column, how it is linked to the COA table
It is possible to call the COA table by F2 When you are editing this field. Also, when entering the "Account" value and by pressing ENTER there will be an automatic validation (there is a value in the COA table, and if there is no, the COA table is called for selecting another value).

In the third table Tab3 the parameter is highlighted:

BUTTON {' BAR RIGHT TARNA BOTTOM ', ' ADD ', ' DEL ',, ' PRINT: TAB3 '}

These are defined (to the right of the table) buttons to add, delete, and insert a row.
The Print button is also added to this one 

Behind the print: TAB3 (behind the colon) is a true file called in the following way:
First, the file of the TAB3.MNU is being verified (in our example exists)
Contain this file (TAB3.MNU) is:

TITLE = Choice
[PROMPT] = Report-(Form 1)
CALL = {| y, x | Call ("TAB3_1. FRP")}
[PROMPT] = Report-(Form 2)
CALL = {| y, x | Call ("TAB3_2. FRP")}
The menus are read from this file so that clicking the Print button opens the menu with options:
Report-(Form 1)
Report-(Form 2)

Selecting one of the options is TAB3_1.FRP report file and TAB3_2.FRP
If this file didn't exist, you'd be checking TAB3.CMD or TAB3. The .FRP when writing a code, we could also strictly specify PRINT: TAB3.MNU call extension.

Class:

T_XBWEB

Basic XbWEB Tool Class

Init (cRoot, cApp_Title, lCreateDirs, lAutoLoginAgain, nCP)

Default value: cRoot = cRoot
cApp_Title = ' '
lCreateDirs =. F.
lAutoLoginAgain =. F.
nCP = 1

The base class that is generated in the source code by an function:
Start_XBW ("oXbWEB", "xbWEB + + DEMO", "en",,. T.)

A DataBase object oDB is also created when you create a class:
:: oDB: = T_MySQL (): New ()
:: oDB: cServer: =:: cServer_Root
Return value: Self

Show (bStart, lLoad)

Default value: bStart = NIL
lLoad =. T.
About thewhole content of SET.INI file from folders SYS
If there's a file on the LOGO.BMP in SYS folder This picture will be displayed as a logo in the underlying window
If there's a BACKGRND.BMP in SYS folder This picture will be displayed as the background of the root window.
Menus are loaded from file MENU.INI and buttons from the file BAR.INI
Calls an old CMD file if specified:: Call (:: CMD_Start)
The basic window of the program is being invoed.
Upon exiting it, a method is called:: Logout ()

Return value: NIL

Bye (lExit)

If there is a file defined in the variable:: CMD_End The call for this CMD file will be performed follows shutdown (closing) programs

Return value: ::lExit

Font (cFont, nCP)//FontName. SIZE ITALIC BOLD

Default value: nCP = :: nCP
// 1 - Default
// 186 - Baltic
// 161 - Greek
// 0 - Latin
// 255 - OEM
// 204 - Rusian
// 2 - Symbol
// 162 - Turkish
Returns the font of an object. The font name and size are listed in the cFont parameter

Example::: Font ("Arial. 14")
Return value::: oFont | NIL (if no font exists, the NIL value is returned)

Login (lAgain, aAuto, lAutoLogin)
Default value: lAgain =..F.
lAutoLogin =. F.
Calling dialog box to log on to server
Calling method: :: SrvLogin
SetHelpFil (' KIP. HLP ')
Return value: lOK

Login_Key (cFile, cPass, cLink, cName, cID, cType)
Reserved
Returns a series of values read from. KEY file
Return value: uRet

SrvLogin (cIP, cName, cPass, cLang)
Server login
cIP can be a guide to a local disk or link (for Example:http://Www.xbweb.rs/demo)
cName and cPass (username and password of the user whologs on to the server)
cLang: C2- language designation (en-English, SR-Serbian , etc .) Default: en
Return value: lRet

If the value to be wrought:
1. .F. then the variables:: oDB: cErr contains an error message
2. . T. There are some value of the object stored in the following ::oDB in Variables:
::nID - Number of users
::cFullName - Full name of user (logged on server)
::clocation - User location (logged on server)
::cIP - IP address of the user
::clang - Language label C2
::nlevel - Level
::clevelname - Level name (from the list of the welled levels on the server)
::lreadonly - If it is .T. then readonly mode, otherwise full access read/write

db_Open (cDB, lBY, lTestStruct, lReopen)
Default value: Lteststruct =..F.
lReopen =. F.
A database opens. This also checks the structure if the lTestStruc parameter = .T.
Return value: Lret

Logout ()
Log off the server.CallsF-Ju: ::Odb: Logout()
Return value: lRet

Load_Menu (lLoad)
Default value: lLoad =. T.
In theentire standard menu and user menus from file SYS/MENU.INI if the variable lLoad = .T.
Return value: Self

Load_Bar ()

Loads the buttons from the file SYS/BAR.INI

Return value: NIL

Var_Get (cFile, cPath, lSetvar)

Loads the declarations is PUBLIC variables from cFile

Default value: cFile = "GLOBAL. VAR "

lSetvar =. T.

Return value: aRet

Var_Set (Cfile, cPath)

Recorded (declared in. VAR file) PUBLIC variables in the file

Default value: cFile = "GLOBAL. VAR

Return value::: oDB: File_Save (cFile, aData)

INI_Get // O Theentire file of the file CODEBOOK.INI from SYS folders

Defines the variabl ::akeytables , which is a series of tables that are coders

Content ::aKeyTables : = { { Elements_Key_Tables },.. }

Each array element is a series of content:

1. Alias Name
2. Key filed Name (:: cKey_Field)
3. Table Name
4. aOrder
5. ::lSif_RAlgin (If cField is character -> right alligment (equivalent numeric))
6. cPicture
7. Call bCode of Fn
8. For Combo Create List
9. cExpr return Value: value for F2 call
10. Init order for Create combo sheet
11. bCode for Edit line selected in combo box
12. Transform = VEdit for combo box

Return value: NIL

INI_Set // Record in a file array of tables represented by Shifters

Variabla ::aKeyTables

Return value: NIL

About // displays the program information calling an option from the root menu.

If the CMD file name is specified in the variable:: CMD_About that CMD file will be called usually file ABOUT.CMD has content:

```
Info ("About Progrramu..",;
      "Author:... ")
```

Return value: NIL

Choice_Comp // Reserved

Return value: NIL

Get_Msg (nID, cLang, lAll, lWEB)

Get_Msg () - Returns a series of possible languages-read from a file LANG.INI
Get_Msg (cLang) - Returns a series of messages from the unwritten language of cLang
GET_MSG (cLang, nID) - Returns the message ID of the selected cLang language, nID message
Default value: lAll =..F.
lWEB =. F.
Return value: Char

Get_BY (lSave)

The dialup view to Select a business year
Default value: lSave =. T.
Return value: lOK

is_Shift

Returns a status if the SHIFT key is pressed
Return value: Logical

is_CTRL

Returns a status if CTRL is pressed
Return value: Logical

XCall (Parm1 [, Parm2 [..,Parm_n]]

Performsan function from the specified first parameter
Example: xcall ('Info ', ' current date:', date())
The result is view of message: current date:
9.5.2018
Return value: Any type

Xclass (cCmd [,param ..]) - Call spetial function. The first parameter is function name and then ollow others parameters that this f requires

cCmd can be:
cCMD = = "EVAL"
Eval (..) -Performs a block of the code that was wail as a second parameter
Example: XClass ("EVAL", {| | Info ("Hi!")})
Or
Xclass ("EVAL", {| d | Info ("Date:", D)}, Date ())

cCMD = = "CBEVAL"
cbEval (..)-Calls cbEval F with the specified parameters
(See the explanation in the feature guide)
cCMD can be one of FiveWin classes or f:

TDIALOG	=>	oRet := TDialog():New(..
TWINDOW	=>	oRet := TWindow():New(..
TWBROWSE	=>	oRet := TWBrowse():New(..
TIMAGELIST	=>	oRet := TImageList():New(..
TBITMAP	=>	oRet := TBitMap():New(..
TBITMAPD	=>	oRet := TBitMap():Define(..
TBTNBMP	=>	oRet := TBtnBmp():New(..
TBUTTON	=>	oRet := TButton():New(..
TIMAGE	=>	oRet := TImage():New(..
TCHECKBOX	=>	oRet := TCheckBox():New(..
TCOMBOBOX	=>	oRet := TComboBox():New(..

TGET	=>	oRet := TGet():New(..
TDATE	=>	oRet := TDatePick():New(..
TMEMO	=>	oRet := TMultiGet():New(..
TSAY	=>	oRet := TSay():New(..
TGROUP	=>	oRet := TGroup():New(..
TLISTBOX	=>	oRet := TListBox():New(..
TRADMENU	=>	oRet := TRadMenu():New(..
TMULTIGET	=>	oRet := TMultiGet():New(..
TCCHECKBOX	=>	oRet := TCheckBox():New(..
TMSGBAR	=>	oRet := TMsgBar():New(..
TMSGITEM	=>	oRet := TMsgItem():New(..
MENU	=>	oRet := MenuBegin (..
MENUITEM	=>	oRet := MenuAddIte (..
ENDMENU	=>	oRet := MenuEnd()
TTREEVIEW	=>	oRet := TTreeView():New(..
SETFOCUS	=>	oRet := SetFocus (..

Or some of the classes or function xbweb tools:

TDLG	=>	oRet := T_Dlg():New(..
CHOICEYN	=>	oRet := ChoiceYN (..
TSAYA	=>	oRet := _SayA(
T_WTAB	=>	oRet := T_WTAB():New(..
T_MSG	=>	oRet := T_MSG():New(..
T_DBF	=>	oRet := T_DBF():New(..
T_INI	=>	oRet := T_INI():New(..
T_INIV	=>	oRet := T_INIV():New(..
TFOLDER	=>	oRet := Tfolder():New(..
INFO	=>	Info (..)

Byusing The XClass method, you can from your application (or. DLL) Use the specified F-is or class

Call_Tab (CTab_Link, oWnd, cSeek, p1, p2, p3, P4, p5)

Calls a tabledefined as a sneezeon a table

cTab_Link canbe:

Block code:	Return	Value: Eval (CTab_Link, oWnd, cSeek, p1, p2, p3, P4, p5)
File. CMD	Return	value::: Call (CTab_Link, oWnd, cSeek, p1, p2, p3, P4, p5)
File. DLL	Loads the DLL file and calls a function with the same name DLL file	

Return value: & (cFn) (oWnd, cSeek, p1, p2, p3, P4, p5)

Call (cParm)

Calls a file execution with extension:

The cParm may have extensions:

.CMD - invoked program code from filea CMD

.MNU - calls to me from filea MNU

.FRP - calls for the (report) file .FRp

.DLL - is checked whether it exists on the server in folder ..\DLL same DLLfile
If there is adownload of it from the server to the local folder DLL
(download) for the next call of the same file to be called from the locale
(without redownload)

Exclude if the file has been changed on the server and will then occur
New Downloads. Then it is the same name (as the name of the DLL file)
This means that you must have an F-name in your DLL file.

Return value: xVal

Test_Sif (cTable, cFld, lEmpty, lF2, lMsg, oDlg, Oget, lNoTest)

Default value: lEmpty =..F.
lMsg =. F.
oDlg = GetFocus ()
lF2 = (:: nLastKey = = 113)
lNoTest =. F.

Tests the validity of the cFld field in the cTable table

Return value: lRet

Test_File (cFile, lUpload, lAlways_Dl)

Default value: Lupload =..F.
lAlways_Dl =. T.

It checks the existence of a file.
Default file folders with specific extensions

Folder	Extension
/DLL	DLL
/HLP	PDF, HLP, TXT
/APL	CMD, MNU, FRP, DLG, TBL, VAR
/SYS	ARR, INI, BMP

If the file exists on a local folder and does not exist on WEB folder and the lupload parameter .T. they're going to make the Upload file

```
// Upload to Server
&_oWEB_: oDB: Upload (cPath_L + cFile, cPath_S + cFile)

// Download from server
&_oWEB_: oDB: Download (cPath_S + cFile, cPath_L + cFile)
```

Return value: {File (cPath_L + cFile). Or. lExist, cPath_L + cFile, cPath_S + cFile, lLocal}
Regarding: { logical_Exist_file, Full_name_local, Full_name_WEB_server, is_local }

SendMail (cTo,cSubject,cMsg,aAtach) **- reserved**

Return value::: oDB: SendMail (cTo, cSubject, cMsg, aAtach)

Editor (cText , cTitle)

Calls the dialog box to edit cText variable
This application is allowedto be found in XBWEDIT.DLL file. If this file does not exist on Local disk will be downloaded from the server from the folders DLL

Return value: cData

Server_Date ()

Reads the date from the server
Return value: cDate (Format: MM.DD.YYYY)

Server_Time

Talk about time from server
Return value: cTime (Format: HH:MM:SS)

Edit_File (cFile)

Invoked method :: Editor of the method for editing the contents of the file with the extension: .CMD , .FRP or .MNU
Return value: .T.

Edith (cVar)

Invokes the dialog with editing the MEMO field of the cVar variable
Return value: NIL

Setup

Calls a special program for important application setup.
Only the Supervisor user can access this option.
An XBWSETUP program is being called. DLL. If this DLL file does not exist on the local disk, it will be downloaded from the server in folders DLL
Return value: NIL

Export (Oalias)

Calling EXPORT.CMD file to export data from specified alias
Calling: Call ("EXPORT.CMD ", Self, oAlias)
Return value: NIL

Import (Oalias)

He's calling IMPORT.CMD file that is being supplied by data in the specified alias
Calling: Call ("IMPORT.CMD ", Self, oAlias)
Return value: NIL

T_DBASE

A basic class that manages the MySQL or ACCESS database

Init (cServer, cVer_Access)

Defines the class
cServer can be: MySQL or ACCESS
If ACCESS is specified , then the value of the variable cVer_Access that can be ".mdb" or ".dbacc" in dependencies of version Access Bases 4.0 or 12.0
The standard value is ".mdb"
Return value: Self

Login (cServer, cUser, cPass, _ cLang)

Server Login
Default value: cServer =:: cServer
cUser - User name
cPass - Password
cLang - Selected language (scraper: sr, en,..)
Return value: lRet

The Logout

Log Off server
Return value: lRet

File_Load (cFile, lArr, l_INI)

Teaches a file and returns a value as text or an array of lines
Default value: lArr =..F.
l_INI =. F.
If it's lArr =. T. then the lines of the read file are returned in sequence
If l_INI is=. T. then the file reads like the INI file
(each line has two parameters VAR=VALUE)
Return value: cText | aRray

File_Save (cFile, cData, cPermision)

Records data in the file
Return value: lRet

File_Read (cFile, nOffset, nLen)

Loads the data from the starting point of noffset 's position in Douini nlen
Return value: cText

File_Write (cFile, cData, cPermision)

Write data to file (append to end of file)
Return value: c

Download (cFile_Server, cFile_Local, lShow, cTitle, oParent)

Downloads the file from the server to the local disk
Default value: lShow =.T. // show process in the dialog
cTitle = ' Download ' // dialog name
oParent = oFocus //
Return value: lRet

Upload (cFile_Local, cFile_Server, lShow, cTitle, oParent, cPermision)

Transfers the file from the local disk to the WEB server
Default value: lShow = .T. // show process in the dialog

cTitle = ' Upload' // dialog name
oParent = oFocus
Return value: lRet

File_Del (cFile)
Delete file
Return value: lRet

File_Exist (cFile)
Check exist for the specified file
Return value: lRet

File_Size (cFile)
Return the size of file
Return value: nSize

Dir_Make (cFile)
Creates a folder
Return value: lRet

Dir_Del (cDir)
Delete folder
Return value: lRet

Dir_Exist (cDir) //checks The existence of folders
Return value: lRet

Dir_List (cDir, lDir, lFull)
File Listing or subfolders on the current folder
Default value: cDir = "./" // Current directory
// ./APL - Sub Folder
lDir = "F" // "D" - Only folders
// "F" - Only files
// "A" - All
lFull =. F. // .T. - View: {{FileName, Size, Date_time},..
// .F. - View: {FileName,..
Return value: aRet

db_List (cDB)
Database list
Return value: aRet

db_Exist (cDB)
Check the existence of the database that is named in variable cDB
Return value: lRet

db_Create (cDB)
Creates a database depending on the data source
The data source is located in the variable :: cSrv_type, and may be "MySQL-WEB" or ACCESS
If ACCESS is checked for aversion of ACCESS file that is. accdb or .mdb and it is designated invariables:: cSrv_Version that can be "12.0" or "4.0" (4.0 default)
Return value: lRet

db_Drop (cDB)
Brisling the database
Return value: lRet

db_Select (cDB_Sel)
Select database
Return value: ::cDB_Sel

db_Backup (cDB, cFile)
How to create backup data in a file on a local disk
Default value: cDB = cDB_Sel
cFile = cDB
Return value: lRet

db_Restore (cDB, cFile)
Restoring data from a local disk
Default value: cDB = cDB_Sel
cFile = cDB
Return value: lRet

db_OpenIni (cINI_File, lTest, cMode)
Opens a databasethatcontains definitions of table structure and
IndexcEva
Default value: cINI_File = ' DBSTRUCT. INI '
cMode = ' '
lTest = .T.
If it is lTest = .T. check the structure of tables in a database relative to the content
It cINI_File. If there is a difference, you modify the tables
Return value: aTbl
Returns a series of data related to each table individually:
{{cTable_Name, aIndex, aStructure, lArr },..}

Tab_List (cDB_Sel)
List of tables that exist in selected database
Return value: Aarray

Tab_Exist (cTab)
Check The table of data in a selected database
Return value: lRet

Tab_Create (cTable, aStruc, aIndex)
Creates a table cTable
aStructoattach a series of data to each field:
1. FieldName
2. Type
3. Width
4. Decimal
5. lNull (. T. value canbe null)
6. lPrimaryKey (. T. field is the primary key)
7. Default Value
8. lAuto. T. field is a car increment)

The Type parameter can be:

- L - Logical
- N - Numeric
- D - Date
- C - Char
- R - Real
- M - Memo

Return value: lRet

Tab_Drop (cTable) // Delete table
Return value: lRet

Tab_Repair (cTable) // Table reparation
Return value: lRet

Index_List (cTable) // List of the index keys of the selected table
Return value: aRet//{{Index_Name, Index_Key, l_Unique},..}

Index_Create (cTable, aIndex, lReindex) // Creates an index table
Default value: lReindex =. F.
Return value: lRet

Index_Drop (cTable, cIndex) // Delete an index kchand a table
Return value: lRet

Get_Struc (cTable) // Return series of table Structures
Return value: aRet
A series of Sadrand for each row of a table series of values:

1- Name
2- XBase_Type
3- MySQL_Type
4- Width
5- Decimal
6- Null value
7- Key: _ | PRI | UNA | MUL
8- Default value
9- L_Auto_increment

Test_Struc (cTable, aStr)
Check the structure of the selected table and field on the existing database tables
If the structure is different from the set that is modified by the table.
Return value: lRet

Tab_Modify (cTable, aStr, lAdd) // Table Modification
Default value: lAdd =. F.
Return value: lRet

Alias_Add (cAlias, cTable, aStruc, acOrder, lZap, lShared)
Opens a work area that can be: Array, DBF file, MySQL, Access
Depending on:
a) cTable < > NIL
1. cTable is array or file name with extension . ARR . INI file
oAlias := T_ARRAY (): New (Self, cTable, cAlias, aStruc, acOrder, lZap, lShared)
2. in the otherwise, dbf files are declared
oAlias:= T_DBF (): New (Self, cTable, cAlias, aStruc, acOrder, lZap, lShared)
b) cTable = NIL
If the server MySQL-WEB = > oalias := T_MYSQL (): New (Self, cAlias)
If ACCESS = > oalias := T_ADO (): New (Self, cAlias)
Return value: oAlias

Alias_Close (cAlias) // Close the work area
Return value: NIL

isAlias (cAlias) // Check oppened the work area
Return value: lRet

oAlias (cAlias) // Return is the object of thespecified work area
If the specified work area does not exist, there is no worth of it returning is NIL
Return value: oAlias | Neal

aLastTime (cTable)
Return value: aRet //{Last_Date_of_Change, Current_System_Date}

isSif_Tab (cTab_Link, xVar, cMode, cType)
Return value: xRet
xRet is depending on:
1. cMode = = NIL // Returns Logical
2. ValType (cMode) = ' L ' // Return NIL If there is/cMsg message that the code does not
// exist
3. ValType (cMode) = ' C ' // Returns the field value if there is a code

Query (Cquery)
VRAand the value of a specified query
Return value: xRet

Query_CMD (Cquery, lResult)
Default value: lResult =. F.
Return value: lRet
Search (cTable, acWhere, cFLD)
Vraća result (fieldvalue) for the specified condition in the specified table
Return value: xRet

SendMail (CTO, Csubject, Cmsg, Aatach) - **reserved**
Return value: lRet

db_SaveIni (CINI_File, aData)
Save table structure to file
Default value: cINI_File = ' DBSTRUCT. INI'
Return value: NIL

T_MySQL

Opens a work oblas mysql data source from a WEB Server

Init (oDB, cTable)
oDB - standard is oxbweb: oDB
cTable - table name in database MySQL
Return value: Self

Close // Closes the work area
Return value: NIL

Query (cSql,; // SQL Query
cTable ,; // Tablesfrom theselectedBasesData
caFld,; // Fieldor A series ofthe
acWhere,; // Cconditionor A series ofconditions
acOrder,; // IndexKlječlikenametheor A series ofthe
ncOrder,; // An initialIndexKljeC as a number or name
nOffset,; // positionOffset
nLimit) //
In the cSql variable SQL expression may be specified
Example: Query ("SELECT * FROM TAB1 WHERE ID1 = ' G1 '")
or: Query (, ' TAB1',,{{' ID1 ', ' G1 '}})

If the operation succeeds in apattern variable ::cErr is empty otherwise it contains a description of the error
Return value: NIL

Recno // The position of thecurrent order
Return value: nRecno (::nOffset + ::nPos)

RecCount (lRefresh)
Returns the total number of rows in a table
Default value: lRefresh =. F.
Return value::: nReccount

Sort (ncSort, lGoTop)
Neatfies the table by index
Default value: lGoTop =. F.
ncSort: + // increment
- // decrement
cIndex name// same index name
Numeric // index order
Return value: cIndex_Name

Skip (nSkip)
Moves the row position in a table
Default value: nSkip = 1
Return value: +-nSkipper

PgUp ()
Return value: NIL

PgDn ()
Return value: NIL

GoTo (nRec)
Return value: NIL

GoTop ()
Return value: NIL

Gobottom ()
Return value: NIL

Eof ()
Return value::: lEof

Bof ()
Return value::: lBof

Get (ncVar, nRecno)
Returns the value of a field.
ncVar can be the field name or position of the field in ::aFld array
Default value: nRecno = ::nPos (current value)
Return value: xValue

Prop (ncVar)
Returns field attributes:
1. cName
2. MySQL_Type
3. Xb_Type
4. nMaxWidth (defined field length)
5. nDec
6. nWidth (read field length)
7. xDef
8. lNull

Return value: Array

SQLPrepare (cMode)
Default value: cMode = "" | RECCOUNT | MAX |
Return value: cSQL

Add (aR, lRefresh)
Add row in table
aR Array of fields and values ({{cField_Name, xvalue},..}
Return value: lRet

Del ()
Delete surrent row in table
Return value: lRet

Del_All (aWhere)
Deletes all rows in a table
Default value: aWhere =:: aWhere
Return value: lRet

Ins (nPos)
Inserts a row in a table if the key field is a numeric
Return value: lRet

Road (ncVar, xVal)
Sets the value of a field in a table
Return value: NIL

Flush (xSif)
Physically writes the value of the field ::Put method
The value of a key xSif field (if the key field is not specified is ::cKey_Field)
Return value: lRet

Repl (ncVar, xVal)
This method is called ::Put and then ::Flush
Return value: NIL

aEval (aFld)
Return value: aR

Expr (cExpr, nRecno)
1._Cexpr canbe the name of the prefixed field #
Example: ::Expr ("#NAME") -> The value of the NAME field
2. _cexpr can be a block code:
Example: ::Expr ({| | Name }) -> The value of the NAME field
3. _cexpr is a character:
Example: ::Expr ("ABC ()") -> call to function ABC
Return Value: _xRet

Seek (cSeek, lSoft, [reserve] , [cFld])
Default value: lSoft =. F.
cFld – Key field name
Return value: lRet

Refresh
Return value: NIL

Search (cSeek, cbRet)
Find a value and return The result from the cbRet
Return value: xRet

TimeQuery
Return value: lRet

Rlock (anRec)
Lock of the record (virtual lock)
anRec - can be a number of styles or a number of locked locks
The default value is ::Recno()
Return value: lRet | aRet – A set of logical values if anRec is a string

Note: If the user did not unlock the style, it will be unlocked after the expiration date
TimeOut variables are defined in the xset.php file
Locking of the record is regulated exclusively for data editing
(this is applied to W_TAB) and does not apply to data entry using ::Get metode

Unlock (anRec)

UnLock of the record (virtual unlock)
anRec - can be a number of styles or a number of locked locks
The default value is ::Recno()
Return value: lRet | aRet – A set of logical values if anRec is a string

IsRlock (anRec)

Check if the record is locked
anRec - can be a number of styles or a number of locked locks
The default value is ::Recno()
Return value: lRet | aRet – A set of logical values if anRec is a string

T_ADO

Working with an Access database
Explanations for certain methods are analog class T_MySQL

Init (oDB, cTable)
Return value: Self

Close
Return value: NIL

BeginTrans
Return value: NIL

EndTrans.
Return value: NIL

RollbackTrans
Return value: NIL

Filter (cKey)
Default value: cKey = ' '
Return value: NIL

Query (cSql,;
cTable ,;
caFld,;
acWhere,;
acOrder,;
ncOrder,;
nOffset,;
nLimit)
Return value: NIL

Recno
Return value: ::oRS: AbsolutePosition

RecCount
Return value::: oRS: RecordCount

Sort (nStep)
Return value: NIL

PgUp ()
Return value: NIL

PgDn ()
Return value: NIL

GoTop ()
Return value: NIL
GoBottom ()
Return value: NIL

Goto (nRec)
Return value: NIL

Skip (nRec)
Default value: nRec = 1
Return value: (:: Recno ()-_ R)

Eof ()
Return value::: lEof//:: oRs: Eof

Bof ()
Return value::: lBof//:: oRs: Bof

Get (ncVar, nPos)
Return value: xValue

Put (ncVar, xVal, lUpdate)
Default value: lUpdate =. T.
Return value: NIL

Repl (ncVar, xVal)
This method is called ::Put and then ::Flush
Return value: NIL

Prop (ncVar)
Return value: A

SQLPrepare (cMode)
Default value: cMode = ' '
Return value: cSQL

Add (aR, lRefresh)
Return value: lRet

Del (r)
Return value: {:: Reccount (),:: Recno ()}

Del_All (aWhere)
Return value:. F.

Ins (_r, aRec)
Return value:. T.

Flush (xSif) // value in ::cKey_Field at addNewRec
Return value: NIL

aEval (aFld)
Return value: NIL

Expr (cExpr, nRecno)
Return Value: xRet

Seek (cKey, lFirst, lSoft,[cFld])
Default value: lFirst =. T.
cFld – Key field name
Return value: lRet

Search (cSeek, cRet, cIdx)
Return value: xRet

Refresh
Return value: NIL

TimeQuery
Return value: lRet

Rlock (anRec) (this method is described in the T_MySQL class)
Lock of the record (virtual lock)

Unlock (anRec) (this method is described in the T_MySQL class)
UnLock of the record (virtual unlock)

IsRlock (anRec) (this method is described in the T_MySQL class)
Check if the record is locked

T_DBF

Working with a DBF file. Explanations for certain methods are analog class T_MySQL

Init (oDB, cTable, cAlias, aStruc, acOrder, lZap, lShared)
Default value: lZap = .F.
lShared = .F.
oDB = :oDB
Return value: Self

Close
Return value: .T.

Recno
Return value: Recno()

Reccount (lRefresh)
Return value: Reccount()

Sort (ncSort)
Return value: ::aOrder [::nOrder]

Skip (nSkip)
Return value: dbSkip (nSkip)

PgUp ()
Return value: NIL

PgDn ()
Return value: NIL

GoTo (nRec)
Return value: NIL

GoTop ()
Return value: NIL

GoBottom ()
Return value: NIL

Eof ()
Return value: Eof()

Bof ()
Return value: Bof()

Get (ncVar, nRecno)
Return value: x

Prop (ncVar)
Return value: a

Add (aR)
Return value: .T.

Del ()
Return value: lRet

Ins ()
Return value: lRet

Put (ncVar, xVal)
Return value: NIL

Repl (ncVar, xVal)
This method is called ::Put and then ::Flush
Return value: NIL

Eval (aFld)
Return value: aR

Expr (_cExpr,nRecno)
Return value: _xRet

Seek (cSeek,lSoft, [reserve] , [cFld])
Default value: lSoft = .F.
cFld – Key field name
Return value: lRet

Refresh (ncOrder)
Return value: NIL

Reindex
Return value: NIL

Search (cSeek,cbRet)
Return value: xRet

Rlock (anRec)
Lock the record (this method is described in the T_MySQL class)

Unlock (anRec)
UnLock the record (this method is described in the T_MySQL class)

IsRlock (anRec)
Check locked the record (this method is described in the T_MySQL class)

T_ARRAY

Work with a table that is located in a file or a series of data
Explanations for certain methods are analog class T_MySQL

Init (oDB, _cTable, cAlias, aStruc, acOrder, lZap)
Default value: lZap = .F.
oDB = :oDB
Return value: Self

Close
Return value: NIL

Flush
Return value: NIL

Recno
Return value: ::nPos

Reccount
Return value: ::nReccount

Sort (ncSort)
Return value: ::aOrder [::nOrder]

Skip (nSkip)
Default value: nSkip = 1
Return value: ::nPos - oPos

GoTo (nRec)
Return value: NIL

GoTop ()
Return value: NIL

GoBottom ()
Return value: NIL

Eof ()
Return value: ::lEof

Bof ()
Return value: ::lBof

Get (ncVar, nRecno)
Default value: nRecno = ::nPos
Return value: x

Prop (ncVar)
Return value: a

Add (aR , lRefresh)
Return value: .T.

Del ()
Return value: .T.

Del_All
Return value: .T.

Ins (nPos)
Default value: nPos = ::nPos
Return value: .T.

Put (ncVar, xVal)
Return value: NIL

Repl (ncVar, xVal)
This method is called ::Put and then ::Flush
Return value: NIL

aEval (aFld)
Return value: aR [1]

Expr (_cExpr,nRecno)
Return value: _xRet

Seek (cSeek,lSoft, [reserve], [cFld])
Default value: lSoft = .F.
cFld – Key field name
Return value: lRet

Refresh
Return value: NIL

TimeQuery
Return value: lRet

Search (cSeek,cbRet)
Return value: xRet

Rlock (anRec)
Lock the record (this method is described in the T_MySQL class)

Unlock (anRec)
UnLock the record (this method is described in the T_MySQL class)

IsRlock (anRec)
Check locked the record (this method is described in the T_MySQL class)

T_WTAB

Editing and reviewing any work area

```
Init   ( nY           ,;
         nX           ,;
         nW           ,;
         nH           ,;
         oDlg         ,; // cTitle | { oDialog, cTitle } | NIL - same dialog
         ocFont       ,; // Font_Name or oFont
         oAlias       ,;
         aFld         ,; // array of Field_Name
         aTitle       ,; // NIL or array
         aTrn         ,; // NIL or array
         aSize        ,; // NIL or array
         aAlign       ,; // NIL or array
         aPic         ,; // NIL or array
         aVal         ,; // NIL or array
         aLink        ,; // NIL or array Links of codebook table
         aCopy        ,; // NIL or array nField_Pos or cField_Name
         aFilter      ,;
         nVTab        ,; // Begin Position of VTAB
         Vtab_Size    ,; //
         aSort        ,; // { Sort_Name, .. }
         aButton      ,;
         aKeyb        ,;
         cADI_Mnu     ,;
         lAdd         ,;
         lAdd_Ch      ,;
         lDel         ,;
         lDel_Ch      ,;
         lIns         ,;
         lIns_Ch      ,;
         nMin_Edit    ,;
         aTotal       ,;
         bEnd         ,;
         lHScroll     ,;
         cFSeek       ,;
         lReadOnly    ,;
         aTriger      ,;
         aJoin_Tab    ,;
         lResize      ,;
         aRClick_UserMenu ,;
         aPic_Fld     ,;
         aItems )
```

Note: some parameters are explained on page 41 in the function description BROWSE within the .CMD file

Default value: lHScroll = .T.
lResize = .F.
aPic_Fld = {"",0,0,.F.,,0}
oDlg = :oWnd
lAdd = .T.

lAdd_Ch = .F.
lDel = .T.
lDel_Ch = .T.
lIns = .T.
lIns_Ch = .T.
lReadOnly = .F.
nVTab = 0

Return value: Self

Show (lWait, _bStart)

Default value: lWait = .F.

Return value: xRet

Resize

Return value: NIL

Keyb (nKey, nFlags, oBtn)

Return value: nKey

KeybS (nKey, nFlags)

Return value: nKey

rClick (nY,nX)

// User Menu

Return value: NIL

MenuPopup (oBar)

Return value: NIL

Sort (ncSort)

Return value: NIL

Seek (cKey, lSoft, [cFld])

Return value: lRet

VTab (lVTab, lResize)

Default value: lResize = .T.

Return value: NIL

Change (lNull, nKey, lChng)

Default value: lNull = .F.
lChng = .T.

Return value: NIL

_Eval

Return value: NIL

VEdit

Return value: NIL

Filter

Return value: NIL

Print (oBtn)
Default value: oBtn = ::oPBtn
Call (a[2],Self,oBtn) // .MNU | .CMD | other
Return value: NIL

_End
```
IF ValType(::bEnd) = 'B'
  Eval (::bEnd,Self)
ENDIF
```
Return value: .T.

LbEval () // create PRIVATE var of ::aFld and call lbEval for each parameter
Return value: IF(Len(aRet)=1,aRet[1],aRet)

Triger (cMode, p1)
::aTriger => Array or Block
Call each ::aTriger or Block
Return value: xRet

Menu_Pic (nY,nX)
Create picture menu on position cursor
```
c := "Load from file"          //&_oWEB_:Get_Msg(_MSG_EDIT_)
c := "Load from clipboard"     //&_oWEB_:Get_Msg(_MSG_EDIT_)
c := "Save = file"             //&_oWEB_:Get_Msg(_MSG_EDIT_)
c := "Copy = clipboard"        //&_oWEB_:Get_Msg(_MSG_EDIT_)
c := "Erase picture"           //&_oWEB_:Get_Msg(_MSG_EDIT_)
c := "View = normal size "     //&_oWEB_:Get_Msg(_MSG_EDIT_)
```
Return value: NIL

Oper_Pic (cCMD)
Poziv jedne od operacija u zavisnosti od:
```
1. cCMD == "LOAD"                  // Učitava sliku iz fajla
2. cCMD == "DEL"           // Briše sliku
3. cCMD == "FROM_CLIP" .and. !EmptyClipB() // Učitava sliku iz clipboard a
4. cCMD == "TO_CLIP"       // Odlaže sliku u klipboard
5. cCMD == "SAVE"          // Snima sliku u fajll
6. cCMD == "VIEW"          // Prikaz slike
```
Return value: NIL

Display (lOnOff) Display On/Off table and corresponding buttons

Call (bStart) Prikaz tabele i vraćanje sadržaja iz polja ::F2_Ret

T_CMD

Perform source code by using a interpreted

Init (cFile)
Return value: Self

Prepare
Return value: NIL

Exec
Return value: xRet

Fn (cFn, p1, .. p30)
Return value: xRet

Get_Param (cTxt,aPar,lTestSintax)
Default value: lTestSintax = .T.
Return value: aValue

T_DLG

User dialog

Init (oParent, cTitle, ocFont)
Default value: oParent = :oWnd
oFont = :cFont_Dlg
Return value: Self

Add
Return value: NIL

Show (aPos, l_BtnOK, l_BtnCancel, nHeight, lReadOnly)
Default value: aPos = .T.
l_BtnOK = .T.
l_BtnCancel = .T.
lReadOnly = .F.
Return value: aPar

ShowResize ( )
Return value: NIL

CallF2 (o)
Return value: NIL

TestValid (oG,lF2)
Return value: l

T_INI

Working with INI files

Init (cFile, cPath, lWEB)

Default value: cPath = " // :cPath
lWEB = .F.
Return value: Self

Close (alArr)

Default value: alArr = {} // niz naziva ili dela naziva promenljive
// koja se snima kao niz u vise redova
Return value: nil

ReadLn ()

Return value: c

_ReadLn ()

Return value: (Left (c,n))

Get (cEvent , cDef , cMac , cType)

Default value: cMac = "
cType = "
Return value: (c)

Set (cEvent , cValue , cMac , lMacro)

Default value: lMacro = .F. // ako je .T. onda je cValue pod navodnicima
Return value: nil

Clear

Return value: nil

T_INIV

Working with INI file (II)

Init (cFile, cPath, lWEB, cMem)

Default value: cPath = '' // :cPath
cMem = ''
lWEB = ('/' $ (cPath+cFile))
Return value: Self

Close (lWrite)

Default value: lWrite = .T.
Return value: nil

ReadLn ()

Return value: c

_ReadLn ()

Return value: (Left (c,n))

Get (cEvent , cDef, cType, pVar)

Return value: xVal

Set (cEvent, xValue, cType, pVar)

Return value: lRet

Clear

Return value: nil

T_MSG

View messages when processing data

Init (_Col, _Row, cTitle, ocFont, nWait_Sec, _MaxLn, lElapsed, cStyle, oWnd)
Default value: _Col = 40
_Row = 3
_MaxLn = -2
ocFont = :cFont_Msg
lElapsed = .F.
cStyle = 'NORMAL'
oWnd = oWndFromhW(GetFocus()) //:oWnd
Return value: Self

Close
Return value: NIL

Test
Return value: ! ::lExit

Wait (nSec)
// nSec ili niz: { nSec, cText, nRow, nCol, cFont, cColor, lCountdown, lDec_millisec }
Default value: nSec = 1
Return value: NIL

Cls (lAll_Ln)
Default value: lAll_Ln = .F.
Return value: NIL

Say (y,x,cText,cColor,oFont,lStayCoor,cAlign)
Default value: oFont = ::oFont
lStayCoor = .T.
cAlign = 'L'
Return value: NIL

Ln (cText,cColor,oFont,lStayLn,lOriginSize)
Default value: lStayLn = .F.
Return value: NIL

Print
Return value: NIL

Elapsed (lFull, lMilliSec)
Default value: lFull = .F.
lMilliSec = .T.
Return value: cRet

isStop
Return value: NIL

Press (cTxt) //,cBtn)
Default value: cTxt = :Get_Msg(_MSG_PRESS_ENTER_)
Return value: NIL

XPR_REPORT

Report Builder

Init (cPaper, cExport, lSilent, lLand)
Default value: cPaper = 'A4'
cExport = NIL
lSilent = .F.
lLand = .F.
Return value: Self

Start (cFile)
Return value: Self

Open_FRP (cFile)
Return value: lRet

SetPaper (cPaper, nMarg_L, nMarg_R, nMarg_U, nMarg_D, nCols, nRows)
Default value: cPaper = _aPaper [::nPaper,1]
nMarg_L = 12
nMarg_R = 8
nMarg_U = 8
nMarg_D = 8
nRows = 66
nCols = 80
Return value: NIL

Prepare
Return value: Logical

Close
Return value: Logical

Create_XPr (lOnlyHeader)
Default value: lOnlyHeader = .F.
Return value: Logical

Report
Return value: Logical

Show (lPreview, lDesign)
Default value: lPreview = .T.
lDesign = .F.
Return value: NIL

FR_cName (cName)
Return value: cName

FR_Memo (acTxt, nLeft, nTop, nWidth, nHeight, cName, cBorder, _nLn)

Default value: nLeft = 0
nTop = 0
nWidth = 0
nHeight = ::nUnit_Y
cName = 'Memo#'
cAlign = 'L'
cBorder = ''

Is acTxt array: { 1. cText,
2. cAlign
3. cFont
4. cMacro
5. cColor
6. cX
7. lVert
8. cVAlign
9. lWordWrap
10. if aTxt char: %nnn je nTop = Pg_Height * %nnn
else nTop = nUnit * nnn
if aTxt numeric => nHeight = nUnit * nnn
11. cFill }

cText = acTxt ili acTxt [1]

If cText starts with a suitable prefix, a corresponding method is called:

#PIC: cFileName
#BARCODE: Code
#LINE: Text, Y: | R:, X: | C:, L:, S:
#ELLIPSE: Y: | R:, X: | C:, W:, H:, L:, S:
#BOX: Y: | R:, X: | C:, W:, H:, L:, S:

Return value: C

FR_SetText (cAlign, cVAlign, cFont, cColor, lVert, lWordWrap)

cAlign -> moze biti: L | LEFT
R | RIGHT
C | CENTER

cVAlign -> moze biti: T | TOP
B | BOTTOM
C | CENTER

If lVert =. T. Then the text appears at an angle of 90 degrees

Return value: C

FR_Font (ncFont, cColor)

If ncFont text is:
The Font is specified in [Size.] FontName [Bold] [Italic]
Standard value for font size is 12
cColor can be a numeric value
or test: BLACK, MAROON, GREEN, OLIVE, NAVY, PURPLE,
TEAL, GRAY, SILVER, RED, LIME, YELLOW, BLUE,
AQUA, WHITE, SKYBLUE, BRUSH1, BRUSH2, BRUSH3
If ncFont is nonnumeric, then the font is taken from a series of predefined fonts
Variable:: aFonts [ncFont]
The font name contains the ' $ ' sign then the font name is taken by the standard font
Return value: cString

Line (nY, nX, nY2, nX2, cPen, cStyle, cColor, cArrow, nW, nH)
Default value: Cpen = Standard line raw value
nW = 0
nH = 0
Return value: C

FR_TBox (nY,nX,aZT,nHT,nLevel,nW_Max,cNo,_cFont)
Default value: nLevel = 1
cNo = ''
Return value: nX

cGet_Param (cTxt, cCmd, cEnd, cType, cRetStr, _nLn, _0Val)
Default value: cCmd = '?'
cEnd = ','
cType = 'N'
nLn = 0
Parameters may be: C: | X: | W:-> Column | X | Width
Or: R: | Y: | H:-> Row | The Y | Height
Value can contain the character:
-% with a paper-size relationship being taken
width, or height, depending on the specified parameter.
-# then the value behind is viewed as macro * Unit Y or X
Return value: xRet

FR_PIC (acTxt, cName, nCols, _nLn)
acTxt is array: {nTop, nLeft, nWidth, nHeight, cFile_Name [, Color]}

acTxt is char: FILE:...., X:nnn, Y:nnn, W:nnn, H:nnn, COLOR: CCC
If the file name begins with the sign ' # ' is being performed by a pimp.
Example: FILE: #cFile, X:120, Y:20...
The name of the file is #cFile and in that case, the & will be called (cFile)
Return value: C

Skip (nDirection)
Return value: Logical

FR_BARCODE (acTxt, [cName], nPos, _nLn)
Is acTxt array: { Top, Left, Width, Height,
cText, cData, cType, cFont, cColor, cFill, cRotate }
Is acTxt char: TEXT: .., TYPE: .., DATA: .., NO: ..,
X: .., Y: .., W: .., H: ..
Default value: nPos = 1
cName = ::FR_cName ('BC#')
cFont = '12.Arial'
cColor = 0
cType = 'EAN13'
Return value: c

Print (cTxt, cAlign, cFont, cColor, nY, nX, nW, nH)
Default value: cFont = ::cFont
cAlign = 'L'
cColor = ::cColor
nY = ::nLn_Cnt
nX = 0
nW = '100%'
nH = 1
If the line number is:: nLn_Cnt larger than the maximum page line:: nMaxLine
The feature will be called:: Graph ("EJECT")
Return value: NIL

Graph -> Depending on the first parameter can be:
Graph (' TEXT ', cText, nLeft, nTop, nWidth, nHeight, cName, cBorder, nLine)
Graph (' LINE ', y, x, y2, x2, cPen, cStyle, cColor, cArrow)
Graph (' RECTANGLE ' | ' ROUNDRECTANGLE ' | ' TRIANGLE ' | "ELLIPSE",
Y, x, y2, x2, cPen, cStyle, cColor, cFill, cBrush, W, h)
Graph (' EJECT ' | ' NEWPAGE ' [, ' LANDSCAPE '])
Graph (' TABLE ', y, X, W, H, nCol, nRow, aHeader, lNoRows)
Graph (' PIC ', Y, X, Width, Height, cFile, Color)
Graph (' BARCODE ', Y, X, Width, Height, cText, cData, Type, Font, Color, Fill, Rotation)
Return value: xRet

FR_Ellipse (nY, nX, nW, nH, cPen, cStyle, cColor, cFColor)

Shape (cShape, nY, nX, nY2, nX2, cPen, cStyle, cColor, cFColor, cBrush, nW, nH)
Default value: cPen = ::Width_Line //'0,5'
Return value: c

Call_FRP (cFile)
Return value: lRet

NewPage (lLand)
Return value: NIL

ShowProgress (x, y, z)
Return value: NIL

ShowProcess (sObjectName)
Return value: NIL

XPR_TABLE

Table View

Init (oRep, nY, nX, nWidth, nHeight, nRows, aCols, aHeader, lNoheadLn)
Default value: lNoHeadLn = .F.
Return value: Self

Show // View table
Return value: NIL

SetCell (_y,x,xVal,cAlign,cFont,cColor,lAll,lVert,cFill)
Default value: lAll =. F.
Cets data in array cell:: aCell
_y - Row of the table (if 0 refers to the header)
_x - Colona tabel
xVal - A value that is entered in a cell (if not inserted , the value is not changed .
is predefined in cell :: aCell)
cAlign - Alignment: C | CENTER
L | LEFT
R | RIGHT
cFont - size.Font
Ccolor - color can be:
BLACK | MAROON | GREEN | OLIVE | NAVY | PURPLE
TEAL | GRAY | SILVER | RED | LIME | YELLOW
BLUE | AQUA | WHITE | SKYBLUE
BRUSH1 | BRUSH2 | BRUSH3
lAll -
Lvert - Text appears upright
cFill - Fill (brush) cells

Return value: NIL

Cell (y, x)//display a cell
Return value: NIL

SetRow (nLn,aVal,acAlign,cFont,cColor)
Default value: Cfont =:: CFont_Body
cColor =:: cColor_Body
Define all the rows of the selected lines
Return value: NIL

The FUNCTIONS:

Start_XBW (cVar, cApp_Title, cPath_Root, cLang_Def, lCreateDirs, lAutoLoginAgain, nCP)
Default value: Cvar = "oXbWEB"

This function runs at the beginning of the application and the cVar returns the base-class object

Source code Example:

```
FUNCTION MAIN
    Start_XBW ("oXbWEB", "XBWEB + + DEMO", "en",,. T.)
    IF! oXbWEB: Login ()
        Quit
    ENDIF
    oXbWEB: Show ()
RETURN NIL
```

Return value: NIL

_ Date_Time (dDate, cTime)// returns the string value of the specified date and time
in the appropriate format
Default value: dDate = Date ()
cTime = Time ()
Return value: Strzero (Year (dDate), only, for "." +;
Strzero (dDate) 2) + "." +;
Strzero (Day (dDate) 2) + "" + cTime

SQL_Prepare (cTable,_aWhere,cOrder,_aFLD) // prepare SQL Syntax
Default value: _afld = "*"

Preparing SQL string according to the specified parameters
aWhere can be specified as a block code. the return result must be a string.
Such a string is returned in the aWhere variable

An array of values in the awhereconsists of:
1. A series of two elements. The first The field inTableandothervalue
Example: aWhere: {{' You > = ',ctod(' 1.1.2018 ')}, {' DATE < = ',ctfrom(' 31.1.2018 ')},..}
Condition for the period from 1.1 to 31.1.2018
or {' DATE ',ctod(' 11.1.2018 '} condition if date equals date 11.1.2018

2. String values. Example: { "NAME like ' A% '",..}

Return value: cSQL

Example 1: SQL_Prepare ('TAB2 ', {{' ID1 ', ' G1 '}}, ' ID2 ')
Rea string: SELECT * FROM TAB2 WHERE ID1 = ' G1 ' order BY ID2

Example 2: SQL_Prepare ('TAB2 ', {{' ID1 ', ' G1 '}}, ' ID2 ', {' ID2 ', ' DATE2 '})
Willowa STRING: SELECT ID2, DATE2 FROM TAB2 WHERE ID1 = ' G1 ' order BY ID2

Srch_Fld (cTable,aWhere,cbRet)

Invokes the query above the table for the appropriate condition and returns the specified value, or NIL

Parameter: Cbrett can be a string (thename of a field whosevalue is " re") or block code

Example: srch_fld (' COA ', {{' ACCOUNT ', ' 1001 '}}, ' DESCRIPTION ')

If there is a result for the ACCOUNT condition = ' 1001 ' in the COA table, the result is returned from the DESCRIPTION

Return value: xRet

CreateShortCut (cDestDir,; // Destination folder to create. lnk
cSC_Name ,; // Shortcut Name
cTargeted File,; // Shortcut Taget full file name
cTargeted Dir,; // Shortcut working folder
cTargeted Descr,; // Shortcut description
cTarget Arg) // Shortcut arguments

Return value: NIL

isShortCut (cDestDir,; // Destination folder to create. lnk
cSC_Name ,; // Shortcut Name
lErase)

Return value: File (sPath + "\" + cSC_Name)

Possible values of DestDir: Allusersdesktop
Allusersstartman
AllUsersPrograms
AllUsersStartup
Desktop
Favorites
Fonts
MyDocuments
NetHood
PrintHood
Programs
Recent
The Sands
Starting Menu
Startup
Templates

lFile (cFile) // returns the logic of the existence of the file. According to whether a local
// computer is selected
// or the Web is evaluated for the existence of the file on the local PC or web

Return value: &_oWEB_: Test_File (cFile)

Upload_File (cFile) // file transfer to WEB server

Return value: &_oWEB_:Test_File (cFile,. T.)

Call_MNU (cFile, o, oBtn, cFont, nInit) // menu Call

The cFile parameter may be a file name (extension . INI) which houses the menus.

Example of the Dr . INI file

1= {' Example 1 ', ' Example1. cmd ',110, ' '}

2= {' Example 2 ', ' Example2. cmd ',0, ' '}

You are generating me in two rows and selecting one of them is called the call function

With corresponding values

Heknowsand if option 2 is chosen by "Example 2 " will be called CALL ("Example2. cmd")

Thereis an execution . CMD file.

In addition to this , the cfile parameter canbe a series of menu names

Example: {' Example 1 ', ' Example 2 '}

In that case function Call_Mnu reand the serial number of the selected menu or 0 (Zero) ifis notchoosesManny (ESC key)

If a paramemetard is an object Button then the gen is at the point of the button

Parameter: cFont - style velizina font. Font name

Example: Arial. 14

nInit -is the initial line value of the menu

Return value: v

Bkgr_SQL (cAlias, cSQL)

oThread: Start ("_Call_Query", oThread, cAlias, cSQL)

Return value: NIL

_ Call_Query (oThread, cAlias, cSQL)

o:Query (cSQL)

Return value: NIL

Bkgr_Finish (cAlias)

Return Value: oAlias | Neal

Bkgr_Del (cAlias)

Return value: NIL

Int_SetOption (A, B, C, D) /// internet Setoptiona function

Return value: DllCall ("WinInet. dll", DLL_STDCALL, "Internet Setoptiona", A, B, C, D)

setURLTimeOut (nMin)

Return value: NIL

ChngF_CP (oFont, nCP) // change the code side corresponds to the font

Return value: NIL

_SF_ (cKey, cA)

IF Empty (cKey)

Return value: ' '

ENDIF

Return value: (&_oWEB_): oDB: isSif_Tab (cA, cKey, "", "C")

// 3-th parameter "" means an automatic field suggestion that returns from aKeyTables

GetFld_Sif (cTable, cFld, cbRet, cPreTxt, cRet0_Txt)

The function returns a value from the corresponding codebook.
Typically, a table (cTable) that is specified as a code list is searched for key field (cFld) and returns the corresponding value (cbRet), which can be the name of the array in the requested table or block code. The cPre_Txt fields are specified as text before the returned value, that is, the cRet0_Txt field is a predefined value unless f finds a key field.
Default value: cPreTxt = ''
cRet0_Txt = ''
Return value: xRet

Call_Msg (cFn, cMsg) // This function invoked cFn when it first opens the dialog with the // message cMsg and eventually closes the dialog.

Example: Call_Msg ('account() ', ' Please wait ..')
Opens dialog ' wait .. ' and thecallsF-IAccount().
When you do and the called function dialog is closed.
Default value: cMsg = &_oWEB_: Get_Msg (_ MSG_LOOP_ESC)
Return value: xRet

Call (cFn [, xParm1, [xPArm2 ..]])

This function performs a file that corresponds to an extension:

DLL It loads the specified DLL file and calls the same name function. DLL file if it does not exist on the local folder ..\DLL will be dragged from the WEB (from the same folder). Too if the DLL file changes to the WEB folder, it will come first to the dowload of the same file. Then, the DLL file is loaded and calls the same name function with parameter indication.
Example: CALL ("CALC.DLL", 5,6)
Loads CALC.DLL file and calls f: CALC (5,6)
A typical example of DLL files are XBWEDIT.DLL and XBWSETUP.DLL which are editor and setup programs that are part of the XbWEB tool.

CMD Loading and executing CMD file with parameter transfer
Example: the content of the CALC.CMD file is:

```
PARAM A, b
Z:= a + b
Info (z)
RETURN NIL
```

If called ::Call("CALC. CMD ",5,6)
Perform the code above and display (Info - function) value of parameter sum a and b in the case 5 + 6

FRP call to .FRP file to generate the report, function: Call_FRP

MNU menu call, function: Call_MNU

Return value: xRet

x2SQL (xVar) // Convert any type = SQL String type
Return value: cRet

d2SQLc (_Dat) // Convert date = SQL present
Return value: cRet

cNullType (cType, nWidth, nDec) // return value: null value any type how string
Default value: nDec = 0 // If type is numeric
Return value: cVal

NullType (cType, nDec, nW) // return value: null value any type
Default value: nDec = 0 // If type is numeric
Return value: xVal

_Download (cFile_Server, cFile_Local, oDB, oSay, oDlg, bTest, oMeter, lEnd)
Download file from folder WEB server to local disk
Default value: ODB = &_oWEB_: ODB // source database

Primer: MsgMeter ({|oM,oSay,oDlg,lEnd| lRet := _Download (cFile_Server,
cFile_Local, oSelf, oSay, oDlg,,oM,@lEnd) },,cTitle)
Return value: lRet

_Upload (cFile_Local, cFile_Server, oDB, oSay, oDlg, bTest, cPermision, oMeter, lEnd)
Upload a local file to a WEB folder

Primer: MsgMeter ({|oM,oSay,oDlg,lEnd| lRet := _Upload (cFile_Local,
cFile_Server, oSelf, oSay, oDlg,,cPermision,oM,@lEnd) },,cTitle)
Return value: lRet

AES_SetKey (cKey) // setting AES encryption key
Return value: NIL

AES_Crypt (CBuff, CKey) // encryption of the Sadrin cBuff variable
Default value: cKey = __ckey
Return value: oAes: Encrypt (cBuffer)

AES_DECR (cBuff, cKey) // decrptizing of the Sadr is variable cBuff
Default value: cKey = __ckey
Return value: oAes: Decrypt (cBuffer)

EnCryptFile (cSource,cTarget,cPass,bTest) // encryption file
cSource - crypted file
cTarget - decrypted file
cPass - key
bTest - block codeof execution, call: Eval (bTest, nCurrent_Pos, nMax_Pos)
Return value: lRet

DeCryptFile (cSource,cTarget,cPass,bTest) // decryption of file
Return value: lRet

cNext_No (c, n)
The return string value follows The value in relation to the specified value .
Example: cNext_No (' AG001 ')
Returns the value: AG002
Default value: n = Len (c)
Return value: cRet

aMaxStrLen (aString) / the maximumlength of the element in the array
Return value: nLen

a2c (a) // converts an array in a string
Return value: cArr//{..,}

x2c (c, cNavod, lTrim, nLen) // convert any value in string
Default value: lTrim =. T.
Return value: s

c2x (c, cType) // converts a string to the value of the specified type: N C D L A
Return value: s

SayMsg (aMsg) //// opens dialog with the amsgmessage. Message can be any value and array
Return value: NIL

StrToken (cTxt, cSep, nPos) // returns the value of the specified parameter from the text ctxt
Default value: Npos = 1
cSep = ', '
Return value: cRet

x2n (c) // any value in the number of
Return value: nVal

x2d (c) // any value in the date
Return value: dVal

x2l (c) // any value in the logicku
Return value: lVal

x2Type (c, cType) // any value in the value of the specified type
Return value: xVal

d2c (d) // date in character
If the date value is less than 01.01.1900 returns: '. . '
Return value: dToc (d)

c2uc (c) // String in Unicode string
Return value: C, a Str2unicode (c)

uc2c (c) // Unicode in string
Return value: C, Unicode2str (c)

cNavod (cTxt) // and the value of the string enclosed in quotationmarks.
If the cTxt contains the character ' (apostrophe) then it is a value enclosed in quotation marks "" otherwise, it is a value under the apostrophic "
Example 1: cnavod ("Hello Mr's ") "Hello Mr's"
Example 2: cnavod ("Hello Mr") to "Hello, Mr. "
Used to format a stream when memory is saved to a file.
Return value: C

Info ([..,]) // display the message (dialog) from the specified parameters of any type
Example: Info ("CurrentDate:", Date ())
Return value: NIL

DeskSize () // Gives you desktop size
Return value: aSize//{nWidth, nHeigth} AppDesktop (): CurrentSize ()

Dec2Hex (n) // Convert numeric value in Hex number
Return value: cRet

Hex2Dec (cH) // Convert Hex number in numeric
Return value: nRet

__Hex2Str (cHex) // HH, hh, hh → String
Return value: cData

__Str2Hex (cData) // String → HH, HH..
Return value: cHex

Hex2Str (cHex, lExtend) // Convert Hex values in string
Default value: lExtend =. F.
IF lExtend
Return value: cData

Str2Hex (cData, lExtend, nL) // Convert string to Hex values
Default value: lExtend =. F.
Return value: cHex

FWrite_s (h, s) // Entering text (value s) in a file whose handle is h
Return value: NIL

FWrite_Ln (h, s) // Same as FWrite_s with new line added
Return value: NIL

FRead_s (h, N) // It reads text of length n from a file whose handle is h
Return value: Left (S, N)

FRead_Ln (h, nLen) // Reads the maximum length line nLen from the file whose handle is h
Default value: nLen = 4096
Return value: (Left (c, N))

IsDir (cPath) // Checks the folder. Returns the logical value
Return value: lIsDir (c)

MkDir (cPath) // Creates a folder
Return value: lMkDir (c)

Sql2d (cTxt) // Sql date format in date value
Return value: cTod (substr (cTxt,9 2) + '. ' + substr (cTxt,6, 2) + '. ' + Left (cTxt, not)

Call_Block (bBlock, oSelf) // call block code
Return value: lRet // if it is committed . T. If there 's a mistake there will be value .F.

GetHostIP () // Returns the IP value of your WEB Server
LOCAL cFTPServer: = ' www.ip2location.com '
Return value: cIP

aCopy2 (a1, a2) // copies one data set to another
Return value: NIL

ChoiceYN (_aMsg, alPrompt, nInit) // Display dialog box with the selection of specified values
If alPrompt is a logicka value , it offers a Yes / No statement (Yes/No)
In this case , the logais returned to a value that is opsia chooses

In the second case , alPrompt mo can be a series of specified option descriptions.
Example: {'One ', ' Two ', ' Tree'}
Then the number of selected elections
Default value: nInit = 1
Return value: lRet / nRet

PadC (cTxt, nLen) // Aligns the specified string, adding a left and right blank character
// in length nLen string
Return value: PadR (Space ((nLen-Len (cTxt))/2) + cTxt, nLen)

GetLocalIPAddress () // Return the local IP address of your PC
Return value: (IIf (Empty (cIP), ' 127.0.0.1 ', cIP)

GetLocalName () // Return the name of your PC
Return value: cName

Is_Var (oObj, cName) // Returns the logical value . T. If the specified variable (cname) exists
// in the specified object
Return value: aScan (ClassInfo (oObj, ' VARS '), {| x | Upper (x) = = Upper (cName)}) > 0

Is_Method (oObj, cName) // Return logic value . T. If the specified metof (cname) exists in
// specified object
Return value: aScan (ClassInfo (oObj, ' BY '), {| x | Upper (x) = = Upper (cName)}) > 0

ClassInfo (Oobj, CType) // Return in array a series of names of variables or methods of the
// specified object
cType can be ' by ' or ' VARs '
Return value: Array

Sec2Time (n) // return formatted value of seconds in the string time format hh: mm: ss
Return value: Strzero (H, 2) + ":" + Strzero (M 2) + ":" + Strzero (s 2)

cbEval (xFn [, p1 [, p2..]])//Performs xFn depending on type:
The logical value returns xFn
For a block code called Eval (xFn [, p1..])
For a string value, performs F-ju: & (xFn) (P1, P2, p3, P4, P5, P6, P7, P8, P9, P10,;
Return value: Xvalue

aToken (cTxt, cSep) // Returns a series of values specified in the cTxt
Return value: _ aToken // separate delimator cSep

Show_Printers // Display dialog with Printers
Return value: NIL

Config_Prn // oPrn:Setupdialog () - printer configuration
Return value: NIL

cStr2CP (cTxt) // Convert Unicode in string or ANSI
```
IF isUnicode (cTxt)
   cRet: = Unicode2Str (cTxt)
ELSE
  cRet: = ConvToAnsiCP (cTxt)
ENDIF
```
Return value: cRet

Font_W (a, b, c)
Return value: GetTextWid (c, a, b)

Font_H (a, b, c)
Return value: GetTextHei (c, a, b)

_SayA (nRow, nCol, _cText , oWnd, cPict, oFont, lCenter , lRight , lBorder ,;
lPixel , nClrText, nClrBack, nWidth, nHeight ,;
ldesign , lupdate , lShaded , lBox , lRaised , Chr_Align,;
lRAlign)
// Call say is right - aligned
Example:_Saya (5.5, "Get Name", ownd,,,,,,,,, 100.18,,,,,,': ',.T.)
Result: "Get name.....:"
Default value: Chr_Align = ': '
The lRAlign =. F.
Return value: o

Null (cType, nLen) // returns the zero - value of the selected type
Return value: xDef

Macro (cTxt) // committing macro cTxt
Return value: xRet

TypeStruc (acType)
Return value: xRet

File_Size (cFile)
Return value: N

File_Load (cFile, lArr) // Loads the file as string or as a series of lines if the parameter
// is lArr =.T.
Default value: lArr =. F.
Return value: uRet

File_Save (cFile, cBuffer) // Record cBuffer in the file name
Return value: NIL

c2a (cTxt, cCR) //
Default value: cCR = Chr (13) + Chr (10)
Return value: Array

Shift_Pressed // if the pressed Shift key is returned .T.
Return value: GetKeyStat (VK_SHIFT)//(KbdStat (1) = 1. Or. KbdStat (2) = 2)

fWriteLine (nHhandle, cTxt) // Writting a line (cTxt) in an open file handled by Handle
Return value: fWrite (nHandle, cTxt + CHR (13) + CHR (10), Len (cTxt) + 2)

fEof (nHandle) // is end of File
Return value: lEof

fLineCount (nHandle) // Returns the number of file lines.
Return value: nLine

FSeekLine (nHandle, nline, nMode)// Positioning on the line number in file
nMode = 0-Begin file
1-Cuur position
2-End file
Return value: FSeek (H,0, 1)

Freadline (nHandle, nDirec, nLen, cCR) // Reads the line from the open file
Default value: Nlen = 512
cCR = Chr (13) + CHR (10)
Return value: cRet

Temp_Path // or who does not exist creates a temporary folder and returns the
// same ctemp:= "c:\temp\" + &_oWEB_: Cby + ' \ '
Return value: cTemp

cSelect (cAlias) // if there is a working area , it returns its name and selects it and
// value .F.
IF ValType(cAlias) == 'C'
IF aScan (WorkSpaceList(),{|x| x==cAlias})=0
Return value: .F.
ENDIF
ENDIF
dbSelectArea (cAlias)
Return value: (Alias()==cAlias)

aGetBorder (oDlg) // return array of coordinates of the dialog box
LOCAL o1 := oDlg:getRect() ,;
o2 := oDlg:getCliRect()
Return value: { o1:nRight -o1:nLeft - o2:nRight + o2:nLeft ,;
o1:nBottom -o1:nTop - o2:nBottom + o2:nTop }

aGetCoors (oDlg) //
LOCAL o: = oDlg: getRect ()
Return value: {o:nTop, o:nLeft, o:nBottom, o:nRight}

Dlg_Coors (oDlg, oParent, aPos, aSize)
Default value: oParent = GetDesktop ()
a2 := oDlg size / a := Desktop size
oR: nTop: = a [1] + a2 [1]

oR: nLeft: = a [2] + a2 [2]
oR: nBottom: = a [3] + a2 [3]
oR: nRight: = a [4] + a2 [4]
Return value: oR

aVarStruc (cVar, xVal)
Return value: {cVar, cType, nLen, nDec}

aFile_Read (cFile)
Return value: aLn

aFile_Write (cFile, aLn)
Return value: NIL

Load_Var (cFile, aDef, lLocal)
Loads variables from the file as arrays: {{cVar_Name, xValue, cType, nLen, nDec},..}
aDef - a series of standard values: { { cVar_Name, xvalue, CType, nlen, ndec},..}
Default value: aDef = {}
The lLocal =. F. // Local is. T. or Server is. F.
Return value: aVar

Save_Var (cFile, lLocal) // Return and value variables in a predefined file
Default value: lLocal =. F.
Return value: aVar

IsKeyPress (cKey) // Checks the status of the key (is key pressed)
cKey can be: "CTRL" or "SHIFT" or "RETURN" or numeric value
If a key is specified, returns a value. T.

Return value: lRet

Word3Hex (n) // Returns the hex value of the specified number of 3 hex numbers
Example: Word3Hex (241) = > 0000F1

Return value: HHHHHH

Load_Mnu (cFile, lWEB) // Loads the menu from the specified file
If the lWEB file is being read from a WEB folder. Otherwise, from the local disk.
Content. MNU file:
[TITLE] = Title_text
[PROMPT] = Name_item_1
IF = lbCondition
CALL = Block Code
[PROMPT]..

The IF parameter does not have to be specified. If the condition is specified, and if it is filled with a menu Exists.

Return value: aRet //A series of menu values {{Menu_Name, bCall},..}

DLG_FSave (cMask, cDefault_Path, cDefault_Name, cTitle)
Invites the file's Recording dialog
Return value: cFileName

DLG_FOpen (cMask, cDefault_Path, cDefault_Name, cTitle)
Invites the dialog box to open the file
Return value: cFileName

DLG_Folder (cPath, cTitle)//Displays dialog box to select Folders
cPath is a standard folder suggestion
Return value: C

Get_Computer_Name ()
Return value: cComputer_Name

Str_z (nValue, nWidth, lZero, nDec, lNoDec, lCondition, cComa, cSep)
Formats the specified number of nValue

Default value: Nwidth = 16 // length of formatted stream
lZero =. F. // If True and the value of nValue = 0 returns blank
nDec = 2 // Decimal
lNoDec = .F. // If True and the number does not have
// decimals it is not displayed As. 00
lCondition =.T. // condition that if the unfulfilled function
// returns a string of blank length of nWidth.
// The lCondition can be a code block
cComa = '. ' // Display a decimal point view
Csep = ', ' // Separator for the separation of thousands

Example: Str_z (1234.5, 10,. T. 2)-> "1,234.50"
If nValue is zero and lZero =. T. Then returns the string length 10 (blank)

Return value: PadL (s, nWidth)

cAdd_Coma (cTxt, W, oDlg, oF) // format streaming by adding symbols "." (point) and
// ultimately ":" (colon) According to the selected font and
// length in pixels (w)
Example: cAdd_Coma (' Name ',100, oDlg, oFont)
Returns a value: ' Name.....: '-> This string when it is displayed in the selected dialup
The selected font will have the length of 100 pixels.
Return value: c + ': '

Random (nLimit) // Random F-ja-> returns the number of values Between 1 and Nlimit
Return value: Numeric

// Suggested string with a similar choice of values according to a specified stream in cMask
cRandomPass (cMask)
Default value: cMask = ("UUNNNULXNNNXLUNN")
The cMask stream contains wildcard symbols that are converted to accidentally selected values: N - digit
U - Uppercase Char
L - Lowercase Char

X - Any sybbol of: **#! & @ *-+ =.,**

Example: cRandomPass ("UNNLX")-return the string length of 5 characters where the The first character of the capital letter, followed by two characters (digits), followed by a small letter and then one character, some of the symbols randomly selected.

Return value: cPass

IsRunning (cName, lCount, lArr) // if the program is already running.T.

Default value: lArr =.F.
cName = cAppName
lCount =. F.

cName is the name of the application that is checked to make sure the current PC is running.

If you are a "lArr". F. Then returns the logical value
(is the app already running)
Return value: lRet

Otherwise, if the value is lArr =. T.
Returns a series of two values: {lRet, aObj}
The first parameter is the status of the running application (True/False) and the second parameter is a series of objects indicating the application.

In both variants, the lRet value will be nonnumeric if lCount =. T.
This value will indicate the number of processes that are running (how many times the application Running

File_Info (cFile, lList) // Information about the file: { name, velic, date, time }

Default value: lList =. F.
If the lList parameter =. T. Then returns the sequence of information about the files
In this variant, cFile can also contain wildcard symbols: *.*, etc.
Return value: {{cFile_Name_1, nSize_1, dDate_1, cTime_1},..}
If lList =. F. Then the single file information is returned:
Return value: {cFile_Name, nSize, dDate, cTime}

Bye ([cMsg])

If the message is specified, it first appears on the screen, and then the Program closes.
Callsto getQUIT
Return value: NIL

_Strpm (cStr, nAdd)

Default value: nAdd = 1
Value in cStr increases (decreases) for the specified value (nAdd can be a positive or negative number)
Example: if cStr = ' ABC0010 '
_Strpm (cStr) = > Returns a value: ABC0011
_Strpm (cStr,-2) = > Returns a value: ABC0008
Return value: PadR (cRet, Len (cStr))

SetCoors (oDlg, Y, X, W, h)

Sets the position (Y-column, X-row) and the Raw (w) and the height (h) of the specified "oDlg"
Return value: NIL

Call_FRP (cFile, cPaper, cExport, lSilent, lLand) // creation of a call to the FRP file
Return value: NIL

c2Arr_Param (cP, aDef, cSep)
Default value: cSep = ', '

From the string value cStr forms a sequence parameters to get separate.
Separator specified in the cSep.
A series of which isa Rebar of a series of adef that contains standard values.
Example: c2Arr_Param ('123, "Xbweb", 01.12.2018,. T. ', {0, ' ', Date (),0,. F. 2})
Return series: { 123, 'xbweb', 1.12.2018,. T., 2}
There are standard values in the aDef-Revaluation parameter and also types of values.
Type a series of values separated by separators less than
The length of the adef array will be added to standard values. In this case , it is number 2 as the last element of the array.

Return value: Array

GetFontName (oFont)
Return: Font name (Example: 12.Arial Bold Italic)

aReadMemo (cMemo)
Return: an array of content lines memo fields