Programming tutorial for PSG & VFP

Programming Solutions Group

Note:
Please check PSGSDK website for the last relase of this tutorial

www.psgsdk.com

Table of Contents

1 Introduction

PSG programming tutorial for Visual FoxPro programmers.

Based on VFP demo project on PSG platform, this document present step by step how to create an application module using base classes and templates of PSG.
Programming client/server database applications is easier than ever using your programming language at choice.

References:
- PSG demo projects.
- PSG server programming help file
- PSG client VFP programming help file

Requirements:
- download and install the demo project on PSG for VFP (www.psgsdk.com)
  - PSG Server manager
  - PSG Demo Project server
  - PSG Client

- Visual FoxPro 9 programming language
- The Demo project includes NET-Reports engine
  - for Excel reports Ms.Office is required
  - Ms. Query from Ms. Office CD should be installed for Excel PivotTable reports (it is not installed by default with Ms.Office by some distributions of Ms. Office)
- Visual FoxPro programming skills (this tutorial covers only the basic but nothing related to VFP programming in general like adding controls on form or using the VFP IDE)

The demo project where built in fifteen working hours during a week time. Considering the well known Northwind used database documentation time was short also.
Here we will present step by step the "Orders" module of the demo project. The source code is available for entire project.

PSG officially do not support any VFP wizard for building applications or forms. All samples here are based on templates and base classes for VFP.
Please check the website for news.

1.1 Welcome topic

PSG could be used with different programming languages like C#,Java and VFP.
We have found VFP a very strong database oriented programming language from many years. His power has no equal even this days if we consider database programming.
The client PSG application is fully compatible with Windows Vista and Windos 7 OS.

PSG extends the power of Visual Foxpro programmers offering next attributes for a more modern architecture:

- a n-tier client/server architecture for database applications
- base classes and templates
- database server independent platform
- high security communication protocol (HTTPS communication protocol and support for client

hardware keys by default)
- short development and implementation time
- REST based web service client/server communications
- users management
- automatic update of client application
- simple help system
- international toolkit
- integrated reporting and data analysis tools
- server side licenses management

Programming Intranet and Internet database applications are main task for PSG platform.
Please check the documentation that comes with the server and the PSG programming manual for VFP for more information.

1.2 VFP Demo project

The demo project for VFP on PSG where build around a modified version of Northwind database.

Northwind is a small and less complex database that's used from many years as a demo database, there is a chance to be already known by many programmers, that give us the reason to use it for demo and tutorial.

The differences form the original version of Northwind where in structure of some tables.
All tables that used to have a numeric autoincrement primary key where changed to a twelve character primary key (and all references)
All original data are in the database, the DATE field values where changed to actual period to be easy used in reports.

Two fields where added to each table:

- sid - code of the last user who modified the record

- mdl - datetime of the last update of the record

There are three databases servers to chose from:

- Ms. SQL database server

- PostgreSQL

- Sqlite 3.0 (embedded database server component)

The application work in the same manner with all of them (Sqlite 3.0 can be used with fewer users only).

By default the server is installed with the Sqlite 3.0 database as there no installation is required for the database.

The databases backups for PostgreSql and Ms. SQL are in the DB_back directory. To change the database just install the databases from backup and modify the server connection strings as follow.

There are two connections strings to be modified:

- server main connection (server configuration program)

- reports management - connections

Samples of connection strings:

PostgreSql	Driver={PostgreSQL UNICODE};Server=localhost;Database=northwind;Port=5432;Uid=postgres;Pwd=123456;SSL=No;
MS.Sql	Driver={SQL Server};Server=ADRIAN-PC\SQLEXPRESS;Database=northwind_psg;Uid=sa;Pwd=123456;
Sqlite 3.0	DRIVER=SQLite3 ODBC Driver;Database=C:\projects\wwb\db3\psg\northwind.db3;LongNames=0;Timeout=1000;NoTXN=0;SyncPragma=NORMAL;StepAPI=0;

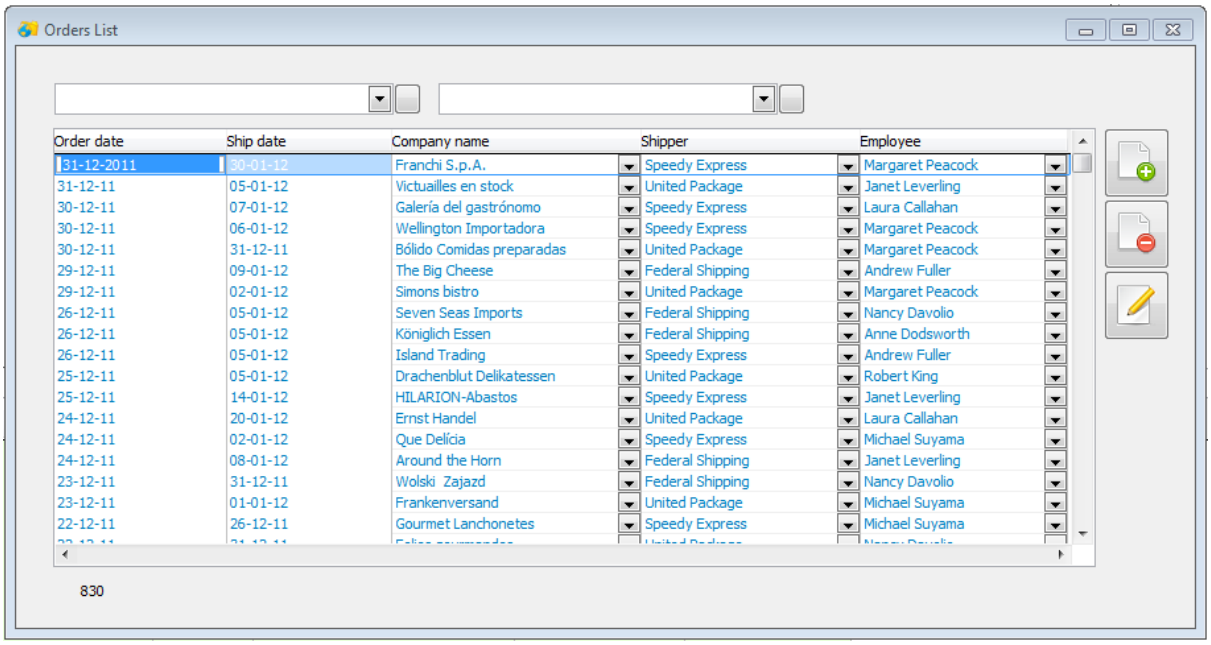
2 Northwind "Orders" module

The Northwind database is simple offering the basic functionality for a sales recording application.

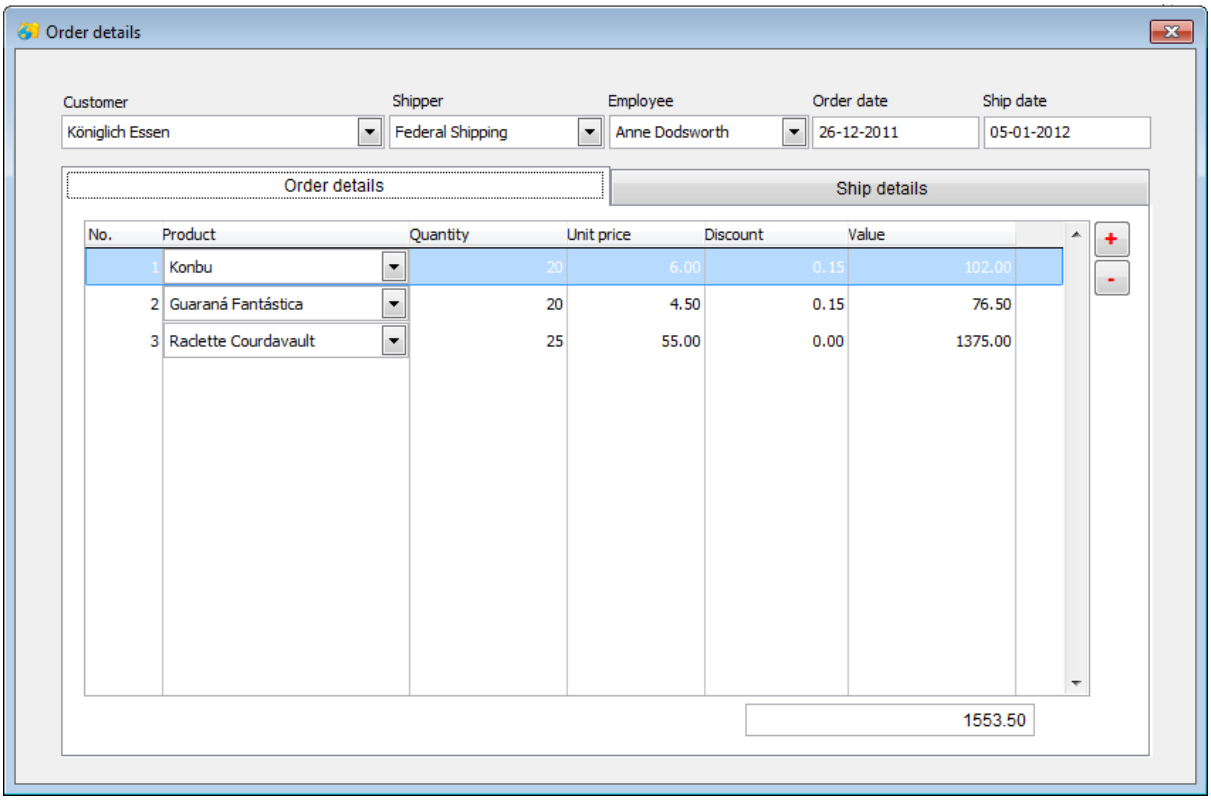
The Demo application is built around Northwind database, covering the interfaces for all lookup tables and the main module that record the sales orders.
On top of that the NET-Reports engine present some nice reports and data analysis.

2.1 Overview

The 'Orders' module:
- Orders list form



- Order details form



Tables used by orders:

orders	orders list
orderdetails	order details
customers	list of customers
employees	list of employees

products	list of products
shippers	list of shippers

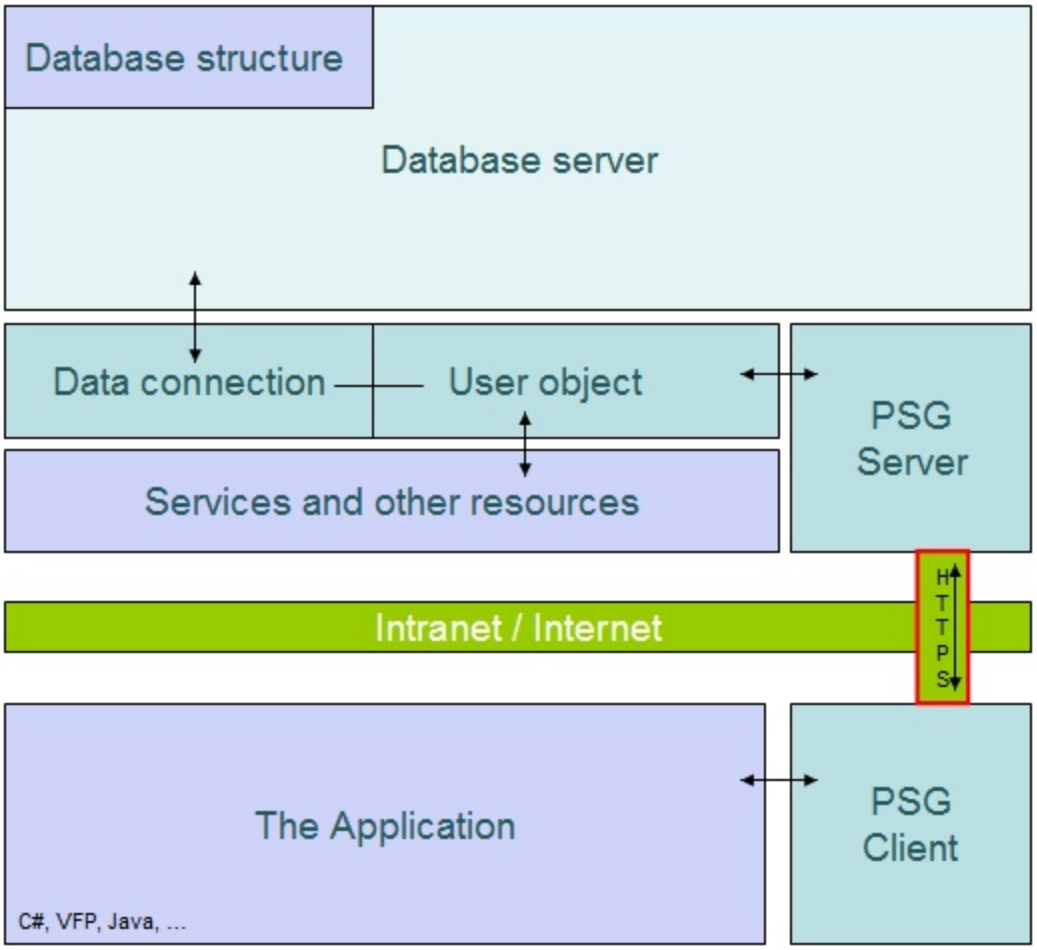
We need to create the PSG web services and the presented user interfaces.

2.2 PSG programming model

PSG platform offers a multi tier programming model.
Next picture offers a quick view of the platform main components.

The Application - all modules that interact with the end user, the application user interface.
PSG Client - Offers the main screen and menu for the application and the communication object PSGCON.
PSG Server - communication and application server
User object - for each user there is an user object instantiated on PSG server
Data connection - the connection to the database server, one connection for each user
Services and other resources - requests sent to the server are based on web services, the service model is REST.
There is a big difference between REST and SOAP web services, while SOAP services are objects to be instantiated by the client, the RESTf model is more simple and deals with requests and responses. The REST model is more reliable on heterogenous networks and offers better scalability, it doesn't need WSDL XML files and there is nothing to be registered or instantiated on the server OS, the response is very fast compared with SOAP model.
Database server and database structure - these are not part of PSG platform, are represented just to complete the picture, any relational database server like Ms.Sql, PostgreSql, Oracle, Sybase, MySQL and others could be used and off course the database itself.

To simplify the concept we consider the PSG server and the PSG client as one application tier. Considering this, the model works as a four tier. As a programmer of a PSG application you can interact with three tiers : the application, services and the database structure/database triggers if used.
Business logic of the application should be balanced to the platform levels as needed, better to put into services or database triggers.



To create a PSG based application you will need first a database.

Programming with PSG:
- programming PSG web services
- the application user interface modules
- the database

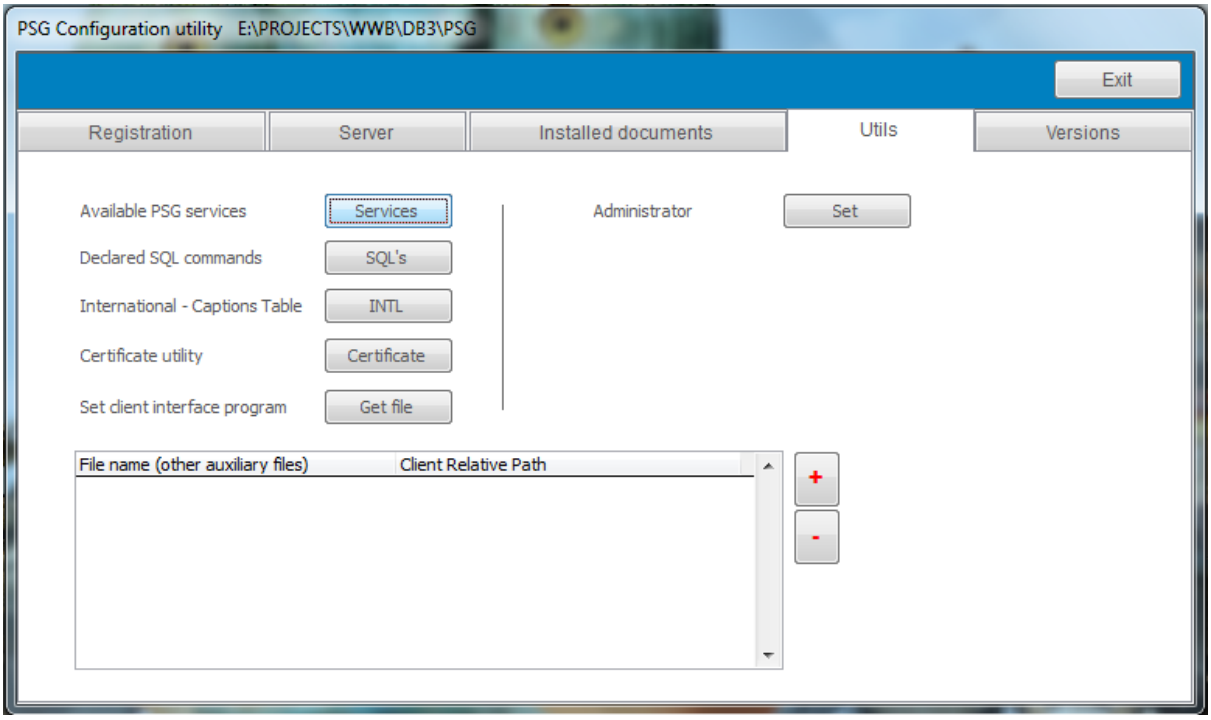
Next we will keep focus on the tutorial topic, please check the documentation for more details.

2.3 Web services

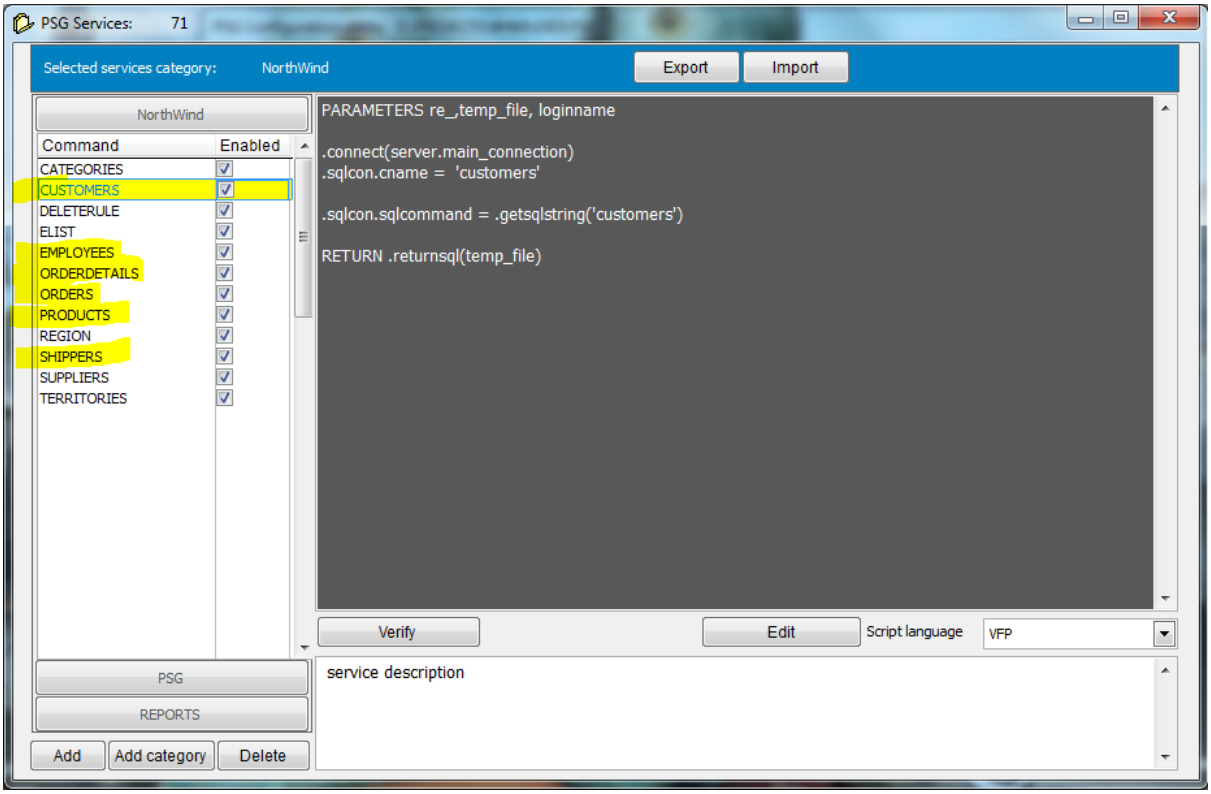
Created PSG web services for "Orders" module are used for data requests only.

For each required table we need a web service that prepare the data on server to be transferred.

Use the server configuration utility to open the services manager.



Service editor and used services for the ORDERS module:



Sample of "CUSTOMERS" service:

```
PARAMETERS re_,temp_file, loginname

.connect(server.main_connection)
.sqlcon.cname =  'customers'

.sqlcon.sqlcommand = .getsqlstring('customers')

RETURN .returnsql(temp_file)
```

This service is used to return the list of customers without any filter.
The service is used by another one called "DATA", the DATA service is used to request temporary tables from the server to be used locally client side, please check also DATA service presented into the server programming help manual.

The web services editor uses a category list of services Northwind, PSG, Reports respectively application services, base services (PSG), NET-Reports reporting engine services (REPORTS). Services could be written using VFP, C#, VBscript, Javascript. Good to know that all base services are written in VFP. Services written in different languages could coexist without interference. The service name is unique regardless the category name, the category is here just to help organize the services has no influence at run time.

Coming back to "CUSTOMERS" service, next will explain line by line:

```
PARAMETERS re_,temp_file, loginname
```

re_ - is here for backward compatibility, the parameters where used into PSG 1.0, some applications still use it.
temp_file - name of the archive to be sent back to the client.
loginname - username, added automatically to each service request by the user object accept method.
If there was any parameter sent to the server, it should be between 're_' and 'temp_file'.

```
.connect(server.main_connection)
```

check if the used ODBC connection is pointed to the main database, if not a connection to the database is established.

```
.sqlcon.cname =  'customers'
```

set the returned cursor name

```
.sqlcon.sqlcommand = .getsqlstring('customers')
```

set the SQL string to be sent to the database server, to be more easily managed there is the possibility to store the command, 'GETSQLSTRING' is used to retrieve the command. Use "SQL's" from "Utils" page of the server configuration to access stored SQL's.

```
RETURN .returnsql(temp_file)
```

RETURNSQL - execute the SQL statement and returns the cursor

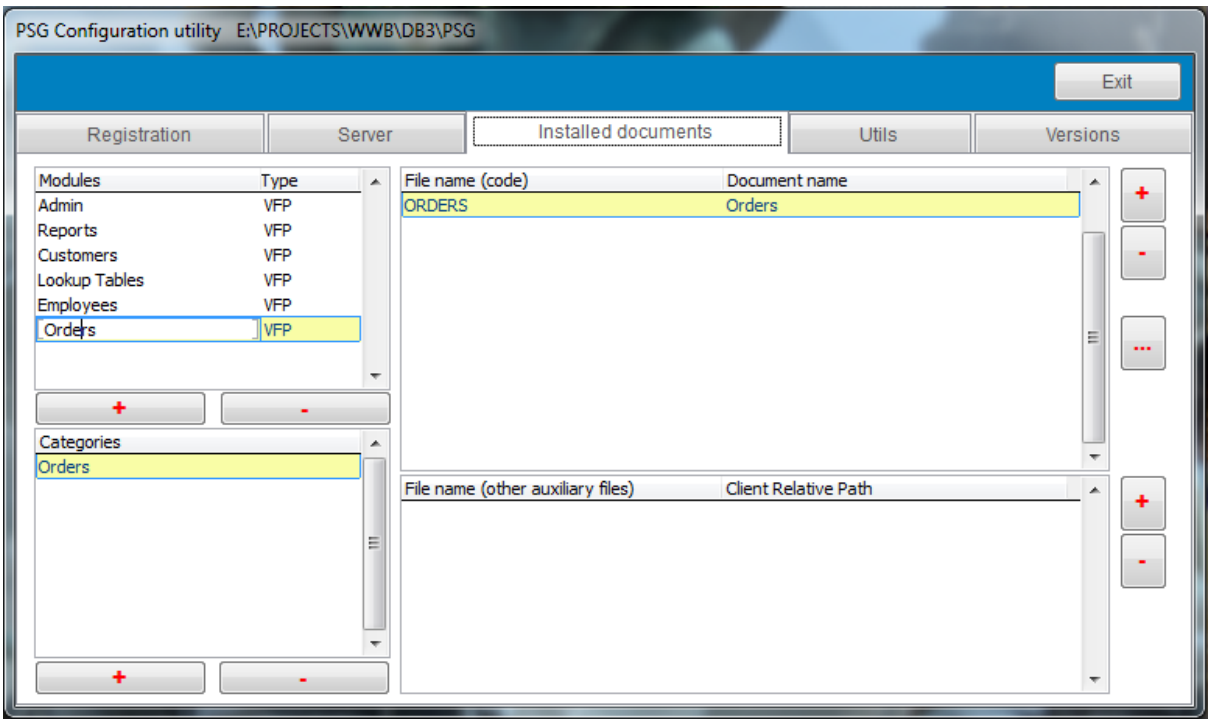
To create a service, just copy and paste from another one and modify the code to fit the new service.

Services could be used for a lot of purposes.
By using services a communication between client and server could be established as the response from server could be a command to be executed client side if implemented.
Please check the documentation of GETDATA into client programming help file for more details.
Some of the services here where created for others application modules and are used for ORDERS too.

2.4 The application

The client application offers a basic user interface that loads the application menu and initial modules like report engine. The main application screen embeds also the communication object PSGCON. By using the PSGCON the client could send request to the server. The server will respond with data or a command to be executed locally client side, a kind of devices communication could be established using web services and local commands. For simple applications like one presented here only basic communications is needed that's handled automatically, base classes and templates are available for each control that need to communicate with the server for insert, update or delete.

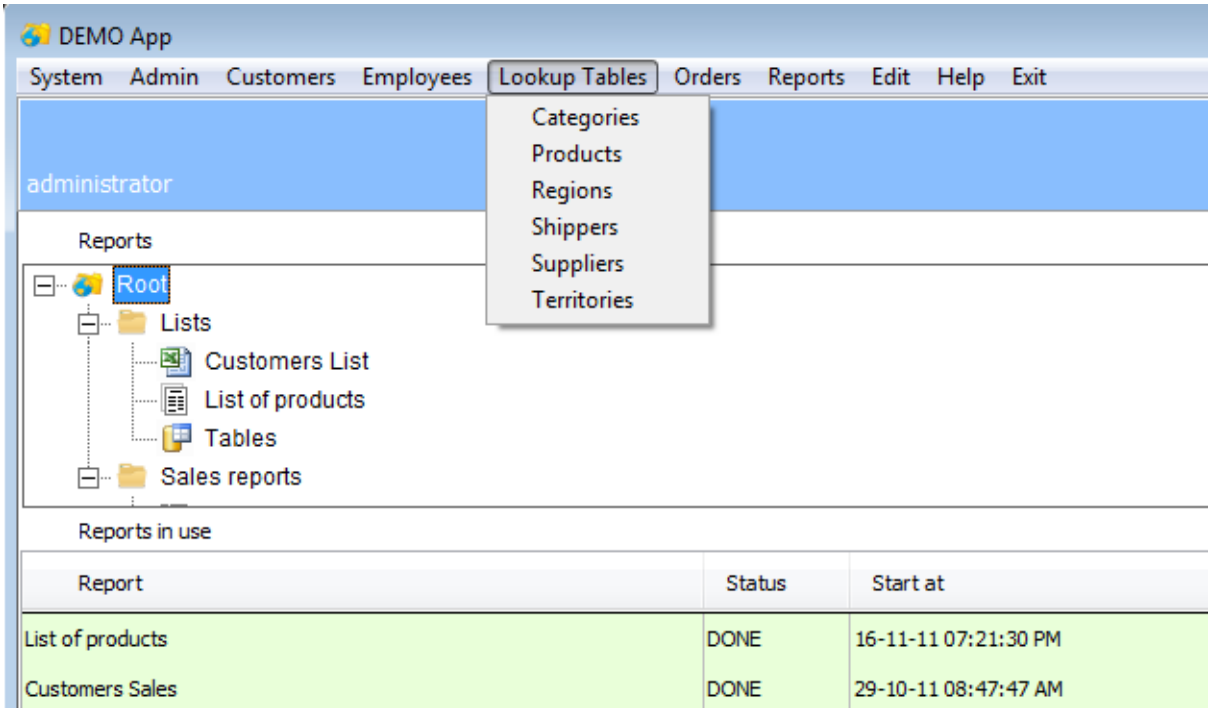
Orders module is started from the main menu of the application. All used modules should be registered on server using the server configuration utility, the registration is stored into the PSG structures no operating system registration is needed. By registering the module the application menu is created also.



One module should be registered, after that category and finally the document. After that the executable file should be added to the document record.
For VFP a document is a compiled project as exe file, for C# the document could be a DLL. A VFP application main interface could load only VFP documents now, C# interface could load also VFP exe documents.

Next time you could use a small utility (publish.exe) to register automatically the new release of the exe file to the local computer PSG server at development time.

Each module has categories and documents, finally the module will translate into a menu item and the documents into menu pads under an item. Categories are here just for better management of added documents.



System, Admin,Edit, Help and Exit are automatically added to the main menu.
F1 for help, a web based help is opened into a small web browser window based on IE controls.

2.5 Orders module - tutorial

There are no wizards used here to automate tasks, only PSG templates and basic classes for VFP are used.
PSG SDK for VFP do not provide any wizard at this time, please check the website for news.

This tutorial do not cover all objects and methods involved into the project forms or containers leaving you details to work on as you may want.
Check also the demo projects that comes with the source code.

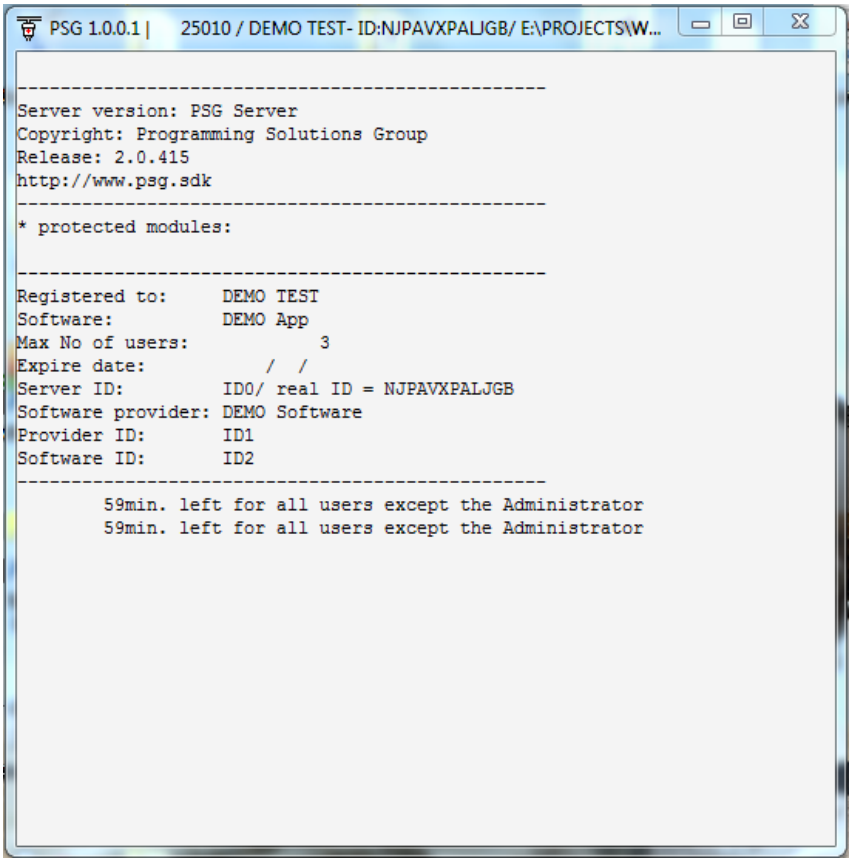
2.5.1 1. Orders project requirements

1. Prepare the working place:

1.a A instance of PSG server should be opened, better to have the PSG DEMO for VFP where the Nortwhind demo application is used.

The PSG server can run as a service in production, preferable to have it run as an application for development.

Open the psg.exe in the server director. By default the server has a demo licence that allows tree users for one hour, after that only the administrator can use the server.



Messages from the connected user object are directed to the screen.

1.b Create a director for you project on your hard drive and copy next files there:
- template.scx, template.sct (a loading form template, used to request and load temporary tables from server)
- psgtext.vcx, psgtext.vct (base class for text control, embeds code to communicate with the server via PSGCON)
- psgedit.vcx, psgedit.vct (base class for edit control)
- psgcombo.vcx, psgcobo.vct (base class for combobx control)
- publish.exe (a utility that helps publish a document to the local computer server, very useful at design time)

1.c Open Visual FoxPro 9
Navigate to the new project directory.
Create a new project named "orders"

2.5.2 2. Main program file

2.a Create the main project program file named orders.prg

2.b Write next code:

```
m.objectname = psgcon.getnewobject()
DO FORM orders_list WITH m.objectname NAME &objectname
```

2.5.3 3. First loading form

3.a Open form template.scx and save as orders_list.scx

3.b Open orders_list in form designer

3.c Modify the INIT method of orders_list

```
PARAMETERS ce_
THIS.Name = ce_

this.interface_class  = 'orders'
this.interface_container  =  'usrcontrol'

this.getdata.addtable('categories')
this.getdata.addtable('shippers')
this.getdata.addtable('customers')
this.getdata.addtable('elist')
this.getdata.addtable('orders')


this.getData.gettables()
```

For each requested table a corresponding web service should exist on server.
Please check web services for details.
The GETDATA object deals with server communication, please check the SDK help file for more information.

3.d Modify the LOAD_DATA method.
Generally for simple lists there maybe nothing to modify or add here.

Here we have added code for:
- first we need an index on ORDERS table in order to display data ordered by date descending.
- we need a small cursor from one table (code added as sample, data could be retrieved from the server service as needed)

LOAD_DATA is triggered after the data archive was loaded from the server and need to be used

```
PARAMETERS ce_
LOCAL class_, interface_, nial
FOR m.nial = 1 TO this.getdata.aliases.Count
            SELECT 0
            m.ext_file    = this.getdata.aliases.Item(m.nial) +
this.getdata.file_ext
            m.alias_name  =  this.getdata.aliases.Item(m.nial)
            IF LOWER(m.alias_name) = 'orders'
                              jjj = "use work_\"+ m.ext_file + "
ALIAS " + m.alias_name + " exclusive"
                              &JJJ
                              INDEX on orderdate TAG orderdate
```

```
            ENDIF
            jjj = "use work_\"+ m.ext_file + " ALIAS " +
m.alias_name + " SHARED"
            &JJJ
            this.remove_nulls()
            this.getdata.tables.Add(m.ext_file)
ENDFOR
SELECT ALLTRIM(firstname) + ' ' + ALLTRIM(lastname) as employee,
employeeid FROM elist INTO CURSOR elist2
SELECT elist
USE
SELECT orders
SET ORDER TO orderdate DESCENDING

class_ = this.interface_class
SET CLASSLIB TO &class_
this.AddObject(this.interface_container,this.interface_container)
```

2.5.4 4. Loading form interface container

When opened each loading form need to load data from the server, therefore no interface control could be displayed as it has no control source available.
The LOAD_DATA method of loading form also loads a container with all needed controls for form interface.

4.a Create a new class into the project
usrcontrol based on container and save to orders.vcx

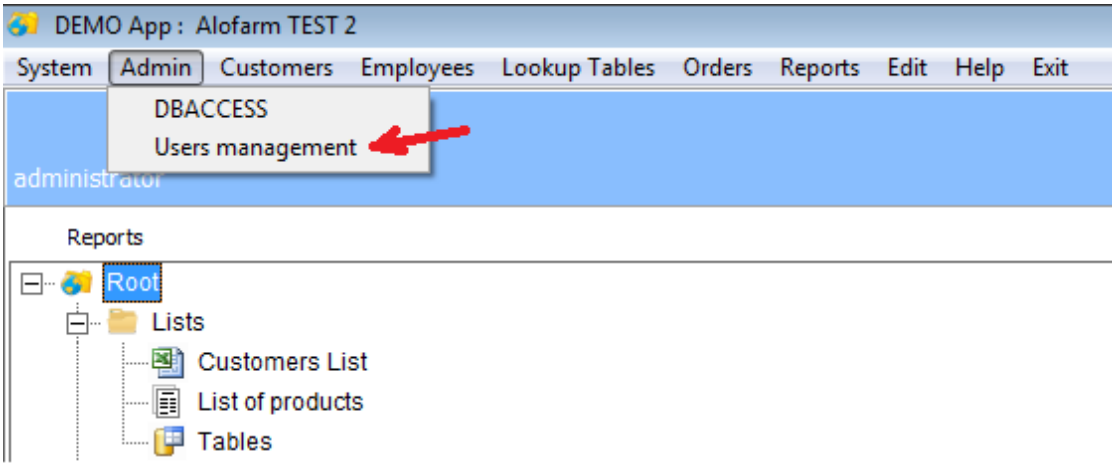
4.b Modify the INIT method of the usrcontrol and add next code:

```
&& position the control into the form and the form in application
main interface
this.top = 0
this.left = 0
this.Parent.Top   = int((_screen.Height - this.Height)/2)
this.Parent.Left = INT((_screen.Width   - this.Width)/2)
thisform.Height = this.Height
thisform.Width  = this.Width
thisform.Refresh

this.Visible = .t.

this.Anchor = 15 && set this control to scale with the window

TRY
      this.parent.Caption = psgcon.intl[2044] && International
```

```
caption loaded by PSGCON when the application start
CATCH
ENDTRY
thisform.ShowTips = .t.
thisform.image1.visible = .f.

PUBLIC m.srchstring && variable needed by this project later
```

4.c Till now we have first part of the project ready.
Compile the project and publish the exe file (document) to the server first time.
Use the publish.exe utility if the exe file is already registered/published into the server records.

Launch the client and give rights to use the new module to the Administrator.
Use the Users Management utility found under Admin menu



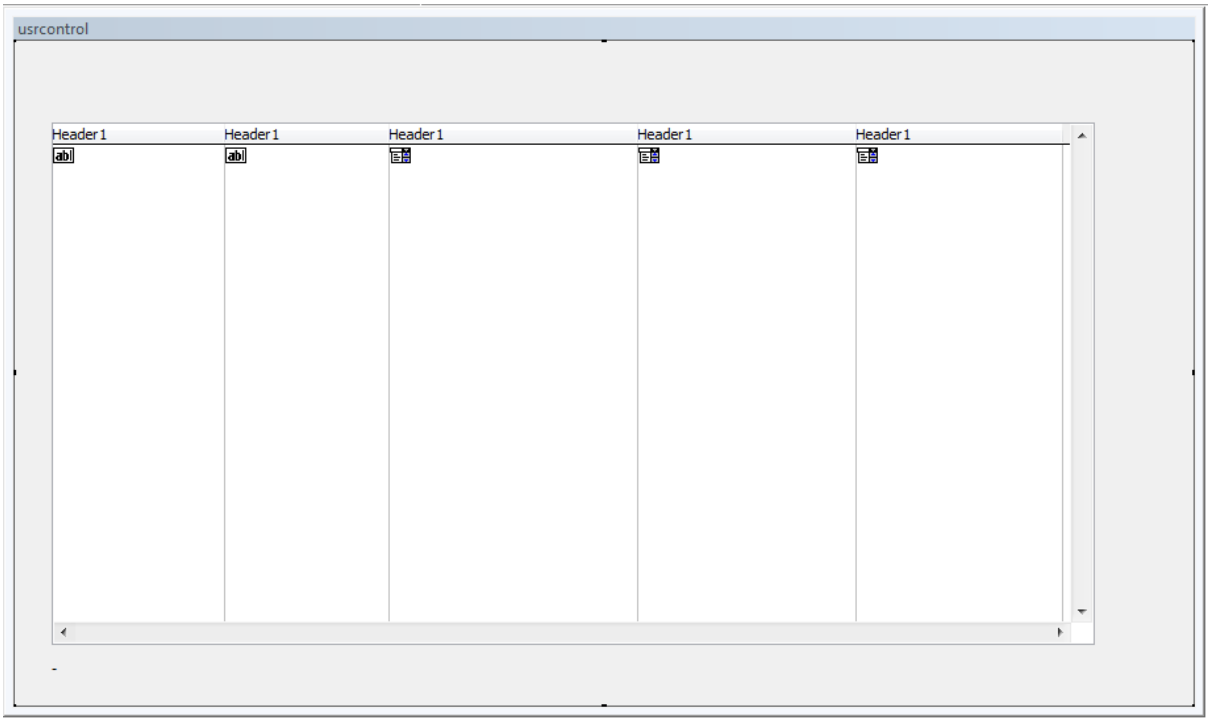
Close the client application and open again to refresh the access rights.

4.d Run ORDERS from the menu and.
One clear container will appear, if not you will have some errors messages.
If everything OK you can go to next step.

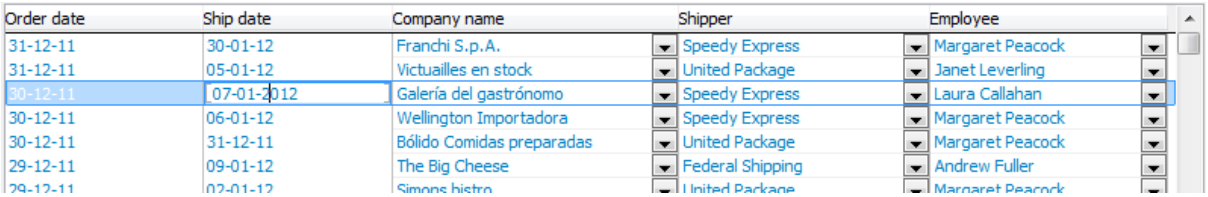
2.5.5 5. Adding controls to form container

Open the 'usrcontrol' container and start to add controls on it.
It is a good ideea to not add all controls once:
- add some controls
- publish the exe and open the client to check if works

5.a add the grid container



We do not use this grid to modify the data, it uses standard controls to present data from orders like next picture.



We have combo boxes to link the order table to customers, shippers and employees (elist2 cursor). Check the source code of demo project for more information.

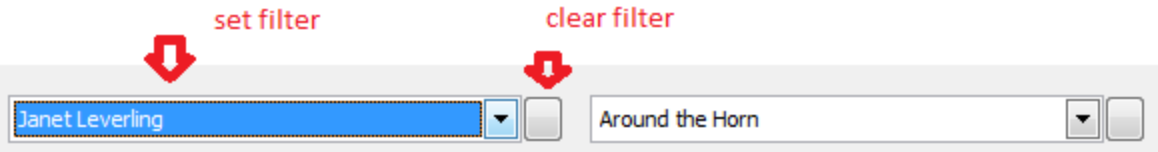
5.b Add international caption to grid column headers.

Open grid INIT method and write next code:

```
this.column1.header1.caption =  psgcon.intl[2045]
this.column2.header1.caption =  psgcon.intl[2046]
this.column3.header1.caption =  psgcon.intl[2001]
this.column4.header1.caption =  psgcon.intl[2027]
this.column5.header1.caption =  psgcon.intl[2047]
```

Captions are loaded form the server by PSGCON and stored to an internal array. INTL from PSGCON will give the right caption for language in use.

5.c Add filters options at the top of container.



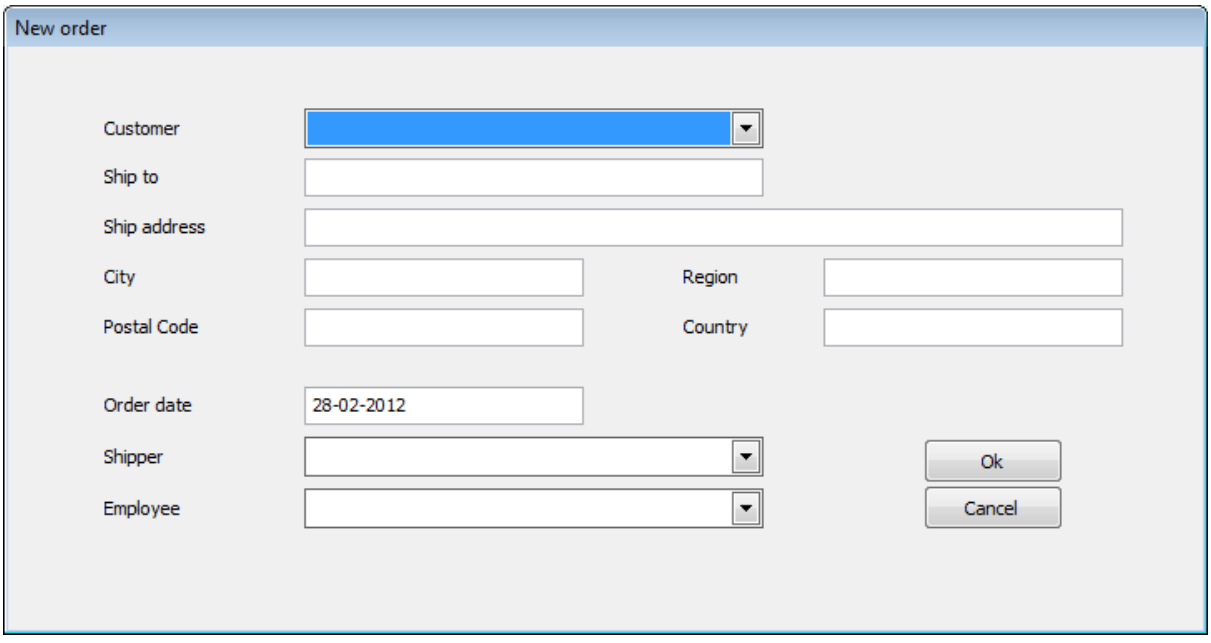
Feel free to write code for this filters or take the code from the demo source code.

5.d Add buttons for NEW, DELETE and MODIFY record.
Pictures are available with the demo project.

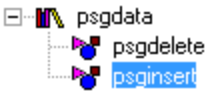
NEW record button click method code:

```
this.Parent.combo1.Value = ""
this.Parent.combo2.Value = ""
this.Parent.combo1.Valid

SELECT orders
m.ckey = orders.orderid
DO FORM orders_add

SELECT orders
IF orders.orderid # m.ckey
      this.Parent.label1.Caption = STR(VAL(this.Parent.label1.
Caption)+1)
ENDIF

this.Parent.grid1.Refresh
this.Parent.grid1.SetFocus
```

Create a new standard form "orders_add"



add a new control on form PSGINSERT from PSGDATA



"OK" button click method code:

```
m.orderid         = RIGHT(SYS(2015),9) + RIGHT(psgcon.usrcode,3) &&
create a unique record id of 12 characters
m.customerid      = this.Parent.combo1.value
m.shipname        = this.Parent.text1.Value
m.shipaddress     = this.Parent.text2.Value
m.shipcity        = this.Parent.text3.Value
m.shipregion      = this.Parent.text4.Value
m.shippostalcode  =  this.Parent.text5.Value
m.shipcountry     = this.Parent.text6.Value
m.orderdate       = this.Parent.text7.Value
m.shipperid       = this.Parent.combo2.Value
m.employeeid      = this.Parent.combo3.Value

this.Parent.psginsert1.insert('orders','orders','orderid') &&
communicate with the server for data insert
thisform.Release
```

"Cancel" button click method code just release the form

DELETE Record button click code:

Add "psgdelete" class from psgdata.vcx to the container first, this class deals with DELETE request.

Please check the SDK help file for more information on DELETE.

```
m.dltmsg = 7
SELECT orders
m.dltmsg = MESSAGEBOX('Delete record' + CHR(13) +
DTOC(orders.orderdate),292,'Delete  record')

IF m.dltmsg = 6
      &&table_name,key_name,record_id,local_table
      m.key_todelete = orders.orderid
      this.Parent.psgdelete1.delete
('orders','orderid',m.key_todelete,'orders') && delete record from
orders table
      this.Parent.psgdelete1.delete
('orderdetails','orderid',m.key_todelete,'orders') && delete
related records from details table
      this.Parent.label1.Caption  =
STR(VAL(this.Parent.label1.Caption)-1)

ENDIF

this.Parent.grid1.Refresh
this.Parent.grid1.SetFocus
```

EDIT Record button click code:

This code looks like the main program code as it launch a loading form also

```
SELECT orders

m.objectname = psgcon.getnewobject()
DO FORM orders_edit WITH m.objectname NAME &objectname

this.Parent.grid1.Refresh
this.Parent.grid1.SetFocus
```

2.5.6 6. Details form

6.a Loading form. Open form template.scx and save as orders_edit.scx

6.b Open orders_edit in form designer
Set the DataSession propery to 1, need to use tables opened previously by the orders_list form.

6.c Modify the INIT method of orders_edit as follows:

```
PARAMETERS ce_
THIS.Name = ce_

this.interface_class = 'orders'
this.interface_container = 'usrcontrol1'
this.getdata.addtable('products')
this.getdata.addtable('orderdetails')
this.getData.addparameter(orders.orderid) && a new method of
GETDATA, add a parameter to a table as filter
this.getData.gettables()
```

6.d Modify the LOAD_DATA method of orders_edit as follows:

```
PARAMETERS ce_
LOCAL class_, interface_, nial
FOR m.nial = 1 TO this.getdata.aliases.Count
            SELECT 0
            m.ext_file    = this.getdata.aliases.Item(m.nial) +
this.getdata.file_ext
            m.alias_name  =  this.getdata.aliases.Item(m.nial)

            jjj = "use work_\"+ m.ext_file + " ALIAS " +
m.alias_name + " SHARED"
            &JJJ
            this.remove_nulls()
            this.getdata.tables.Add(m.ext_file)
ENDFOR
SELECT orderdetails
REPLACE nrrec WITH RECNO() all && in order to display details
numbers
GO top
class_ = this.interface_class
SET CLASSLIB TO &class_
```

```
this.AddObject(this.interface_container,this.interface_container)
```

6.e Modify the UNLOAD method of orders_edit as follows:

```
SELECT 0
SELECT  products
USE
SELECT  orderdetails
USE
```

Manually close the used tables, the form DataSession property is set to 1.

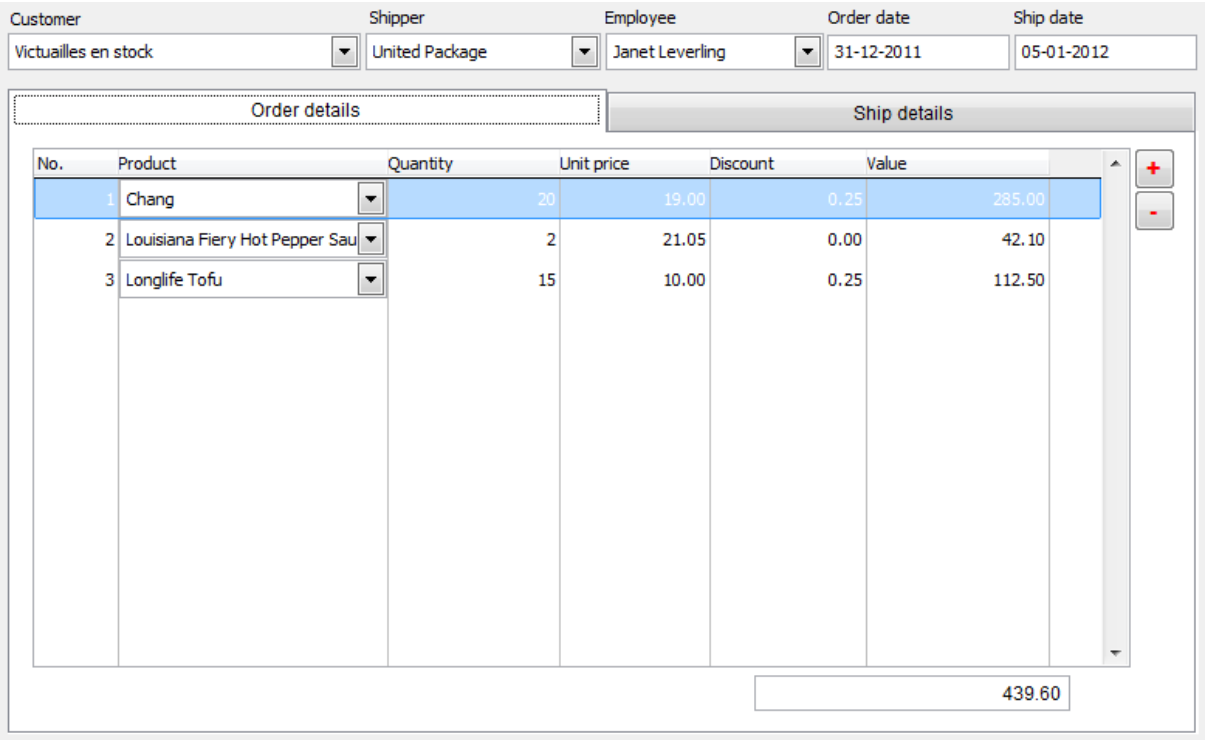
2.5.7 7. Details form container

Each control in this form is used to modify data therefore PSG base controls should be used.
If one control is not properly set (the controlsource property maps to inexistent data cursor the form will not load the control at run time.
tip: add only few controls at a time and then publish and open the client to check if it works, it will be easier to see the errors on time with fewer controls to check.

Each PSG control have additional properties needed to communicate with PSGCON.

KEY_FIELD - name of primary key field of the table (the key field should be found also into the local cursor)
SERVERTABLENAME - the name of the table to be modified on server database

tip: you can set the property at once for more controls that have the same table on control source, by selecting the controls and after that set the properties into properties window



Labels should be used from PSGLABEL class.
Each PSGLABEL should have the LID property to the caption ID from INTERNATIONAL table.
Check the SDK documentation for more details.

Here we have added code also on valid method of column two combo box in order to set also the unit price to details table.

```
DODEFAULT() && do what the control needs to do to work as espected

SELECT products
LOCATE FOR productid = this.Value
SELECT orderdetails

this.Parent.Parent.column3.text1.When
this.Parent.Parent.column3.text1.value = 0 && set quantity to 0 as we
have changed the product
this.Parent.Parent.column3.text1.valid

this.Parent.Parent.column4.text1.When
this.Parent.Parent.column4.text1.value =  products.unitprice && set the
price from products table
this.Parent.Parent.column4.text1.valid

this.Parent.Parent.column5.text1.When
this.Parent.Parent.column5.text1.value = 0 && set the value to 0
this.Parent.Parent.column5.text1.valid
```

ADD details record button

add a new control on first page PSGINSERT from PSGDATA

Click method code:

```
SELECT  orderdetails
m.detailid  =  RIGHT(SYS(2015),9)+RIGHT(psgcon.usrcode,3)
m.orderid  =  orders.orderid

this.Parent.psginsert1.insert('orderdetails','orderdetails','detailid')

REPLACE nrrec WITH RECNO() all
this.Parent.grid1.SetFocus
```

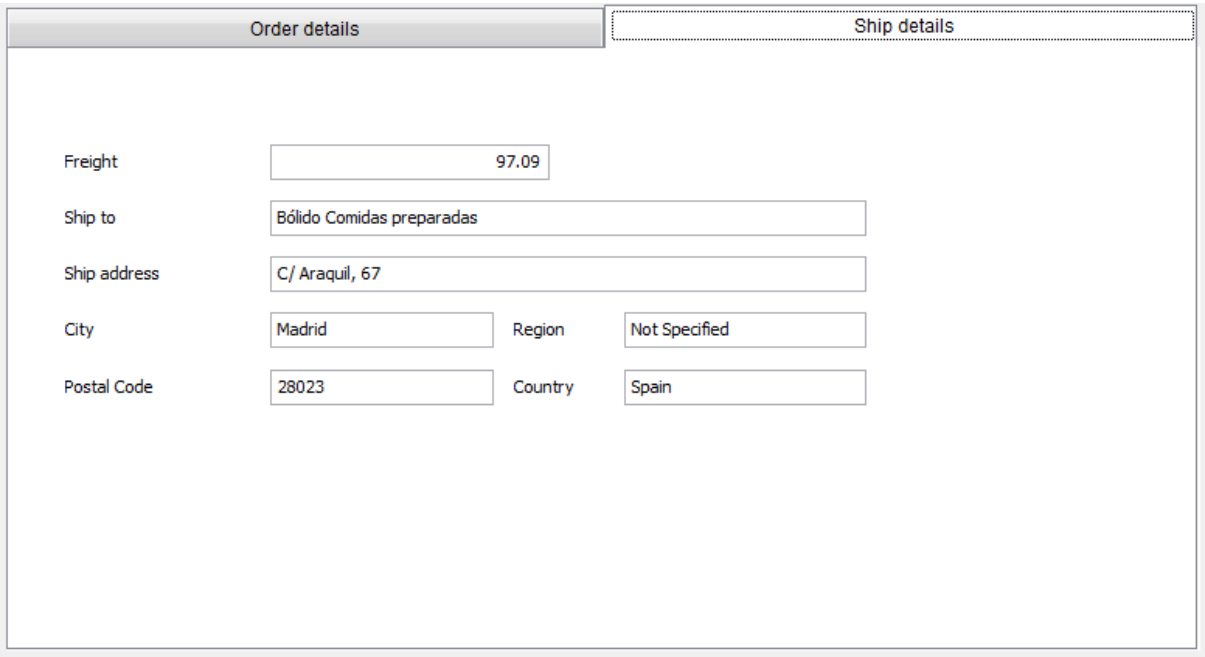
DELETE details record button

add a new control on first page PSGDELETE from PSGDATA

Click method code:

```
m.dltmsg  =  MESSAGEBOX('Delete  record',292,'Delete')
IF m.dltmsg = 6
      SELECT  orderdetails

this.Parent.psgdelete1.dlt('orderdetails','detailid',orderdetails.detail
id,'orderdetails')
      m.grec  =  RECNO()
      REPLACE nrrec WITH RECNO() all
      TRY
            GO  m.grec
      CATCH
      ENDTRY
ENDIF
this.Parent.grid1.SetFocus
```

Do the same for controls in Page2



2.5.8 Final

Feel free to improve the basic design of the demo project as you want.

Help and technical support from PSGSDK is available by email, please check the psgsdk.com .

Other articles and information could be found on the website.