# Introducing Synergy .NET – Demos

## Demo 1 – Creating a Simple Desktop App

Make sure that the four code snippets are in the toolbox Make sure existing files are not read only

* Start VS and create a **New Synergy WinForms project** (Framework 4)
  + Mention WPF probably way to go, but most of us recognize WinForms more
  + Name the project **ContactBrowser**
* Examine default content
  + **Form1.dbl** is a windows form, and has a .designer.dbl file
  + **program.dbl** is the entry point which launches the form
  + **References** folder has default references, like other languages add other references if required.
* Doing Synergy development, so I need some environment variables
  + RPSMFIL C:\SPC2010\_STEVE\04\_MDU\rps\rpsmain.ism
  + RPSTFIL C:\SPC2010\_STEVE\04\_MDU\rps\rpstext.ism
  + DAT C:\SPC2010\_STEVE\04\_MDU\dat
  + MANUALLY SAVE (CTRL\_SHIFT\_S) THEN CLOSE AND RE-OPEN THE SOLUTION
  + Mention: Could also use system environment, or synergy.ini
* Resize the form and set some form properties.
  + Text Customer Contact Browser
  + StartPosition Center Screen
  + MANUALLY SAVE (CTRL\_SHIFT\_S)
* Run the application
  + Like other languages, creating a new project gives us the shell for a new windows app
  + Every time we do a Run, a Build will happen if needed
* Build the UI
  + Menus & Toolbars
    - Add a MenuStrip with a File / Exit option
    - Add a StatusStrip
  + Containers
    - Add a Panel and set to Dock Top
  + Common Controls
    - Add a Label (text Customer)
    - Add a ComboBoxBox (cboCustomer), make it bigger and set DropDownStyle to DropDownList
    - Resize panel to look good.
  + Data
    - Add a DataGridView
    - In SmartTag: disable Add, Edit and Delete, and click Dock In Parent Container
    - **Id grdContacts**
    - AutoSizeColumnsMode Fill
    - SelectionMode FullRowSelect
  + MANUALLY SAVE (CTRL\_SHIFT\_S)
* Run the app
  + UI is complete, but no data, and exit menu doesn’t work
* Double-click on form dragbar to insert the Form1\_Load method and look at the code
  + SLOW DOWN!!!! Explain the \_Load method
* Go back to the designer and double-click the Exit menu option to insert the event handler
  + Add code to do a this.Close() – notice IntelliSense.
  + MANUALLY SAVE (CTRL\_SHIFT\_S)
* Run the app
  + Now the exit option works
* Add a class that represents a Synergy record (customer)
  + Add > New Item > Synergy > Class and name the class Customer
    - Tack about the new item dialog, different content for different project types
    - Notice new class placed into apps default namespace
  + We’re going to pass in a record via a structure parameter, so in the **NAMESPACE**:
    - **.include “CUSTOMER” repository, structure=”strCustomer”, end**
  + Add constructor
    - ~~public method <tab><tab> Customer <esc> and take off ,void~~
    - Going to use a SNIPPET, explain
    - **ctor <tab> <tab> and set name to Customer**
  + Add constructor parameter
    - **required in a\_customer, strCustomer**
      * **INTELLISENSE NOT PICKING UP ON NEW STRUCTURE**
  + Add a private field to store the contact
    - **private mCustomer, strCustomer**
      * **INTELLISENSE NOT PICKING UP ON NEW STRUCTURE**
  + Store the passed in contact
    - **mCustomer = a\_customer**
  + Use “prop” to add one property, CustomerId, int, mCustomer.customer\_id
    - **INTELLISENSE NOT PICKING UP ON NEW STRUCTURE**
  + Use “prop” to add one property, CompanyName, string, mCustomer.name
    - **INTELLISENSE NOT PICKING UP ON NEW STRUCTURE**
  + MANUALLY SAVE (CTRL\_SHIFT\_S)
* Add second class, contact
  + Ways of adding things to a project
    - New Item creates a new file
    - Existing item COPIES an existing file into the project
    - Add Reference... adds a REFERENCE to an existing file (in situ)
  + Right-click project > Add > Existing Item and pick the Contact.dbl file
  + Review file – similar to class we just added
* Add two new existing functions (GetCustomers.dbl and GetCustomerContacts.dbl)
  + Right-click project > Add > Existing Item and pick both files
  + Review routines – except **return** parameter type ... typical Synergy code
* BUILD – make sure we’re OK
* Hook GetCustomers routine in to load customers into combo box
  + Go to Form1\_Load
  + In data division,

**record <tab> <tab> (Or drag GetCustomersData)**

**msg, string**

**customers, @List <ESC> <Customer>  
endrecord**

* + Add code to call routine **(Or Drag GetCustomersCode)**

**ifthenelse <tab> <tab>**

if (**%GetCustomers(customers,msg)**) then

begin

**cboCustomer.DataSource = customers**

**cboCustomer.ValueMember = "CustomerId"**

**cboCustomer.DisplayMember = "CompanyName"**end

else

begin

**MessageBox.Show(msg)** – We can use framework classes!  
end

* + MANUALLY SAVE (CTRL\_SHIFT\_S)
* Run the app – now the customer combo is loaded
* When customer is picked, load contacts
  + Go to form designer
  + Double-click on Combo to insert default event handler
  + In data division **(Or drag GetContactsData)**

**record <tab> <tab>**

**cust ,@Customer**

**contacts ,@List <esc> <Contact>**

**msg ,string**

**endrecord**

* + In procedure division **(Or drag GetContactsCode)**

**cust = (Customer)cboCustomer.SelectedItem**

**if %GetCustomerContacts(cust.CustomerId,contacts,msg) then**

**grdContacts.DataSource = contacts**

**else**

**MessageBox.Show(msg)**

* Debugger
  + Set a break in the GetCustomers function
  + Set a break in the GetCustomerContacts function
  + Make sure we’re in a Debug configuration
  + Run the app to break in GetCustomers
    - Review the Locals window – local data
    - Hover over the message field – MAY NOT WORK!!!
    - Do some steps, see the data changing in the Locals window
    - Do a GO, breaks in the GetCustomerContacts routine
* Demo code converter

## Demo 2 – Interop Projects

* Demo the original client application using xfNetLink .NET and xfServerPlus
* Copy the xfNetLinkClient folder to interopClient and rename the solution to interopClient
* Open the new solution with VS, build and run – exact copy of original
* Remove references to SynergyComponent and xfnlnet
* Build – VERY BROKEN! – check out form1.dbl code – red squigglies!
* Add a new Interop project called **InteropComponent**
  + File > Add > New Project > Synergy >Interop
* Explain SynergyRoutines.dbl with helper methods and conversion assist routines – close it
* Need to make sure the namespace we use is the same as used by xfNetLink, project properties and set default namespace to spc2010.
* Methods use repository structures, so set
  + RPSMFIL C:\SPC2010\_STEVE\04\_MDU\rps\rpsmain.ism
  + RPSTFIL C:\SPC2010\_STEVE\04\_MDU\rps\rpstext.ism
  + Save, close and re-open project
* Add our Synergy method sources
  + Add > Existing Item, pick all method sources from C:\SPC2010\_STEVE\04\_MDU\src\SynergyServer
  + Edit some methods to see code, attributes, doc comments etc.
  + These are the same methods we were looking at in the MDU / Attributes session earlier
* Build Interop project – notice “Generated Code” folder and wrappers – explain
* Now add reference to client project
* Build project and run
  + “File not found”
  + Issue is xfServerPlus had the environment with logical names used by the code, and now the client is accessing the data directly, so needs the logicals.
* Add C:\SPC2010\_STEVE\07\_NET\INTEROP\SetLogicals.dbl to the InteropComponent project and examine.
* Rebuild the Interop project
* Edit the client form code
  + Mention there may be code no longer required (connect, disconnect, etc)
  + Add a svr.SetLogicals() to the end of the constructor.
* Run!

## Demo 3 – Using Relational Data

* Using Synergy SQL Connection API
* Using Data Provider for SQL Server