

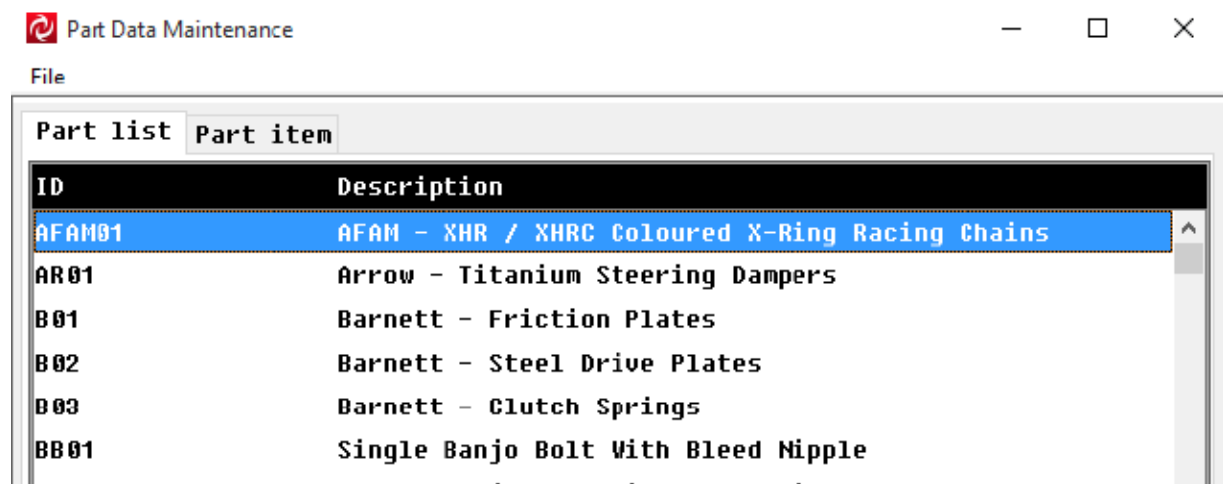
Migrating from UI Toolkit to WPF

Introduction.

The development will be performed using both Synergy Workbench and Microsoft Visual Studio. To begin the workshop please open the Synergy Workbench and locate the UIToolkit workspace:

- Open Workbench
- Select Project->Open Workspace...
- Navigate to Documents\Synergy Tutorials\UIToolkitToWPF
- Select the UIToolkit.vpw file and click the Open button.

The project should build (using Build->Build or CTRL + M). The project should run (using Build->Execute or CTRL + F5)



Here you have the following options:

- From the **File** menu column the **New** option will take you into edit mode and allow you to create a new item in the database.
- The **Exit** option on the **File** menu will terminate the program.
- Selecting any item from the list will bring the item into edit mode.

Migrating from UI Toolkit to WPF

In edit mode:

Part Data Maintenance

File

Part list Part item

Part ID	BBL01
Group ID	BRAKE
Supplier ID	GOODRIDGE
Description	Goodridge - Stainless Steel Braided Brake Lines
Quantity	20
Cost price	83.18

Save Cancel Delete

The options available here include:

- The **Save** button will validate the entered details and save the change to the database.
- The **Cancel** button will return you to the list view.
- The **Delete** button will delete the currently selected item.

Once you are familiar with the program options, close the program.

Within Synergy Workbench on the **Tools** menu column select the **OS Shell** entry to open a command window. Within the command window execute the command **SYNCKINI** to edit the synuser.ini file;


- When prompted to edit the **synergy.ini** file answer "N".
- If you are prompted to create the **synuser.ini** file answer "Y".
- When prompted to edit the **synuser.ini** file answer "Y".
- If you were required to create the **synuser.ini** file add the required **[synergy]** section.
- Within the **[synergy]** section add a new variable named **SYMPHONY_RUNNING_WPF** with a value of **True**.

Migrating from UI Toolkit to WPF

[synergy]

SYMPHONY_RUNNING_WPF=True

Close and save the changes to the **synuser.ini** file. Close the command window.

Return to the Synergy Workbench and re-execute the program. This time the new WPF user interface will be visible. Nothing within the application will currently work – menu entries and toolbar items and there is nothing displayed in the grid. Close the program by **Close** control (). The program will display the standard Synergy “stop” message which will be corrected later.

Migrating from UI Toolkit to WPF

Menu & Toolbar Processing.

To enable menu and toolbar processing and communicate the selected options back to the host Traditional Synergy program we need to perform the following steps:

- Open Visual Studio
- Select File->Open->Project/Solution...
- Navigate to Documents\Synergy Tutorials\UIToolkitToWPF
- Select the WPFToolkit.sln solution file and click the Open button.

The first step is to define the available commands

- Under the ViewModel folder locate the PartMaintenanceViewModel.dbc file. Double click the file to bring it into edit mode.
- Locate the **createCommands** private method and between the `;;#SNIPPET+##;;#SNIPPET-#` snippet markers add the following code to define the available menu/toolbar commands:

```
;;#SNIPPET+##
mCommands = new MenuController()

;;define the required "commands". These can be existing menu and toolbar commands
;;and window/list commands

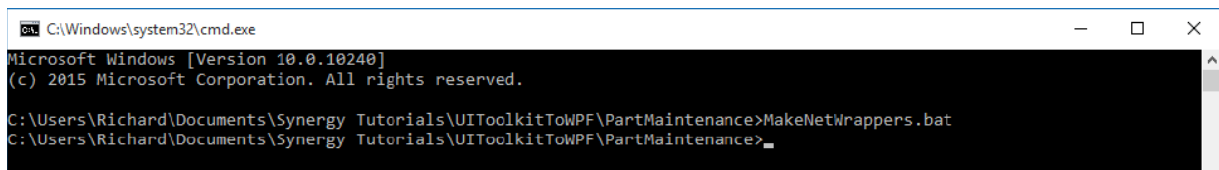
;;these commands map to the original menu entries
mCommands.LoadCommand("F_NEW", doCommandExecuted, Visibility.Visible)
mCommands.LoadCommand("F_EXIT", doCommandExecuted, Visibility.Visible)

;;these commands map to the buttons on the original toolkit UI
mCommands.LoadCommand("B_SAVE", doCommandExecuted, Visibility.Visible)
mCommands.LoadCommand("B_CANCEL", doCommandExecuted, Visibility.Visible)
mCommands.LoadCommand("B_DELETE", doCommandExecuted, Visibility.Visible)

;;#SNIPPET-#
```

Build the Visual Studio solution (Build->Build) and correct any errors.

Return to the Synergy Workbench and select the **OS Shell** option from the **Tools** menu. Inside the command window ensure you are in the **PartMaintenance** folder and execute the **MakeNetWrappers.bat** command script:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Richard\Documents\Synergy Tutorials\UIToolkitToWPF\PartMaintenance>MakeNetWrappers.bat
C:\Users\Richard\Documents\Synergy Tutorials\UIToolkitToWPF\PartMaintenance>
```

Close the command window and return to the Synergy Workbench and from within the **Projects** view locate and select the **MainLine.dbl** code to bring it into the edit window.

Migrating from UI Toolkit to WPF

We need to add the required code to enable access to the program data.

- Between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers add the following code:

```
;;#SNIPPET+##
prgVM.ProgramDataHandle = %mem_proc(DM_REG + DM_STATIC,
&      ^size(program_data), ^addr(program_data))
prgVM.ProgramDataSize = ^size(program_data)
;;#SNIPPET-#
```

From within the **Projects** view locate and select the **BoilerPlate.dbc** code to bring it into the edit window.

We need to add the required code to handle when commands are executed.

- Locate the **static method BoilerPlate** code.
- Between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers add the following code:

```
;;#SNIPPET+##
addHandler(mAppVM.TKMenuSignal, doTKMenuSignal)
;;#SNIPPET-#
```

- Locate the **private static method doTKMenuSignal** code.
- Between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers add the following code:

```
;;#SNIPPET-#
g_entnam = %atrim(menuName)
g_select = 1
g_setsts = 0

using g_entnam select
("STARTUP"),
begin
    ProgramState = ProgramStates.StartUp
    xsubr("PartMaint", program_data)

    ProgramState = ProgramStates.StartUp
    xsubr("Part_list", program_data)
end
("SHUTDOWN"),
begin
    ProgramState = ProgramStates.Shutdown
    xsubr("PartMaint", program_data)
end
("F_EXIT"),
begin
    using mAppVM.CurrentStateName select
    ("InputViewState"),
    begin
        ProgramState = ProgramStates.MenuSignal
        xsubr("Part_input", program_data)
    end
end
```

Migrating from UI Toolkit to WPF

```
("MainViewState"),
begin
    ProgramState = ProgramStates.MenuSignal
    xsubr("Part_list", program_data)
end
endusing
ProgramState = ProgramStates.Shutdown
xsubr("PartMaint", program_data)
end
("F_NEW"),
begin
    ProgramState = ProgramStates.MenuSignal
    xsubr("Part_list", 0, program_data)

    ProgramState = ProgramStates.StartUp
    xsubr("Part_input", 0, program_data)

    mAppVM.MoveToState("InputViewState")
end
("F_MODIFY"),
begin
    ProgramState = ProgramStates.MenuSignal
    xsubr("Part_list", 0, program_data)

    ProgramState = ProgramStates.StartUp
    xsubr("Part_input", 0, program_data)

    mAppVM.MoveToState("InputViewState")
end
("B_SAVE", "B_CANCEL", "B_DELETE"),
begin
    ProgramState = ProgramStates.MenuSignal
    xsubr("Part_input", 0, program_data)

    ;;we only complete the state if the part data object is valid
    if (g_entnam != "B_SAVE" ||
&      (g_entnam == "B_SAVE" && mAppVM.PartItem.IsDataValid == true))
    begin
        mAppVM.StateCompleted()

        ProgramState = ProgramStates.StartUp
        xsubr("Part_list", 0, program_data)
    end
end
endusing
;;#SNIPPET-#
```

Migrating from UI Toolkit to WPF

The program makes changes to the availability of commands in code which needs to be updated.

- In Synergy Workbench, from within the **Projects** view locate and select the **part_list.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+CONTROL OPTIONS#` and `;;#SNIPPET-CONTROL OPTIONS#` snippet markers add the following code:

```
;;#SNIPPET+CONTROL OPTIONS#
prgVM.Commands.EnableCommand("F_NEW")
prgVM.Commands.DisableCommand("B_SAVE")
prgVM.Commands.DisableCommand("B_CANCEL")
prgVM.Commands.DisableCommand("B_DELETE")
;;#SNIPPET-CONTROL OPTIONS#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **part_input.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+CONTROL OPTIONS#` and `;;#SNIPPET-CONTROL OPTIONS#` snippet markers add the following code:

```
;;#SNIPPET+CONTROL OPTIONS#
prgVM.Commands.DisableCommand("F_NEW")
prgVM.Commands.EnableCommand("B_SAVE")
prgVM.Commands.EnableCommand("B_CANCEL")
prgVM.Commands.EnableCommand("B_DELETE")
;;#SNIPPET-CONTROL OPTIONS#
```

Build the Workbench project (using Build->Build or CTRL + M). The project should run (using Build->Execute or CTRL + F5). The toolbar buttons will appear greyed out and not responsive as they are not applicable to that stage of the program. The Exit button should close the application.

Migrating from UI Toolkit to WPF

List Processing and List Load Methods.

To enable list processing the WPF code will call back into the host program to execute the existing list load methods:

- Return to Visual Studio
- Locate the **public method LoadPartList** and between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers add the following code to execute the host program's list load method:

```
;;#SNIPPET+#  
performLoad(mPartList, ^typeof(Part_Data), "", "part_load",  
& Part_Data.RPSStructureSize, mProgramDataHandle, mProgramDataSize)  
;;#SNIPPET-#
```

- Locate the **public method LoadGroupList** and between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers add the following code to execute the host program's list load method:

```
;;#SNIPPET+#  
performLoad(mGroupList, ^typeof(Group_Data), "", "Drill_Group_Id_load",  
& Group_Data.RPSStructureSize, mProgramDataHandle, mProgramDataSize)  
;;#SNIPPET-#
```

- Locate the **public method LoadSupplierList** and between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers add the following code to execute the host program's list load method:

```
;;#SNIPPET+#  
performLoad(mSupplierList, ^typeof(Supplier_Data), "", "Drill_Supplier_Id_load",  
& Supplier_Data.RPSStructureSize, mProgramDataHandle, mProgramDataSize)  
;;#SNIPPET-#
```

Build the Visual Studio solution (Build->Build) and correct any errors.

Return to the Synergy Workbench and select the **OS Shell** option from the **Tools** menu. Inside the command window ensure you are in the **PartMaintenance** folder and execute the **MakeNetWrappers.bat** command script. Close the command window.

Return to the Synergy Workbench and from within the **Projects** view locate and select the **BoilerPlate.dbc** code to bring it into the edit window.

We need to add the required code to handle when commands are executed.

- Locate the **static method BoilerPlate** code.
- Between the `;;#SNIPPET+##;#SNIPPET-#` snippet markers, after any existing code, add the following code:

```
addHandler(mAppVM.LookupLoad, doListLoadMethod)
```


Migrating from UI Toolkit to WPF

So the resulting code should now be:

```
;;#SNIPPET+#
addHandler(mAppVM.TKMenuSignal, doTKMenuSignal)
addHandler(mAppVM.LookupLoad, doListLoadMethod)
;;#SNIPPET-#
```

- Locate the **private static method doListLoadMethod** code.
- Between the `;;#SNIPPET+#`; `;;#SNIPPET-#` snippet markers add the following code:

```
;;#SNIPPET+#
if (!mListMemPnt) then
begin
    ;;need memory allocating
    mListMemPnt = %mem_proc(DM_ALLOC.bor.DM_STATIC, loadDataSize)
end
else
begin
    if (mListMemSize != loadDataSize)
    begin
        ;;need memory resizing
        mListMemPnt = %mem_proc(DM_RESIZ, loadDataSize, mListMemPnt)
    end
end

clear ^m(memStr(1:loadDataSize), mListMemPnt)

if (!itemNumber)
begin
    ;;this is the first time in
    ;;call the host and initialize
    ProgramState = ProgramStates.StartUp
    if (hostMethodName.Length != 0)
        xsubr(hostMethodName)
    itemNumber = 1
    mreturn
end

;;call the requested load method
ProgramState = ProgramStates.ListLoad
xsubr(loadMethodName, -1, listRequest = D_LLOADBOT,
&      ^m(memStr(1:loadDataSize), mListMemPnt), -1,, itemNumber, program_data)

;;end of list load??
if (listRequest == D_LEOF) then
begin
    endOfList = true
end
else
begin
    endOfList = false
    listData = ^m(memStr(1:loadDataSize), mListMemPnt)
end
;;#SNIPPET-#
```

Migrating from UI Toolkit to WPF

- In Synergy Workbench, from within the **Projects** view locate and select the **part_load.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+#`; `;;#SNIPPET-#` snippet markers modify the following code:

```
;;#SNIPPET+#  
if (a_request != D_LEOF)  
    if (DoingTK)  
        i_display(a_inpid,, a_data)  
;;#SNIPPET-#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **Drill_Group_Id.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+#`; `;;#SNIPPET-#` snippet markers modify the following code:

```
;;#SNIPPET+#  
if (a_request != D_LEOF)  
    if (DoingTK)  
        i_display(a_inpid,, a_data)  
;;#SNIPPET-#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **Drill_Supplier_Id.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+#`; `;;#SNIPPET-#` snippet markers modify the following code:

```
;;#SNIPPET+#  
if (a_request != D_LEOF)  
    if (DoingTK)  
        i_display(a_inpid,, a_data)  
;;#SNIPPET-#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **part_list.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+SELECTION#`; `;;#SNIPPET-SELECTION#` snippet markers add the following code:

```
;;#SNIPPET+SELECTION#  
if (DoingWPF) part_data = prgVM.PartListItem.SynergyRecord  
;;#SNIPPET-SELECTION#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **part_input.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+LIST RESET#`; `;;#SNIPPET-LIST RESET#` snippet markers add the following code:

```
;;#SNIPPET+LIST RESET#  
prgVM.LoadPartList()
```

Migrating from UI Toolkit to WPF

`;;#SNIPPET-LIST RESET#`

Please Note there are **two** instances of the `#SNIPPET+LIST RESET` snippet in the **part_input.dbl** code. Both need to be modified as above.

Build the Workbench project (using Build->Build or CTRL + M). The project should run (using Build->Execute or CTRL + F5).

The program will now allow you to select items from the initial list and the field drill methods will work and display items to select. When you select data from the list the details will not appear in the input form.

Migrating from UI Toolkit to WPF

Input Processing.

When records are selected or new items being processed the host program requires the ability to display data through to the new UI and initialize fields. The program also needs to be able to handle input set processing:

- In Synergy Workbench, from within the **Projects** view locate and select the **part_input.dbl** code to bring it into the edit window.
- Locate the `;;#SNIPPET+DISPLAY#` `#SNIPPET-DISPLAY#` snippet markers and add the following code:

```
;;#SNIPPET+DISPLAY#
prgVM.DefineInputSet("modify")
prgVM.PartItem.SynergyRecord = part_data
prgVM.PartItem.FieldValidDetails("ID")
prgVM.PartItem.FieldValidDetails("GROUPID")
prgVM.PartItem.FieldValidDetails("SUPPLIERID")
;;#SNIPPET-DISPLAY#
```

- Locate the `;;#SNIPPET+INITIALIZE#` `#SNIPPET-INITIALIZE#` snippet markers and add the following code to create the change method event:

```
;;#SNIPPET+INITIALIZE#
prgVM.DefineInputSet("new")
prgVM.PartItem.InitData()
part_data = prgVM.PartItem.SynergyRecord
call validate_record
;;#SNIPPET-INITIALIZE#
```

Build the Workbench project (using Build->Build or CTRL + M). The project should run (using Build->Execute or CTRL + F5).

When you select an entry from the list the details will be correctly displayed when in amend mode. Selecting to create a “New” item from the File menu will correctly display a null input form.

Migrating from UI Toolkit to WPF

Field Validation and Change Method Processing.

To enable field validation the WPF code will call back into the host program to execute the existing field level change methods:

- Return to Visual Studio
- Locate the `;;#SNIPPET+CHANGE#` `;;#SNIPPET-CHANGE#` snippet markers and add the following code to create the change method event:

```
;;#SNIPPET+CHANGE#
public delegate ChangeMethodEventHandler ,void
    in rpsName ,string
    inout messageData ,@DataChangedMessage
enddelegate

public event ChangeMethod ,@ChangeMethodEventHandler

private method doChangeMethod_part ,void
    inout messageData ,@DataChangedMessage
endparams
proc
    raiseevent(ChangeMethod, Part_Data.RPSStructureName, messageData)
endmethod
;;#SNIPPET-CHANGE#
```

- Locate the `bindHandlers` private method and add the following code to execute the host program's list load method:

```
;;#SNIPPET+BIND CHANGE#
mPartItem.ExecuteChangeMethod += doChangeMethod_part
;;#SNIPPET-BIND CHANGE#
```

Build the Visual Studio solution (Build->Build) and correct any errors.

Return to the Synergy Workbench and select the **OS Shell** option from the **Tools** menu. Inside the command window ensure you are in the **PartMaintenance** folder and execute the **MakeNetWrappers.bat** command script. Close the command window.

Return to the Synergy Workbench and from within the **Projects** view locate and select the **BoilerPlate.dbc** code to bring it into the edit window.

We need to add the required code to handle when commands are executed.

- Locate the **static BoilerPlate** method.
- Between the `;;#SNIPPET+#;` `;;#SNIPPET-#` markers, after any existing code, add the following code:

```
addHandler(mAppVM.ChangeMethod, doChangeMethod)
```

Migrating from UI Toolkit to WPF

So the resulting code should now be:

```
;;#SNIPPET+#  
addHandler(mAppVM.TKMenuSignal, doTKMenuSignal)  
addHandler(mAppVM.LookupLoad, doListLoadMethod)  
addHandler(mAppVM.ChangeMethod, doChangeMethod)  
;;#SNIPPET-#
```

- Locate the **private static doChangeMethod** method.
- Between the `;;#SNIPPET+#`; `;;#SNIPPET-#` snippet markers add the following code:

```
;;#SNIPPET+#  
program_data = ^m(memStr(1:mAppVM.ProgramDataSize), mAppVM.ProgramDataHandle)  
program_data.part_data = mAppVM.PartItem.SynergyRecord  
  
methodToCall = messageData.ChangeMethodName  
dataStart = messageData.FieldStartPos  
dataEnd = messageData.FieldLength  
  
result = xsubr(methodToCall, part_data(dataStart:dataEnd),  
& part_data(dataStart:dataEnd), pending = 0, , part_data, program_data)  
  
^m(memStr(1:mAppVM.ProgramDataSize), mAppVM.ProgramDataHandle) = program_data  
;;#SNIPPET-#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **Change_Group_Id.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+CHANGE#`; `;;#SNIPPET-CHANGE#` snippet markers add the following code:

```
;;#SNIPPET+CHANGE#  
if (DoingWPF) prgVM.PartItem.FieldValidDetails("GROUPID")  
;;#SNIPPET-CHANGE#
```

- Between the `;;#SNIPPET+INVALID#`; `;;#SNIPPET-INVALID#` snippet markers add the following code:

```
;;#SNIPPET+INVALID#  
prgVM.PartItem.FieldErrorDetails("GROUPID", "Group details not found")  
;;#SNIPPET-INVALID#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **Change_Supplier_Id.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+CHANGE#`; `;;#SNIPPET-CHANGE#` snippet markers add the following code:

```
;;#SNIPPET+CHANGE#  
if (DoingWPF) prgVM.PartItem.FieldValidDetails("SUPPLIERID")  
;;#SNIPPET-CHANGE#
```

- Between the `;;#SNIPPET+INVALID#`; `;;#SNIPPET-INVALID#` snippet markers add the following code:

Migrating from UI Toolkit to WPF

```
;;#SNIPPET+INVALID#
prgVM.PartItem.FieldErrorDetails("SUPPLIERID", "Supplier details not found")
;;#SNIPPET-INVALID#
```

- In Synergy Workbench, from within the **Projects** view locate and select the **part_input.dbl** code to bring it into the edit window.
- Between the `;;#SNIPPET+VALIDATION#` and `;;#SNIPPET-VALIDATION#` snippet markers add the following code:

```
;;#SNIPPET+VALIDATION#
data idDataInError          ,i4    ,0
data groupDataInError       ,i4    ,0
data supplierDataInError    ,i4    ,0

prgVM.PartItem.FieldValidDetails("ID")

if (!%trimz(part_data.id))
begin
    idDataInError = 1
    prgVM.PartItem.FieldErrorDetails("ID", "Part ID required")
end

groupDataInError = %Change_Group_Id(part_data.groupid, part_data.groupid,
&    groupDataInError = D_OK, , part_data, program_data)

supplierDataInError = %Change_Supplier_Id(part_data.supplierid, part_data.supplierid,
&    supplierDataInError = D_OK, , part_data, program_data)

if (idDataInError || groupDataInError || supplierDataInError)
    dataInError = 1
;;#SNIPPET-VALIDATION#
```

- In the **part_input.dbl** code locate the **save_record** label.
- Between the `;;#SNIPPET+SAVE SUCCESS#` and `;;#SNIPPET-SAVE SUCCESS#` snippet markers add the following code:

```
;;#SNIPPET+SAVE SUCCESS#
prgVM.StatusBarText = "Last operation SAVE was successful"
;;#SNIPPET-SAVE SUCCESS#
```

- Between the `;;#SNIPPET+UPDATE SUCCESS#` and `;;#SNIPPET-UPDATE SUCCESS#` snippet markers add the following code:

```
;;#SNIPPET+UPDATE SUCCESS#
prgVM.StatusBarText = "Last operation UPDATE was successful"
;;#SNIPPET-UPDATE SUCCESS#
```

- Between the `;;#SNIPPET+FAILURE#` and `;;#SNIPPET-FAILURE#` snippet markers add the following code:

```
;;#SNIPPET+FAILURE#
prgVM.StatusBarText = "Last operation SAVE/UPDATE was unsuccessful: " + e.Message
;;#SNIPPET-FAILURE#
```

Migrating from UI Toolkit to WPF

- In the **part_input.dbl** code locate the **delete_record** label.
- Between the `;;#SNIPPET+DELETE SUCCESS#`; `;;#SNIPPET-DELETE SUCCESS#` snippet markers add the following code:

```
;;#SNIPPET+DELETE SUCCESS#  
prgVM.StatusBarText = "Last operation DELETE was successful"  
;;#SNIPPET-DELETE SUCCESS#
```

- Between the `;;#SNIPPET+DELETE FAILURE#`; `;;#SNIPPET-DELETE FAILURE#` snippet markers add the following code:

```
;;#SNIPPET+DELETE FAILURE#  
prgVM.StatusBarText = "Last operation DELETE was unsuccessful: " + e.Message  
;;#SNIPPET-DELETE FAILURE#
```

Build the Workbench project (using Build->Build or CTRL + M). The project should run (using Build->Execute or CTRL + F5).

Creating a new item from the File menu and the fields provide indication of required or invalid data. Changing the fields will check that the data is valid by executing the original change methods.

Saving or deleting record will display the appropriate messages to the applications status bar.