Solutions

Unit: User Interface API

Topic: Basics

1-1 Implement a connection to a running SAP Business One application.

1-1-1 Create a new Visual Studio project for a windowless add-on application and add a reference to the SAP Business One DI API COM library and UI API COM library.

1-1-2 Define the variables you need for a connection to a running SAP Business One application.

```
Private WithEvents SBO_Application As SAPbouiCOM.Application

Dim SboGuiApi As SAPbouiCOM.SboGuiApi

Dim sConnectionString As String
```

1-1-3 Connect to the SAP Business One SboGuiApi and get a handle to the running application.

```
SboGuiApi = New SAPbouiCOM.SboGuiApi

sConnectionString = Environment.GetCommandLineArgs.GetValue(1)

SboGuiApi.Connect(sConnectionString)

SBO_Application = SboGuiApi.GetApplication()
```

1-2 Display a MessageBox within SAP Business One.

1-2-1 The method to display a MessageBox has several optional parameters. Check them out.

```
SBO_Application.MessageBox("Connected", 1, "Continue", "Cancel")
```

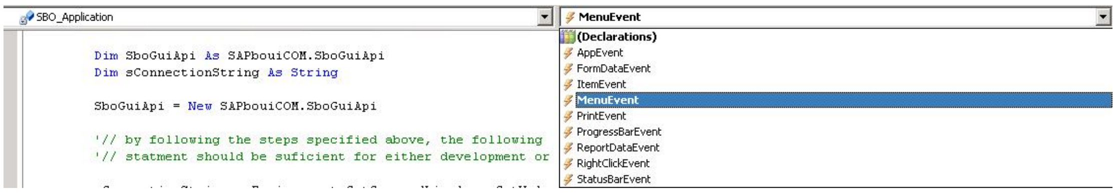
1-3 Use the Single-Sign-On feature (and/or the Multiple Add-On feature) to connect to DI API as well.

Private oCompany As SAPbobsCOM.Company

oCompany = New SAPbobsCOM.Company

oCompany = SBO_Application.Company.GetDICompany()

1-4 Define the AppEvent handler – and implement the handling of these events (which are mandatory to be handled).



```
        Dim SboGuiApi As SAPbouiCOM.SboGuiApi
        Dim sConnectionString As String

        SboGuiApi = New SAPbouiCOM.SboGuiApi

        '// by following the steps specified above, the following
        '// statment should be suficient for either development or
```

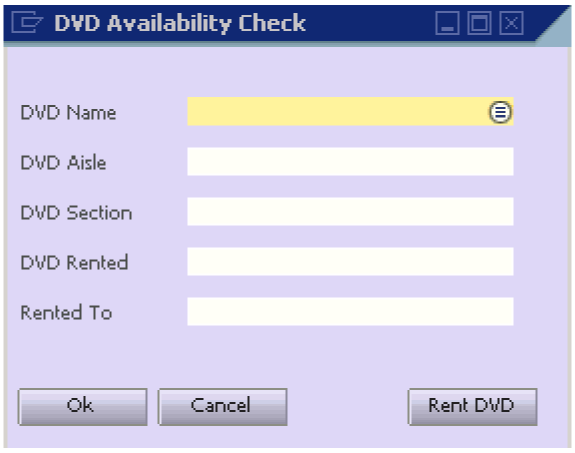
Solutions can be found in the SDK Help Center documentation and SDK samples (in the SDK Folder – see Appendix “SDK Installations” for more information),
COM UI / VB .NET / 01.HelloWorld
COM UI / VB .NET / 02.CatchingEvents
COM UI DI / VB .NET / Hello World

Solutions

Unit: User Interface API

Topic: Creating Forms

2-1 Create a new form within SAP Business One. The form should contain the following items:



Some example code:

Form Creation:

```
        creationPackage =
SBO_Application.CreateObject(SAPbouiCOM.BoCreatableObjectType.cot_FormCreationPa
rams)


        creationPackage.UniqueID = "TB1_DVDAvailability"

        creationPackage.FormType = "TB1_DVDAvailability"

creationPackage.ObjectType = "TB1_DVDAvail"  ‘link form to your UDO


        oForm = SBO_Application.Forms.AddEx(creationPackage)


        oForm.Title = "DVD Availability Check"

        oForm.Left = 336

        oForm.ClientWidth = 280

        oForm.Top = 44

        oForm.ClientHeight = 200
```

Button creation:

```
        oItem = oForm.Items.Add("RentDVD",
SAPbouiCOM.BoFormItemTypes.it_BUTTON)
        oItem.Left = 200
        oItem.Width = 65
        oItem.Top = 170
        oItem.Height = 19

        oButton = oItem.Specific
        oButton.Caption = "Rent DVD"
```

Choose from List

```
Dim oCFLs As SAPbouiCOM.ChooseFromListCollection
oCFLs = oForm.ChooseFromLists

Dim oCFL As SAPbouiCOM.ChooseFromList
Dim oCFLCreationParams As SAPbouiCOM.ChooseFromListCreationParams
oCFLCreationParams =
SBO_Application.CreateObject(SAPbouiCOM.BoCreatableObjectType.cot_ChooseFromListCreationP
arams)

oCFLCreationParams.ObjectType = "TB1_DVDAvail" 'Note – this is the Code you gave in the wizard
when you registgered the UDO  for TB1_DVD in the UDO exercises
oCFLCreationParams.UniqueID = "DVDCFL"
   oCFL = oCFLs.Add(oCFLCreationParams)
```

EditText Creation

```
oItem = oForm.Items.Add("DVDNameT", SAPbouiCOM.BoFormItemTypes.it_EDIT)
oItem.Left = 90
oItem.Width = 163
oItem.Top = 25
oItem.Height = 14
oItem.LinkTo = "DVDNameL"  'link it to the associated Static

oEditText = oItem.Specific
oEditText.DataBind.SetBound(True, "", "DVDName")
oEditText.ChooseFromListUID = "DVDCFL"
```

2-2 Enhance your program so that the form will be saved as an XML file.

Firstly add a reference to Microsoft XML – this references .NET's System.Xml library

```
oXmlDoc = New Xml.XmlDocument

sXmlString = oForm.GetAsXML

oXmlDoc.LoadXml(sXmlString)

oXmlDoc.Save("File location \" & "DVDAvailability.xml")
```

2-3 Change your program. The form should now be loaded from the XML file you have created in the last step. Display the form in the SAP Business One window.

This code uses MSXML library – so this is just for demonstration. It's preferable to use LoadBatchActions for updates and Forms.AddEx to load forms.

```
Private Sub LoadFromXML(ByVal Filename As String)
        Dim oXMLDoc As MSXML2.DOMDocument
        oXMLDoc = New MSXML2.DOMDocument


        oXMLDoc.load("File Location\" & Filename)
        SBO_Application.LoadBatchActions((oXMLDoc.xml)
End Sub
```

2-4 Use the tools from the B1TE toolset (essentially Form Checker) to check whether you have designed your form(s) according to some important UI guidelines…

Similar solution can be found in the SDK UI samples (in the SDK Folder – see Appendix "SDK Installations" for more information), COM UI/03.SimpleForm, 06.MatrixAndDataSources and 04.WorkingWithXML

Some helpful data for designing Forms

Form	
Height	413px
Width	557px
Controls common	
Distance to left edge	5px
Distance to right edge	5px
Distance to top edge	5px
Distance to bottom edge	5px
Button	
Height	19px
Width	65px
Spacing	5px
Label Field	
Height	14px
Width	Depends on text
Horizontal spacing	>=12px (ungrouped)
Vertical spacing	1px (grouped) >=3px (ungrouped)
Input Field	
Height	14px
Width	Enough to show complete field value
Horizontal Spacing	>=12px (ungrouped)
Vertical spacing	1px (grouped) >=3px (ungrouped)
Field Help	
Vertical distance to input field	1px
Matrix Objects	
Width	Form width minus 2 times 5px to the left and right edge
Number of rows	<=7 rows, add scrollbars if more necessary

Please note that more exhaustive information is available in the “User Interface: Standards and Guidelines” in the “SAP Business One Topic Search” on SAP Servicemarketplace at http://service.sap.com/smb/sbo/resources (October 2007).

Solutions

Unit: User Interface API

Topic: Additional Event Handling

3-1 Catch FormLoad event for Order form

```
If pVal.FormType = "139" And pVal.EventType =
SAPbouiCOM.BoEventTypes.et_FORM_LOAD And pVal.BeforeAction = False Then

End If
```

3-2 Display the message "Caught Order FormLoad Event" when the FormLoad event for the Order form arrives (use the Application.Message method).

```
SBO_Application.MessageBox("Caught Order FormLoad Event")
```

3-3 Catch the click event on the “Rent DVD” button you've added in the previous exercise. Again display a message once the event is caught.

```
If FormUID = "TB1_DVDAvailability" And pVal.ItemUID = "RentDVD" And
pVal.EventType = SAPbouiCOM.BoEventTypes.et_ITEM_PRESSED Then

SBO_Application.MessageBox("Caught click on Rent DVD button")

End If
```

3-4 Create a filter to only receive the events we are interested in.

```
Public oFilters As SAPbouiCOM.EventFilters

Public oFilter As SAPbouiCOM.EventFilter

oFilters = New SAPbouiCOM.EventFilters()



oFilter = oFilters.Add(SAPbouiCOM.BoEventTypes.et_ITEM_PRESSED)
oFilter.AddEx("139") 'Orders Form

oFilter.AddEx(“TB1_DVDAvailability”)

SBO_Application.SetFilter(oFilters)
```

A solution implementing events catching can be found in the SDK UI samples (in the SDK Folder – see Appendix “SDK Installations” for more information), COM UI/ 02.CatchingEvents.

Solutions

Unit: User Interface API

Topic: New Menu entries in SAP Business One window

4-1 Add a menu entry to the "Modules" menu called **DVD Store**. Those are the menu entries which are also displayed in the Main Menu. Add a new sub-menu to that menu entry called **DVD Availability**.

```
Dim oMenus As SAPbouiCOM.Menus
Dim oMenuItem As SAPbouiCOM.MenuItem

oMenus = SBO_Application.Menus

Dim oCreationPackage As SAPbouiCOM.MenuCreationParams
oCreationPackage =
SBO_Application.CreateObject(SAPbouiCOM.BoCreatableObjectType.cot_Me
nuCreationParams)
oMenuItem = SBO_Application.Menus.Item("43520") 'Modules

Dim sPath As String

sPath = Application.StartupPath
sPath = sPath.Remove(sPath.Length - 3, 3)

oCreationPackage.Type = SAPbouiCOM.BoMenuType.mt_POPUP
oCreationPackage.UniqueID = "TB1_DVDStore"
oCreationPackage.String = "DVD Store"
oCreationPackage.Enabled = True
oCreationPackage.Image = sPath & "dvd.bmp"

oMenus = oMenuItem.SubMenus

Try
oMenus.AddEx(oCreationPackage)

oMenuItem = SBO_Application.Menus.Item("TB1_DVDStore")
oMenus = oMenuItem.SubMenus

oCreationPackage.Type = SAPbouiCOM.BoMenuType.mt_STRING
oCreationPackage.UniqueID = "TB1_Avail"
oCreationPackage.String = "DVD Availability"
oMenus.AddEx(oCreationPackage)
Catch er As Exception ' Menu already exists
SBO_Application.MessageBox("Menu Already Exists")
End Try
```

4-2 Handle the menu event: When you choose the menu entry, your form should be displayed.

```
        If pVal.MenuUID = "TB1_Avail" And pVal.BeforeAction = False Then
            LoadFromXML("DVDAvailability.xml")
        End If

Private Sub LoadFromXML(ByVal Filename As String)
        Dim oXMLDoc As MSXML2.DOMDocument
        Try
            oXMLDoc = New MSXML2.DOMDocument

oXMLDoc.load("File location\" & Filename)
            SBO_Application.LoadBatchActions(oXMLDoc.xml)

        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try

    End Sub
```

4-3 Add another menu only visible when your form is open. This menu should appear under the GoTo menu.

Dim oCreationPackage As SAPbouiCOM.MenuCreationParams
oCreationPackage =
SBO_Application.CreateObject(SAPbouiCOM.BoCreatableObjectType.cot_MenuCreationParams)
Dim oMenuForm As SAPbouiCOM.Form

oCreationPackage.Type = SAPbouiCOM.BoMenuType.mt_STRING
oCreationPackage.UniqueID = "TB1_TestMenu"
oCreationPackage.String = "Test Menu"

oMenuForm.Menu.AddEx(oCreationPackage)

A similar solution can be found in the SDK UI samples (in the SDK Folder – see Appendix "SDK Installations" for more information),
COM UI/ 05.AddingMenuItems

Solutions

Unit: User Interface API

Topic: Data Binding

5-1 Declare a DBDataSource and UserDataSource object. Link it to the form you created in the Creating Forms exercise.

oForm.DataSources.DBDataSources.Add("@TB1_DVD")

oForm.DataSources.UserDataSources.Add("DVDName", SAPbouiCOM.BoDataType.dt_SHORT_TEXT)

Note the TB1_DVD table should be defined as Master Data UDO table and registered as a UDO

5-2 Bind the form's items with the corresponding data from the User Defined table created in the DI exercises (*TB1_DVD)*.

Firstly for each Edit Text field bind the field to it's corresponding column in the User defined table e.g.

oEditText.DataBind.SetBound(True, "@TB1_DVDAVAIL", "U_AISLE")

5-3 Get the data from the data sources and display them in the corresponding fields on your form. Note you will first need to read the value selected by the user from the Choose from List (Item Event) and then fill all other fields accordingly.

```
If pVal.EventType = SAPbouiCOM.BoEventTypes.et_CHOOSE_FROM_LIST Then

    Dim oCFLEvent As SAPbouiCOM.ChooseFromListEvent

    Dim oCFL As SAPbouiCOM.ChooseFromList

    Dim CFLID As String

    Dim oForm As SAPbouiCOM.Form


    oCFLEvent = pVal

    CFLID = oCFLEvent.ChooseFromListUID

    oForm = SBO_Application.Forms.Item(FormUID)

    oCFL = oForm.ChooseFromLists.Item(CFLID)

    If oCFLEvent.BeforeAction = False Then


        Dim oDataTable As SAPbouiCOM.DataTable

        oDataTable = oCFLEvent.SelectedObjects

        Dim val As String

        Try

            val = oDataTable.GetValue(1, 0)

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try


        If pVal.ItemUID = "DVDNameT" Then

            oForm.DataSources.UserDataSources.Item("DVDName").ValueEx = val

          oDBDataSource = oForm.DataSources.DBDataSources.Item("@TB1_DVD")

            oDBDataSource.Query()

        End If

    End If

End If
```

5-4 Test your application.

A similar solution can be found in the SDK UI samples (in the SDK Folder – see Appendix “SDK Installations” for more information),
COM UI/03.SimpleForm and, 06.MatrixAndDataSources