'Raising Custom Errors :-

```
Option Explicit
Sub RaisingCustomErrors()
  Worksheets("sheet12").select // Run time error '9' subscript out of range.
End sub
```

```
sub RaisingCustomErrors()

On Error GoTo customError
Worksheets("sheet12").select
  Exit sub
CustomError:
  Err.Raise vbObjectError + 1, "RaisingCustomErrors",
      "Worksheet12 doesnt exist"
End sub.
```

=> Generates unique error number which doesn't exists.

Run-time error: 2147... -
Worksheet12 doesn't exists.

Event procedures :-

How to write code which runs itself Automatically in response to certain events in the life of your workBook.

Creating Event procedure :-

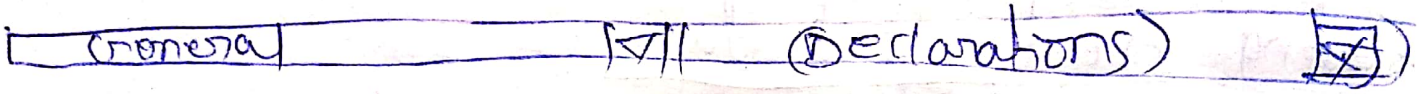
- Accessing the events of an Object
- Adding code to an Event
- Triggering an Event procedure
- A Note on Security
- Cancelling Events
- Disabling events.

Examples of Events :-

- Opening and closing workbooks
- Printing and Saving files
- Inserting worksheets
- Selecting and changing cells

'Accessing the Events of a workbook :-

Here we don't create module, but we will go ThisWorkbook (IV) -> it opens the Workbook object



use the dropdown list in the event window, to add the event what ever you want -

General => Workbook

Declarations => open

private sub workbook_open()

MsgBox "Hello " & Environ("Username"), , Date

End sub.

'Triggering an Event procedure :-

=> we need to trigger the event, instead of running (F5).

=> Just close & open Workbook.

Events and Macro Security :-

Ex :- Enable macro

Developer -> Macro Security ->

⊙ Enable all macros (not recommended; potentially dangerous code can run).

① Disable all macros with notification (recommended).

'Workbook Events - close :-

General -> workbook

Declarations -> Before close

```
option explicit
private sub workbook_BeforeClose (Cancel As Boolean)
msgbox "Goodbye"
end sub.
```

'cancelling events :-

```
private sub workbook_Beforeclose(cancel As Boolean)
MsgBox "You're not leaving"
 cancel = True
End sub.
```

```
private sub workbook_Beforeclose(cancel As Boolean)
 If Hour(Now) < 17 Then
  msgBox "You're not leaving"
  cancel = True
 End If
End sub.
```

'Other Cancellable Events :-

```
 General = workbook
Declaration = Beforeprint
private sub workbook_Beforeprint(Cancel As Boolean)
msgBox "You can't print this file"
 cancel = True
End sub.
```

// we can also test based on User, using Environ("Username").

General = workbook
Declaration = BeforeSave

private sub workbook_BeforeSave(ByVal SaveAsUI
as Boolean, Cancel As Boolean)

msgBox "You can't save this"

cancel = True

end sub.

'Disabling Events :-

General = workbook
Declaration = Newsheet

=> This also runs/triggers during charts sheets also so if condition

This stores a reference to the object which is been created.

private sub workbook_Newsheet (ByVal sh As object)

Dim Howmany As Integer
If Typeof sh Is worksheet Then
Howmany = InputBox ("How many sheets would you like?")

!worksheets.Add Count:= Howmany - 1

// problem is when the new sheets are added, this event is triggered every time.

```
Application.EnableEvents = False
worksheets.Add Count:=Howmany-1
Application.EnableEvents = True
End Sub.
End If
End Sub.
```

'Worksheet Events – selection change :-

=> To create worksheet specific events, simply double click on the sheet.

=> OR in Excel, on Tab -> Right click -> View code.

```
General: worksheet
Declaration: Selection change -> Default event.
private sub worksheet_selectionchange(Byval Target
                                      As Range)
  Target.Interior.color = vbYellow
End sub.
```

Restricting the Target Range :-

```
private sub worksheet_selectionchange(Byval Target
                                      As Range)

If target.Row <=10 And Target.column <=5 Then
Target.Interior.Color = vbyellow
End If                  // If we select, inside the range &
End Sub.                // extends outside, all the cell will
                        // change. so loop over & individually evaluate.
```

Looping over the cells in the Target :-

```
private sub worksheet_selectionchange(Byval Target
                                      As Range)

Dim singlecell As Range
For Each singlecell In Target
  If singlecell.Row <=10 And singlecell.column<=5
                                              Then
    singlecell.Interior.Color = vbyellow
  End If                // The bigger the range
Next singlecell         // will select, the longer the
End Sub.                // time to evaluate. so
                        // only perform if it is
                        // selected to singlecell.
```

'Counting the cells in the Target :-

```
option Explicit
private sub worksheet_selectionchange(Byval Target
                                      As Range)
//count returns long, if we select all cells, runtime error
  so use  CountLarge.                          overflow.
If Target.cells.CountLarge = 1 Then
  If Target.Rows<=10 And Target.column <=5 Then
    Target.Interior.color = vbYellow
End If                  //only single selection will work, but
End If                    Not Range of selection.
End sub.
```

```
'worksheet Events - change.
General = worksheet
Declaration = change
private sub worksheet_change(Byval Target As Range)
'Target.AddComment Now & "_" & Target.value &
                  "_" & environ("UserName")
```

End Sub

// If the cell has already comment, then run-time error '1004'.

If Target.comment is Nothing Then

Target.Addcomment Now & "-" & Target.value & "-" & Environ("UserName")

Else

Target.comment.Text Now & "-" & Target.value & "-" & Environ("UserName")

End If

End Sub

// To add the new comment continue to the old comment

vbNewLine &

Target.Comment.Text, Now & "-" & Target.value & "-" & Environ("UserName"),

Len(Target.comment.Text) + 1,

False

Override parameter.

// if i tried to select multiple cells and try to delete it, run-time error '13' type mismatch.

```
option Explicit
private sub worksheet_change(ByVal Target As Range)
  Dim singlecell AS Range
If Target.cells.countlarge>1000 Then
  For Each singlecell In Target
```

(Target.countlarge=1 => to write specific to cell only.)

(ctrl+h = To replace)

```
Exit sub
If singlecell.comment IS Nothing Then
  singlecell.AddComment Now & "_" & singlecell.value & "_" & Environ("UserName")
Else
  singlecell.comment.Text vbNewline & Now & "_" & singlecell.value & "_" & Environ("UserName"),
  Len(singlecell.comment.Text)+1, False
End If
```

// ctrl + L> To add multiple cells.

~~End sub~~

```
singlecell.Comment.Shape.TextFrame.Autosize=True
Next singlecell
End Sub
```

Events of Embedded objects :-

Developer → Insert → ActiveX controls → commandButton1

General : commandButton1
↓
In properties change
btnClearComments.

Description: click
option Explicit
private sub btnclearComments_click()

cells.ClearComments

End sub.

using Design mode :-
Developer → Design mode →
we can move/~~Resize~~ Relocate button.
If we want to do any more changes, go to ~~Code~~ Design mode → VBA.

properties → caption: Clear Comments.

Application events :-

Application-Level Events

- Recap of workbook and worksheet Events
- Accessing Application-Level events
- The NewWorkbook Event
- Enabling Application-level events
- Using class modules

A Recap of workbook events :-

Thisworkbook :-

```
General = Workbook
Declarations = Newsheet
option explicit
private sub workbook_Newsheet(Byval sh As
                                    object)
Range("A1").Value = "Created on " & Date
Cells.Interior.color = rgbPink.

End sub
```

'A Recap of worksheet Events :-

General = worksheet

Declaration = selectionchange

option explicit

private sub worksheet_selectionchange(ByVal Target As Range)

Target.Interior.color = rgbcoral

End sub.

'Application-level Events :-

View -> Object Browser -> classes (Application) -> members of 'Application' -> => Events symbols like after calculate. New workbook.

'The withEvents keyword :-

choose Thisworkbook.

General = —

(Declaration) = —

option Explicit

private xlApp As Application

private withEvents xlApp As Application

General : xlApp

Declarations: NewWorkbook

private sub xlApp_NewWorkbook(ByVal wb As Workbook)

'The NewWorkbook event

wb.worksheets("sheet1").Range("A1").value
="created on" & Date

wb.worksheets("sheet1").Range("A2").value
="created by" & Environ("UserName")

End sub.

'Enabling Application events

General : workbook

Declarations: Open

```
private sub workbook_open()
  set xlApp = Application
end sub.
```

Using class modules :-

way to seperate Application events from the workbook events is by class modules.

Creating a class module :-

Right click somewhere inside the project → insert → class module.

↓

properties

(Name) : EventApp.

```
option Explicit
private withevents xlApp As Application
```

General: class

Declarations: Initialize.

```
private sub class_initialize()
  set xlApp = Application
End sub
```

General : xlApp

Declarations : Newworkbook

```
private sub xlApp_Newworkbook(ByVal wb As
                                workbook)
wb.worksheets("sheet1").Range("A1").value =
                    "created on" & Date
wb.worksheets("sheet1").Range("A2").value =
          "created by" & Environ("UserName")
End sub
```

// when ever, we open up the workbook, we create a new instance of our Event App class.

General: workbook

Declarations: open

```
option Explicit
```

Just for consistency not any else

```
private xlApp As EventApp
private sub workbook_open()
  set xlApp = New EventApp
end sub
```

---

(24)

User Forms :-

Designing Forms

- Adding a Form to a project
- Adding controls to a Form
- Drawing Tools
- Working with properties
- Adding code to Form Events
- Validating Forms.

---

1 What are User Forms?

=> Designing UserForm in VBA provide a really convinient way to used to interact with the workbook.

'creating a UserForm :-

=> In the project Explorer, Right click some where in the project and choose Insert -> UserForm.

=> you will get Toolbox too.

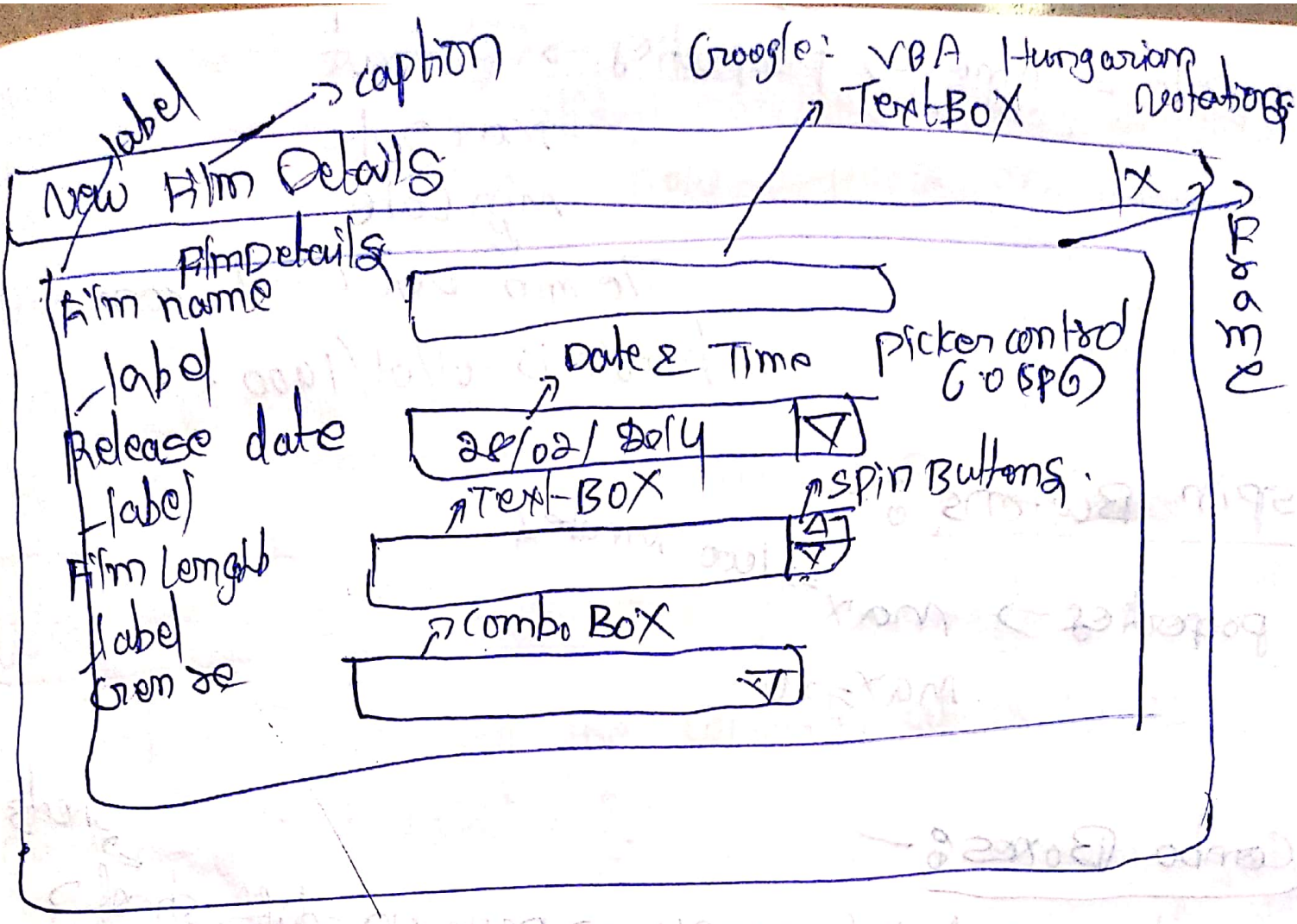
=> Rename
properties -> Form New Film.

'modifying properties of a Form :-

BackColor
caption : New Film Details.

'Adding controls to a Form :-

click on the Background of form, we will get Toolbox.



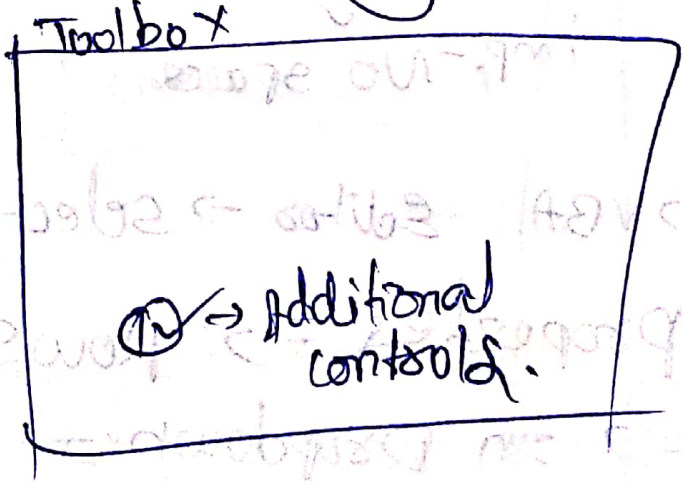
label → properties → (Name) = lblFilmName
caption = Film Name

TextBox → properties → (Name) = txtFilmName
No caption property

F5 → To Run.

Adding Additional controls to Tool box ⇒

☑ Microsoft Date Time Picker control 6.0 (SP6)



Date & Time → properties → Format
Max Date
Min Date
The min Date Excel can handle is 01/01/1900.

Spin Buttons :-
properties → max => 1000 minutes
max => 0

Combo Boxes :-
Excel → select list → copy → paste in diff sheets.
paste → select → Data → Remove Duplicates →
Sort in Ascending order as well. →
Give this list a name in Name Manager
→ List of Genres → TL
imp:- No spaces

=> VBA Editor → select the Combo Box →
properties → RowSource => : ListofGenres
→ in Dropdown it Appears.

Aligning and Distributing Controls :-

=> Drag a box that atleast partially encloses all labels (or)
=> select first label -> ctrl hold -> press on all labels

=> Format -> Align -> Lefts

Format -> Vertical Spacing -> Make Equal

Using Frames to Group Controls :-

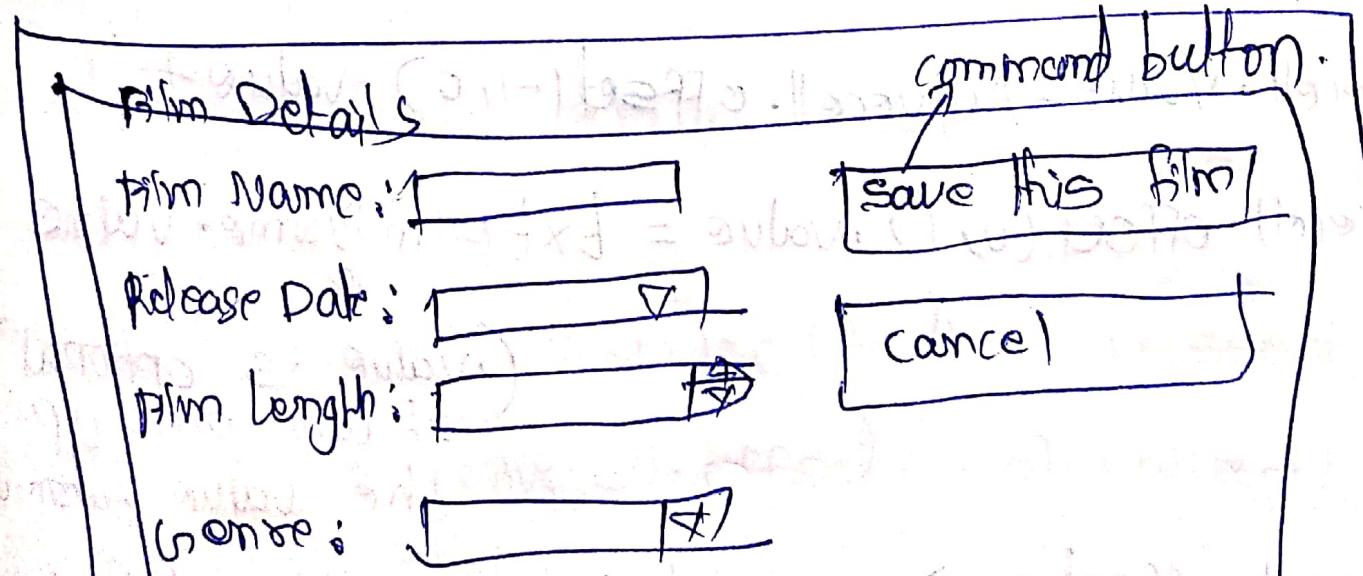
=> Easy to move all the controls at a time.

(Name) : fraFilmDetails

Caption : FilmDetails

Adding code to a Form event :-

=> open form in VBA
click on Background
click F5
to Run form.



save this film (command button) :-

Right click → view code
(or)
Double click

---

option Explicit
General : cmdSaveFilm
~~Decoi~~
Declarations: click

private sub cmdSaveFilm_click()
sheet1.Activate // ctrl+spacebar => Intelligence list.
Range ("A1").End (xlDown).Offset(1,0).Select

Activecell.value = Activecell.offset(-1,0).value + 1
Activecell.offset(0,1).value = txtFilmName.value
(.value is optional)
it will pick up the value automatically
Activecell.offset(0,2).value = ~~txt~~ dtpReleaseDate.value

```
Activecell.Offset(0,3).Value = txtFilmlength.Value
Activecell.Offset(0,4).Value = cboGenre.Value
End Sub
```

```
'changing the Tab Order of Controls.
```

View -> Tab order.

click on Frame -> View -> Tab order -> ok.

Tab -> correct order

shift + Tab -> Reverse order.

Unloading a Form :-

```
Option Explicit
Private Sub cmdSaveFilm_Click()
Sheet1.Activate
Range("A1").End(xlDown).Offset(1,0).Select
Activecell.Value = Activecell.Offset(-1,0).Value + 1
Activecell.Font.Italic = True
```

```
Activecell.offset(0,1).value = txtFilmName.value
Activecell.offset(0,2).value = dtpReleaseDate.value
Activecell.offset(0,3).value = txtFilmlength.value
Activecell.offset(0,4).value = cboGenre.value

'Unload fom.NewFilm
Unload me //we care writing inside fom NewFilm.
End sub.
```

```
option Explicit
General : cmdcancel
Declarations: click
private sub cmdcancel_click()
  Unload me
End sub.
```

```
' Loading and Showing a Form :-
```

⇒ Go To Excel, Developer → Insert →
ActiveX control → choose Command Button.

⇒ Developer → properties → (Name): cmdShowForm
caption: Add New Film.

⇒ Right-click on Button → view code
(or)
Developer → view code.

option Explicit
General : cmdshowForm
Declarations : click

private sub cmdshowForm_click()
⇒ It loads into memory but not on screen.
Load frmNewFilm
we can't use me, because we are writing code in the sheet1 object but not in frmNewFilm object.
frmNewFilm.show ⇒ It loads in memory & display on screen
so, no need load method.
End sub

```
Private sub cmdshowform_click()
Load frmNewFilm                    => spin Button, can't control this value.
frmNewFilm.txtFilmlength.value=100
'
This not shows form in screen, but
will load values to the form before it loads.

frmNewFilm.show

End sub.
```

Developer → Design mode exit → click the Add new film.

'Linking a Text Box to a Spin Button :-

```
General : spnFilmlength
Declarations: change
option Explicit
private sub spnFilmlength_change()
  txtFilmlength.value = spnFilmlength.value
End sub.
```

VBA Editor → select spinButton → properties →
value : 100

Double click on the background of the form.

General : Userform

Declarations : Initialize ⇒ To set value default to Text box, Initialize

```
option Explicit                    option.
private sub UserForm_Initialize()
  txtFilmlength.value = spnFilmlength.value
End sub.
```

Locking and Disabling controls :-

If we enter value in Filmlength & Then use spin Button, it behaves strange. ie., 200 changes to 103.

⇒ we could either prevent the user to Enter Manually (or) of spin to update

→ we could control sub, when ever user changes the value.

VBA editor → select FilmLength TextBox → properties →

Locked : True / False . // we can click inside box

(or)

Enabled : True / False . // we can't even click inside box

' Linking a spin Button to a Text Box :-

Double click on txtFilmLength (TextBox)

~~opt~~ General : txtFilmLength

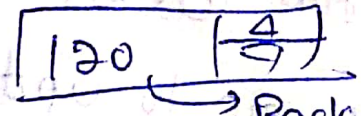
Declarations : Change

private sub txtFilmLength_change ()

spnFilmLength . value = txtFilmLength . value .

End sub .

' Validating a Single Text Box :-

FilmLength : 120 

Backspace 1

The value in the TextBox will become an empty string . VBA can't store

an empty string in an integer/long integer datatype - mismatch error :

Runtime error : 13 : Type mismatch.

Double click on Filmlength Text Box, and

General: txtFilmlength

Declarations: change

option explicit txt

private sub ~~sp~~FilmLength_change()

If IsNumeric (txtFilmlength.value) Then

spinFilmlength.value = txtFilmlength.value

~~End If~~ txtFilmlength.BackColor = vbwhite

lblFilmLength.ForeColor = vbBlack

~~End sub~~.

Else

txtFilmlength.BackColor = vbRed

lblFilmlength.ForeColor = vbRed

End If

End sub.

Checking the Range of a Spin Button :-

General : txtFilmLength

Declarations : change

```
private sub txtFilmLength_change()
If IsNumeric(txtFilmLength.value) And
   txtFilmLength.value >= spnFilmLength.min And
txtFilmLength.value <= spnFilmLength.max Then
  spnFilmLength.value = txtFilmlength.value
txtFilmlength.Backcolor = vbwhite
lblFilmlength.ForeColor = vbBlack
Else
   txtFilmlength.BackColor = vbRed
   lblFilmlength.ForeColor = vbRed
End If
End sub
```

Ex :- 1234 → Run time error 380 → Invalid property value / could not set value Property

Validating multiple Controls :-

Double click on Save This Film Button.

General : cmdSaveFilm

Declarations: click

```
Option Explicit

Private Sub cmdSaveFilm_Click()

If txtFilmName.Value = "" Then
   txtFilmName.BackColor = vbRed
   lblFilmName.ForeColor = vbRed
   txtFilmName.SetFocus
   Exit Sub
End If
Sheet1.Activate
Range("A1").End(xlDown).Offset(1,0).Select
Activecell.Value = Activecell.Offset(-1,0).Value+1
Activecell.Font.Italic = True
```

```
Activecell.offset(0,1).value = txtFilmName.value
Activecell.offset(0,2).value = dtpReleaseDate.value
Activecell.offset(0,3).value = txtFilmLength.value
Activecell.offset(0,4).value = cboGenre.value

Unload me
End sub
```

⇒ Double click on FilmName Text Box

```
General : txtFilmName
Declarations: change
option Explicit
private sub txtFilmName_change()
    txtFilmName.Backcolor = vbwhite
    lblFilmName.Forecolor = vbBlack
End sub.
```

Validating a combo Box :-

Double click on Save this Film Button.

```
General : cmdSaveFilm

Declarations : Click

private sub cmdSave_click()

if ---
  ----          //Refer validating multiple
  ----                controls.

end if

if cboGenre.value = "" Then
   cboGenre.BackColor = vbRed
   lblFilmGenre.ForeColor = vbRed
   cboGenre.setFocus
end sub
end if
Sheet1.Activate.
  --          //Refer validating multiple
  --                controls.
  --
end sub.
```

General : cboGenre

Declarations : change

option Explicit

private sub cboGenre_change()

cboGenre.Backcolor = vbwhite

lblFilmGenre.Forecolor = VbBlack

End sub.

' Validating a Date and Time picker &-

Double click on save this film Button.

General : cmdsaveFilm

Declarations : click

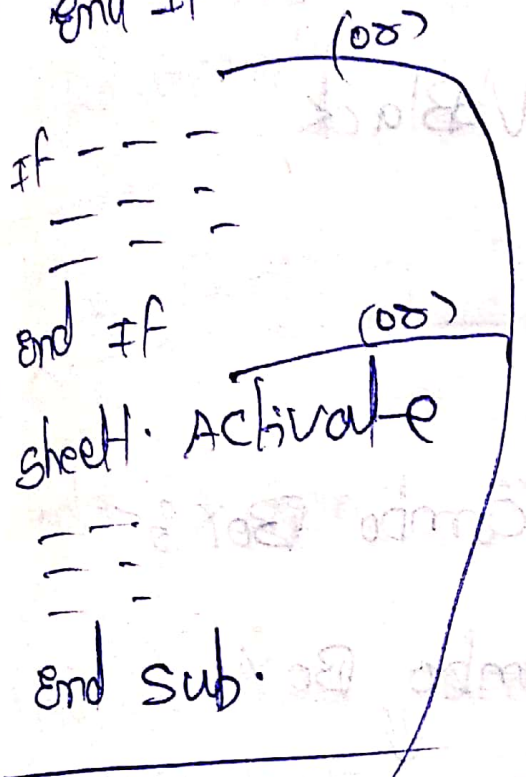
private sub cmdsaveFilm_click()

If ---

End If

(08)

If txtFilmlength.Backcolor = vbRed Then
exit sub
end If



If dtpReleaseDate.value > Date Then
dtpReleaseDate.SetFocus
lblReleaseDate.ForeColor = vbRed.
Exit sub

General : dtpReleaseDate
Declarations : change

```
option Explicit
private sub dtpReleaseDate_change()
 If dtpReleaseDate.value <= Date Then
    lblReleaseDate.ForeColor = vbBlack
 End If
End sub.
```

'Restricting the values in a combo Box%

=> we can directly type in combo Box

⇓

Create ~~Text Box~~ combo Box.

=> select combo box (Create combo Box) → properties → MatchRequired : True.

Files and Folders (Filesystemobject) 25

Working with the File system

- The scripting Runtime library
- Creating a Filesystemobject

- creating Folders
- copying files
- looping over Files in a Folder
- Recursively looping over Folders.

The Scripting RunTime Library :-

Tools -> References -> ☑ Microsoft scripting Runtime
-> OK.

Now we have access to the whole world of objects, collections, methods & properties devoted to work with Files & Folders.

Using the Object Browser :-

View -> Object Browser (F2) ->
scripting (Filesystemobject)

Modules -> FilesAndFolders

Creating a New File System Object :-

General :

Declarations : Using The Scripting Run Time library

option Explicit

sub usingTheScriptingRunTimeLibrary()

Dim fso As scripting.FileSystemObject ⇒ object

set fso = new scripting.FileSystemObject

fso. — BuildPath
— CopyFile
⋮ } we have access to all the filesystem objects.

end sub

Best

Using Auto-Instancing Variables :-

⇒ Other couple of techniques, which are used to create New FileSystem objects.

sub AutoInstancingVariable()

'Dim fso As New Scripting.FileSystemObject

fso.

=> It combines the 2-lines which we wrote previously.

=> This line however doesn't actually creates New instance object, but when we use fso variable name, The VBA Editor checks if fso has been set to a new instance of filesystem object, if it hasn't, it will Automatically creates a new instance.

Pownsides:-

Every single time, we use fso, the VBA Editor checks for New instance.

~~If fso Is Nothing Then~~ => checks & creates.

=> It is always false.

End sub.

'Using the CreateObject Function :-

Tools → References → uncheck → Microsoft Scripting Runtime.

```
sub UsingCreateObject()
    Dim fso As Object
    set fso = createobject("Scripting.FileSystemObject")
fso.
End sub
```

⇒ Drawback.
⇒ No Intelligence.
↓
so we need to know every single method.

Because this is string of text, the VBA editor doesn't need to interpit when we compile that project.

'Creating a Folder :-

```
option Explicit
sub UsingTheScriptingRuntimeLibrary()
```

```
Dim fso As scripting.FileSystemObject
set fso = New scripting.FileSystemObject
fso.CreateFolder "c:\Users\Andrew.Gould\Desktop\wise owl"
```

=>Good practice.

```
set fso = Nothing
```

=>optional in VBA Because end sub will do it.

opposite of what we do while creating new instance.

This releases, the object it destroys/terminates. And releases free space.

```
End sub.
```

'Testing if a Folder Exists :-

If we try to run above code again,

Run-time error '58': File already Exists.

```
If not fso.FolderExists("c:\Users\Andrew.Gould\Desktop\wise owl") Then
```

```
fso.CreateFolder "C:\Users\Andrew.Gould\
                  Desktop\wise owl"
End If.
```

'Using Variables to store paths -

```
option Explicit
sub UsingTheScriptingRunTimeLibrary()
  Dim fso As scripting.FilesystemObject
  Dim NewFolderpath As string
NewFolderpath="C:\Users\Andrew.Gould\Desktop\
                                wise owl"
  set fso = New scripting.FilesystemObject
If Not fso.FolderExists(NewFolderPath) Then
  fso.CreateFolder NewFolderpath
End If
set fso = Nothing
End sub
```

'Detecting the User profile :-

view → Immidiate window.

? Environ("UserProfile")

C:\Users\Andrew.Gould

NewFolderPath = Environ("UserProfile") &
"\Desktop\Wise owl"

'copying Files :-

End If

fso.CopyFile Source:="C:\Users\Andrew.Gould\
Desktop\VBA Files\Files for course
characters.xlsm", Destination:=
~~NewFolderPath~~.
NewFolderPath & "\characters ~~copy~~.xlsm"

//

If already exits, overrides
Default True (parameter).

set fso = Nothing

End Sub

'Testing if a File Exists

```
Dim - - -
Dim oldFolderpath As String

New - - -
oldFolderpath = Environ("Userprofile") & "
    Desktop\VBA Files\Files for course"
set - - -
If - - -
End If
If fso.FileExists(oldFolderpath & "\characters.xlsm")
                                        Then
  fso.CopyFile_
    Source:=oldFolderpath & "\characters.xlsm"_
, Destination:=NewFolderpath & "\characters.xlsm"
End If
set fso= Nothing
End Sub.
```

'Using File Object Variables :-

```
option Explicit
sub UsingTheScriptingRunTimeLibrary()
Dim fso As scripting.FilesystemObject
Dim fil As scripting.File
Dim NewFolderpath As string
Dim oldFolderpath As string
NewFolderpath = Environ("Userprofile") & "\Desktop\
                                    wise owl"
oldpath
oldFolderpath = Environ("Userprofile") & "\Desktop\
                    VBA Files\Files for course"
set fso = New scripting.Filesystemobject
If Not fso.FolderExists(NewFolderpath) then
  fso.CreateFolder Newfolderpath
End If
```

If fso.FileExists(oldFolderpath & "
\characters.xlsm") Then

set fil = fso.GetFile(oldFolderpath &
"\characters.xlsm")

Examing file properties.

~~fil debug.print~~

'debug.print fil.Name, fil.Path, fil.DateCreated,
fil.DateLastModified, ~~fil~~ fil.Type,
fil.Size.

'View → Immediate window.

End If
set fso = Nothing // Example.
End sub.

```
If fil.size > 20000 Then
'fil.copy NewFolderpath & "\characters.xlsm"
                          ||
                    same Name override
                    becoz True.

fil.copy NewFolderpath & "\" & fil.Name

end if
end if
set fso=nothing
End sub
```

'Looping over the files in a Folder :-

```
option Explicit
sub UsingTheScriptingRunTimelibrary()

Dim fso AS scripting.Filesystemobject
Dim fil AS scripting.File
Dim oldFolder AS scripting.Folder
Dim NewFolderpath AS String
Dim oldFolderpath AS string
```

```
NewFolderpath = Environ("Userprofile") & "\Desktop\
                                          wise owl"
OldFolderpath = Environ("Userprofile") & "\Desktop\
                    VBA Files\Files for course"

set fso = new scripting.FilesystemObject

If fso.FolderExists(oldFolderpath) Then
  set oldFolder = fso.GetFolder(oldFolderpath)
If Not fso.FolderExists(NewFolderpath) Then
   fso.CreateFolder NewFolderpath
End If
For Each fil in oldfolder.Files
   'debug.print fil.Name
'If fso.GetExtensionName(fil.path) = "xls" Then
   'debug.print fil.Name
'End If
```

```
If Left(fso.GetExtensionName(fil.path),2) =
                                    "xl" Then
'Debug.Print fil.Name
fil.Copy NewFolderPath & "\" & fil.Name

End If
Next fil
End If
Set fso = Nothing
End Sub
```

'Recursively Looping over Folders :-

```
Option Explicit
Dim fso As Scripting.FileSystemObject
Dim NewFolderPath As String
                    or we can pass as parameter.

Sub UsingTheScriptingRuntimeLibrary()
```

```
Dim oldFolderpath As string

NewFolderpath = Environ("Userprofile") & "
                \Desktop\wiseowl"

oldFolderpath = environ ("Userprofile") & "
                "Desktop\VBA Files\Files for course"


set fso = New scripting.FileSystemobject

If fso.FolderExists(OldFolderpath) Then

If Not fso.FolderExists(NewFolderpath) Then
  fso.CreateFolder NewFolderpath

End If
  'copyExcelFiles oldFolderpath

  Call copyExcelFiles (OldFolderpath)

End If
set fso = Nothing
End Sub
```

```
sub copyExcelFiles(startFolderpath$ As string)
  Dim fil As scripting.File
  Dim oldFolder As scripting.Folder
set oldFolder = fso.GetFolder(startFolderpath)
for Each fil In oldFolder.Files
 If left(fso.GetExtensionName(fil.path),2)
                              = "xl" Then
  fil.copy NewFolderpath & "\" & fil.Name
 End If
 Next fil
end sub.
```

1

Making a subroutine call #self :-

```
sub copyExcelFiles(startFolderpath As string)
Dim fil As scripting.File
Dim subfol As Scripting.Folder
```

```
Dim oldFolder As scripting.Folder
set oldFolder = fso.GetFolder(startFolderpath)
For Each fil In oldFolder.Files
If Left(fso.GetExtensionName(fil.path),2)
                                    ="x!" Then
  fil.copy NewFolderpath & "\" & fil.Name
End If
Next fil
For Each subfol In oldFolder.subFolders
  call copy Excel Files (subfol.path)
            ⇓
      Recursive programming
      (calling itself)
next subfol
End sub.
```

The locals window and Call stacks -

view → locals window

view → call stack (disactivated)

↓

As we step in it will activate. (F8)

view → call stack.

↓

shows all the open & running subroutines.

Text files :-

(86)

Creating, writing and Reading Text files :-

- The scripting Runtime library and FileSystem Object.
- Creating and writing to a Text File
- Opening an Existing Text File
- Appending to a Text File
- Reading from a Text File
- Using Text Files with events.

The Scripting Runtime Library :-

Modules
→ WorkingwithTextFiles.

General :-

Declarations: Creating A NewTextFile

option Explicit

sub CreatingANewTextFile()

'Tool → References → ☑ Microsoft Scripting Runtime (Library)

~~option Explicit~~

~~sub creating~~

Dim fso AS scripting.filesystemobject
Dim ts AS scripting.textstream
set fso = new scripting.filesystemobject

'creating a text file :-

set ts = fso.createTextFile ("C:\users\Andrew.Gould\Desktop\wise owl\Test.txt",2)
//2nd parameter overside = true
//3rd Unicode = False = softer languages/...

End sub

Writing to a Text File :-

```
option Explicit
sub creatingANewTextFile()
Dim fso AS scripting.FileSystemObject
Dim ts AS scripting.TextStream

set fso = New scripting.FileSystemObject
set ts = fso.CreateTextFile("C:\Users\Andrew.Gould\Desktop\wise owl\Text.txt")

'There are 3 different methods for writing to a text file.

ts.write "created on" & Now & vbNewLine

'The writeLine method
ts.writeLine "created by" & Environ("UserName")
                    ⇓
                Here no need vbNewline.
'writing Blank lines
```

ts.writeBlankLines 2
ts.writeLine "Data starts here"
ts.close
set fso = Nothing
End sub.

Dim i As Integer.

'Opening a File for Appending :-

option Explicit
sub AddDataToTextFile()
Dim fso As scripting.FileSystemObject
Dim ts As scripting.TextStream
Dim r As Range)Dim colcount As Integer
set fso = New scripting.FileSystemObject

set ts = fso.OpenTextFile (_ -> continuation character.
Environ("UserProfile") & "\Desktop\wise owl\Text.txt" , _
For Appending 8, 3rd parameter
create Boolean = False
if creates if file not their

Writing Tab-Delimited values :-

```
sheet1.Activate
colcount = Range("A8", Range("A2").End(xlToRight)).
                                                Cells.Count
for each r in Range("A2", Range("A1").
                                    End(xlDown))

  For i=1 To colcount
  'ts.write r.Offset(0, i-1).Value & vbTab
If i < colcount Then ts.write vbTab
Next i
ts.writeline
Next r
ts.close
set fso= Nothing
End Sub.
```

Creating comma-separated values :-

```
Sub AddDataToCsvFile()

 Dim fso As scripting.FileSystemObject
 Dim ts AS scripting.TextStream
 Dim r AS Range
 Dim colcount AS integer
 Dim i AS integer

set fso = New scripting.FileSystemObject
set ts= fso.openTextFile (Environ ("Userprofile") &
                "\Desktop\wise owl\Test.csv",
                ForAppending, True)
 sheet1.Activate

colcount = Range("A2", Range("A2").end(xlToRight)
                                .cells.count

For each r In Range("A2", Range("A1").End(xlDown

For i=1 To colcount
```