Faculdade de Tecnologia Arthur de Azevedo

Mogi Mirim, 30 de Novembro de 2022

Análise e Desenvolvimento de Sistemas

**Trabalho de Programação em Microinformática**

**Prof. Me. Marcos Roberto de Moraes**

Amanda Luisa Pereira RAMOS

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

amanda.ramos6@fatec.sp.gov.br

Andriele Inácio da SILVA

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

andriele.silva@fatec.sp.gov.br

Cecília Martins CASTRO

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

castrocecilia768@gmail.com

Kleber Franzotti das Neves

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

kleber.neves@fatec.sp.gov.br

Laysa Vieira Cavalcante BONATTI

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

laysa.bonatti@fatec.sp.gov.br

Natanael Lucas Pio SIMÕES

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

natanael.simoes@fatec.sp.gov.br

Vinicius Modena NUNES

Fatec Arthur de Azevedo - Mogi Mirim - CEETEPS

vinicius.nunes10@fatec.sp.gov.br

RESUMO

O trabalho tem como objetivo apresentar de forma clara os usos variados do Visual Basic Application (VBA), utilizando pesquisas sobre o assunto em sites confiáveis e no material de aula disponíveis no SIGA. Nosso trabalho tem como objetivo final por nossos conhecimentos a prova em relação ao entendimento geral do que nos foi passado.

**Palavras-chave:** Visual Basic Application, Objeto, Excel

INTRODUÇÃO

Iremos abordar Objetos em Visual Basic Application. Para o Visual Basic, Objeto é qualquer coisa que possa ser manipulada ou visualizada de alguma forma. O Excel possui mais de 120 tipos diferentes de objetos. Para nosso estudo iremos abordar quatro objetos: Objeto Application, Objeto Workbook, Objeto Worksheet e Objeto Range. Com estes vamos explorar algumas de suas funcionalidades e como um objeto pode ser alterado por meio de suas propriedades e de seus métodos.

**OBJETO APPLICATION**

Representa o próprio Excel e seus objetos derivados. Por meio dele, podemos configurar o Excel em termos de visualização, execuções e outras funcionalidades.

**Principais propriedades**

**DisplayAlerts** – Habilita ou desabilita a exibição de mensagens de aviso como resposta de várias ações no Excel. Exemplo:

*Application.DisplayAlerts = False ( Desabilita mensagem de aviso)*

*Aplication.DisplayAlerts = True (Habilita a mensagem de aviso)*

**DisplayCommentlIndicador –** Controla a exibição de comentários dentro da células de uma planilha: exemplo:

*Application.DisplayCommentIndicador = x1CommentAndIndicador*

*Application.DisplayCommentIndicador = x1CommentIndicadorOnly*

**DisplayFormulaBar** – Exibe ou oculta a barra de fórmulas do Excel. Exemplo:

*Application.DisplayFormulaBar = False*

*Application.DisplayFormulaBar = True*

**DisplayFullScreen** – Exibir o Excel em tela cheia. Exemplo:

*Application.DisplayFullScreen = True*

*Application.DisplayFullScreen = False*

**EnableAnimations** – Ativa ou desativa a animação quando inserimos ou excluímos linhas e colunas em uma planilha. Exemplo:

*Application.EnableAnimations = True*

*Application.EnableAnimations = False*

**EnableAutoComplete** – Habilita ou desabilita o recurso Autocompletar em textos já conhecidos. Exemplo:

*Application.EnableAutoComplete = False*

*Application.EnableAutoComplete = True*

**EnableSound** – Habilita ou Desabilita a emissão de beep dentro do Excel. Exemplo:

*Application.EnableSound = False*

*Application.EnableSound = True*

**Principais métodos**

**Onkey** – Responsável pela atribuição de teclas de atalho para procedimentos específicos. Esse método é sempre acionado para designar teclas de atalho no momento de criação de uma macro.

*Application.OnKley (Tecla, procedimento)*

*Tecla = string que mostra uma tecla ou combinação de teclas.*

*Procedimentos= String que representa o nome do procedimento.*

**Quit** – Método que encerra o Excel e executa os procedimentos comuns antes de finalizar.

*Application.Quit*

# Principais propriedades:

* DisplayAlerts

Habilita ou desabilita a exibição de mensagens de aviso como resposta de várias ações no excel. Exemplo:

*Sub Exemplo1 ( )*

*Application.DisplayAlerts = False ‘desabilita mensagem de aviso*

*Application.Worksheets (“Plan2”).Delete*

*Msgbox “Planilha foi excluída sem msg”*

*Application.DisplayAlerts = True ‘Habilita mensagem de aviso*

*Application.Worksheets (“Plan3”).Delete*

*Msgbox “Planilha excluída, agora com mensagem de aviso”*

*End Sub*

# DisplayCommentIndicator

Controla a exibição de comentários dentro das células de uma planilha. Exemplo:

*Sub Exemplo2 ( )*

*Plan1.Range (“A1”).ClearComments*

*Plan1.Range (“A1).Addcomment “Valor do PI”*

*Plan1.Range(“A1”).Value = 3.14*

*Application.DisplayCommentIndicador = x1commentAndIndicador*

*Msgbox “Exibindo Comentário com indicador”*

*‘mudando o estilo do comentário*

*Plan1.Range (“A1”).ClearComments*

*Plan1.Ranger (“A1”).AddComment “Valor do PI”*

*Plan.Ranger(“A1”).Value = 3.14*

*Application.DisplayCommentIndicador = x1commentIndicadorOnly*

*MsgBox “Somente com o indicador”*

*End Sub*

* DisplayFormulaBar

Exibe ou oculta a barra de fórmulas do Excel.

*Sub exemplo3 ( )*

*Application.DisplayFormulabar = False ‘oculta a barra*

*MsgBox “Barra desabilitada”*

*Application.DisplayFormularBar = True ‘oculta a barra*

*MsgBox “Barra habilitada”*

*End Sub*

* DisplayFullScreen

Exibe o Excel em Tela Cheia.

*Sub exemplo4 ( )*

**Objeto Workbook**

O objeto Workbook representa uma pasta de trabalho do Microsoft Excel. O objeto Workbook é um membro da coleção Workbooks. A coleção Workbooks contém todos os objetos Workbook atualmente abertos no Microsoft Excel. A propriedade ThisWorkbook do objeto Application retorna ao workbook onde o código Visual Basic está sendo executado. Na maioria dos casos, isso é igual à pasta de trabalho ativa. No entanto, se o código do Visual Basic fizer parte de um suplemento, a propriedade ThisWorkbook não retornará a pasta de trabalho ativa. Nesse caso, a pasta de trabalho ativa será a pasta de trabalho que chama o suplemento, enquanto a propriedade ThisWorkbook retorna a pasta de trabalho do suplemento. Se você estiver criando um Visual Basic do seu código de Visual Basic, use a propriedade ThisWorkbook para qualificar qualquer instrução que deve ser executada na lista de trabalho compilada no complemento.

Exemplo 1

*Workbooks(1).Activate*

Use Workbooks (index), onde index é o nome da carteira de trabalho ou número de índice, para retornar um único objeto Workbook. O exemplo acima ativa a pasta de trabalho um. O número do índice denota a ordem na qual as pastas de trabalho foram abertas ou criadas. Workbooks(1) é a primeira pasta de trabalho criada e Workbooks(Workbooks.Count) é a última. A ativação de uma pasta de trabalho não altera seu número do índice. Todas as guias de trabalho estão incluídas na contagem de índices, mesmo que elas sejam ocultas.

Exemplo 2

*Workbooks("Cogs.xls").Worksheets("Sheet1").Activate*

O exemplo acima ativa a Planilha1 na pasta de trabalho chamada Cogs.xls (a pasta de trabalho já deverá estar aberta no Microsoft Excel).

Exemplo 3

*Sub Exemplo\_3()*

*Dim wks As Worksheet*

*Application.ScreenUpdating = False*

*Set wks = ActiveSheet*

*Worksheets(Range("C1").Value).Copy*

*ActiveWorkbook.SendMail wks.Range("A1").Value, wks.Range("B1").Value*

*ActiveWorkbook.Close savechanges:=False*

*Application.ScreenUpdating = True*

*End Sub*

Este exemplo envia e-mail de uma guia de planilha da pasta de trabalho ativa usando um endereço de e-mail e assunto especificados. Para executar este código, a planilha ativa deverá conter o endereço de e-mail na célula A1, o assunto na célula B1 e o nome da planilha a ser enviada na célula C1.

# PRINCIPAIS PROPRIEDADES

# **Full Name**

Retorna o nome do objeto, incluindo seu caminho no disco, na forma de uma cadeia de caracteres. String somente leitura.

Exemplo 1

*MsgBox ActiveWorkbook.FullName*

O exemplo acima exibe o caminho e o nome de arquivo da pasta de trabalho ativa (desde que a pasta de trabalho tenha sido salva).

Exemplo 2

*Public Sub Exemplo\_2()*

*Dim nome As String*

*nome = Workbooks(1).FullName*

*MsgBox nome*

*End Sub*

No exemplo acima retorna o Path e o Name do objeto. Digamos que uma pasta chamada Amanha.xls, salva em C:\trabalho. Retornaria: “C:\trabalho\Amanha.xls”.

* **Path**

Retorna uma Cadeia de Caracteres que representa o caminho completo para a pasta de trabalho/arquivo que esse objeto de pasta de trabalho representa.

Exemplo

*Public Sub exemplo()*

*Dim nome As String*

*nome = Workbooks(1).Path*

*MsgBox nome*

*End Sub*

No exemplo acima retorna o caminho onde está salvo a pasta de trabalho desejada.

* **Saved**

True se nenhuma alteração tiver sido feita no manual de trabalho especificada desde que foi salva pela última vez. Boolean de leitura/gravação. Se uma workbook nunca tiver sido salva, sua propriedade Path retornará uma cadeia de caracteres vazia (""). Você poderá definir essa propriedade como True se desejar fechar uma pasta de trabalho modificada sem salvá-la ou ser avisado para salvá-la.

Exemplo 1

*If Not ActiveWorkbook.Saved Then*

*MsgBox "This workbook contains unsaved changes."*

*End If*

No exemplo acima exibirá uma mensagem se a pasta de trabalho ativa contiver alterações não salvas.

Exemplo 2

*Public Sub exemplo\_2()*

*Dim teste As Boolean*

*Teste = Workbooks(1).Saved*

*Msgbox teste*

*End Sub*

No exemplo acima, indica se a pasta de trabalho já está salva ou não. **True** indica que já foi salva. **False** indica que não foi salvo.

* **HasPassword**

É utilizado para verificar se a pasta de trabalho possui uma senha de acesso. Em caso positivo, seu retorno é **True**. Você pode atribuir uma senha de proteção a uma pasta de trabalho usando o método SaveAs.

Exemplo

*Public Sub Exemplo()*

*Dim teste As Boolean*

*Teste = Workbooks(1).HasPassword*

*MsgBox teste*

*End Sub*

No exemplo acima verificamos se a pasta de trabalho possui uma senha de acesso ou não. Em caso de haver uma senha, o retorno será **True.**

**Principais Métodos**

* **Activate**

Ativa a primeira janela associada à pasta de trabalho. Esse método não executará nenhuma Auto\_Activate ou Auto\_Deactivate que possam ser anexadas à guia de trabalho (use o método RunAutoMacros para executar essas macros). Fechar uma pasta de trabalho do Visual Basic não executa macros Auto\_Close na pasta de trabalho. Use o método RunAutoMacros para executar as Auto\_Close macros.

Exemplo 1

*Workbooks("BOOK4.XLS").Activate*

No exemplo acima, ativa o Book4.xls. Se ela tiver várias janelas, o exemplo ativará a primeira janela, Book4.xls:1.

Exemplo 2

*Public sub exemplo\_2()*

*Workbooks(1).Activate*

*Planilha1.Range(“A1”).Value = “teste”*

*End Sub*

No exemplo acima este método torna uma pasta de trabalho ativa.

* **Close**

Neste método é fechado o objeto ou pasta de trabalho desejada.

Exemplo 1:

*Workbooks("BOOK1.XLS").Close SaveChanges:=False*

No exemplo acima, fecha o arquivo Pasta1.xls e descarta todas as alterações feitas nele, pois a opção SaveChanges está como “FALSE”.

Exemplo 2:

*Public Sub exemplo\_2()*

*MsgBox “Fechando a pasta”*

*Workbooks(1).Close*

*End Sub*

No exemplo acima é mostrada uma mensagem ao usuário, “Fechando a pasta” e então a pasta de trabalho é fechada.

* **PrintOut**

Imprime toda a pasta de trabalho desejada, ou parte dela, caso especificado. Páginas nas descrições de From e To refere-se a páginas impressas, não a páginas gerais na planilha ou pasta de trabalho. A seguir alguns parâmetros importantes utilizados.

· **From -** O número da página pela qual começar a impressão. Se esse argumento for omitido, a impressão começará pelo início.

· **To -** O número da última página a imprimir. Se esse argumento for omitido, a impressão terminará na última página.

· **Copies -** O número de cópias a imprimir. Se esse argumento for omitido, será impressa uma cópia.

· **Preview -** True para que o Microsoft Excel invoque a visualização de impressão antes de imprimir o objeto. False (ou omitido) para imprimir o objeto imediatamente.

· **ActivePrinter -** Define o nome da impressora ativa.

· **PrintToFile -** True para imprimir para um arquivo. Se PrToFileName não for especificado, o Microsoft Excel solicitará ao usuário que digite o nome do arquivo de saída.

· **Collate -** True para agrupar múltiplas cópias.

· **PrToFileName –** Se PrintToFile for definido como True, esse argumento especificará o nome do arquivo para o qual você deseja imprimir

Exemplo 1:

*ActiveSheet.PrintOut*

*No exemplo acima, é impressa a planilha ativa.*

*Exemplo 2:*

*Public Sub teste()*

*Workbooks(1).Ptintout*

*End Sub*

No exemplo acima é o exemplo mais comum de PrintOut. Imprime a pasta de trabalho desejada.

* **Save**

Salva as alterações na pasta de trabalho especificada. Para abrir um arquivo de pasta de trabalho, use o método Open. Para marcar uma workbook como salva sem ser escrita em um disco, de definir sua propriedade Saved como True. Na primeira vez que você salvar uma pasta de trabalho, use o método SaveAs para especificar um nome para o arquivo.

Exemplo 1:

*ActiveWorkbook.Save*

No exemplo acima, é possível salvar a pasta de trabalho ativa.

Exemplo 2:

*For Each w In Application.Workbooks*

*w.Save*

*Next w*

*Application.Quit*

No exemplo acima, todas as pastas de trabalho abertas são salvas, então o Microsoft Excel fecha.

Exemplo 3:

*public Sub Workbook\_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)*

*WorksheetFunction.CountA(Worksheets(“Sheet1”).Range(“D5,D7,D9,D11,D13,D15”)) < 6 Then*

*MsgBox “Workbook will not be saved unless” & vbCrLf & \_*

*“All required fields have been filled in!”*

*Cancel = True*

*End If*

*End Sub*

No exemplo acima é utilizado o evento BeforeSave para verificar se determinadas células contêm dados antes que a workbook possa ser salva. A pasta de trabalho não poderá ser salva até haver dados em cada uma destas células: D5, D7, D9, D11, D13 e D15.

**Eventos**

Evento é a ocorrência de uma ação do usuário ou do sistema sobre um objeto qualquer. Exemplo de eventos: Quando uma pasta é aberta, quando uma pasta é ativada ou quando uma pasta é fechada. Estes são alguns exemplos de eventos de um WorkBook. Abaixo estão alguns exemplos dos principais eventos e quando ocorrem.

· **Activate -** Quando a pasta de trabalho é ativada.

· **BeforeClose -** Antes da pasta de trabalho ser fechada.

· **BeforePrint -** Antes da impressão de uma planilha ou parte.

· **Deactivate -** Oposto do activate – pasta desativada.

· **NewSheet -** Quando uma nova planilha é criada.

· **Open -** Quando a pasta de trabalho é aberta.

· **SheetActivate -** Quando qualquer planilha for ativada.

· **SheetBeforeDoubleClick -** Quando é dado duplo clique numa planilha.

· **SheetBeforeRightClick -** Quando é clicado sobre o botão direito.

· **SheetCalculate -** Após qualquer planilha ser recalculada.

· **SheetSeleciontChange -** Quando muda a seleção de células de qualquer planilha.

**Objeto WorkSheet**

A classe Worksheet representa uma planilha.

As classes Application, Chart, Range, Window e Workbook dão acesso à classe Worksheet

**Propriedades**

**CodeName** – Retorna o nome de código de uma planilha. Substitui a declaração worksheets (Index).

Pode ser usado: Plan1.Cells(1,1).Valeu =10

Ao invés de: WorkSheets(1).Cells(1,1).Value = 10

**Name** – Retorna o nome da planilha. Exemplo de aplicação:

*MsgBos Worksheets(1).Name*

**ScrollArea** – Limita a área de rolagem e atuação da planilha. Exemplo de aplicação:

*ActiveSheet.ScrollArea = “A1: D15”*

**Visible** - controle de visibilidade de uma planilha. Exemplo:

xIScheetHidden = ocuta a planilha.

xISheetVisible = deixa visível novamente.

xISheetVeryHidden = permite deixar a planilha visível apenas via código.

**Métodos**

**Active –** Torna uma planilha da coleção workSheets ativa. Exemplo:

*Worksheets(2).Activate*

**Cooy –** serve para criar uma cópia de uma planilha. Exemplo:

*ActiveSheet.copy after: = workSheets (workSheets.count)*

**Delete –** Exclui uma planilha da coleção. Exemplo

*Worksheets (1). Delete*

**Move –** Move uma planilha de posição.

**Select –** Seleciona uma planilha. Diferente do modo Activate, podemos seleciona uma ou mais planinhas. Exemplo:

*Worksheets.Select (worksheets.count)*

**Objeto range**

É um comando para representar intervalos, contendo uma ou mais células.

Para se estabelecer o intervalo do Range é necessário estabelecer uma célula de início e uma de fim.

**Propriedades**

**Cells** – É uma representante de todas as células dentro de uma range.

**Column e row** – Indicam a coluna e a linha, respectivamente que formam uma range, canto superior esquerdo. Agem como coleções, ou seja, armazenam informações de coluna e linhas de uma ranger. Exemplo:

*Worksheet\_selectionChange (ByVal Target as Range)*

*MsgBox “coluna inicial selecionada: “& Target.Column*

*MsgBox “Linha inicial selecionada: “ & Target.Row*

**Fórmula** – Propriedade retorna ou determina qual a fórmula contém em determinação endereço. Exemplo:

*For Each Celula In ranger (“aD3:D8”)*

*MsgBox Celula.Formula*

**Value** – Retorna ou define o valor de uma determinada célula do range. Exemplo:

*Celula.Value = “valor” & i*

**Métodos**

**AddComment –** Adiciona um texto de comentário a uma célula range.

**AutoFilter** – Habilita o recurso de autofiltro do Excel nas células que envolvem o range.

**AutoFit** – ajusta colunas ou linhas para melhor disponibilização de conteúdo no range conjuntura do conhecimento que o autor aborda, alguns teóricos chamam de ‘estado da arte do saber’, é obrigatório o uso de referências para que o leitor saiba quais as correntes de pesquisa foram utilizadas pelo autor para o desenvolvimento deste novo conhecimento.

**CONSIDERAÇÕES FINAIS**

Conclui-se então que os objetos usados no excel são ferramentas importantes no auxílio, edição de programa e resolução de problemas dentro da programação VBA. Você deve identificar um objeto antes de aplicar um dos métodos do objeto ou alterar o valor de uma das suas propriedades.

Contendo mais de 120 tipos diferentes de objetos presentes para auxiliar o programador é possível através deles temos acesso a diversas funções, sendo assim, o programador estará sempre manipulando um objeto. Vimos as funções e utilidades de cada objeto e como podem ser usados, como por exemplo o objeto workbook que contém todos os objetos workbooks abertos nele. Em resumo, todos os aplicativos possuem uma maneira de retornar os objetos que possui.

**REFERÊNCIAS**

[https://learn.microsoft.com/pt-br/office/vba/excel/concepts/miscellaneous/concepts-excel-vba-reference](https://learn.microsoft.com/pt-br/office/vba/excel/concepts/miscellaneous/concepts%EF%BF%BEexcel-vba-reference);