Ambientes Visuais

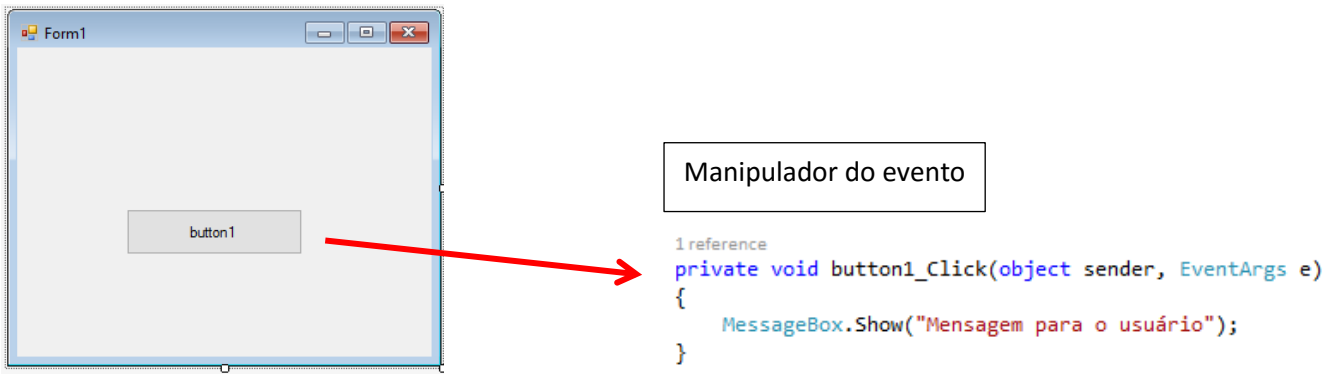
Aula 03

Objetivo: Entender a manipulação básica dos eventos e trabalhar com os componentes CheckBox e RadioButton.

Manipulação de Eventos básica

As GUI são dirigidas por eventos. As integrações normalmente incluem mover ou clicar com um mouse, clicar em um botão, digitar em uma caixa de texto, selecionar um item em um menu ou fechar uma janela. Para todos esses eventos utiliza-se os métodos manipuladores de eventos e executam a tarefas quando um determinado evento e acionado.

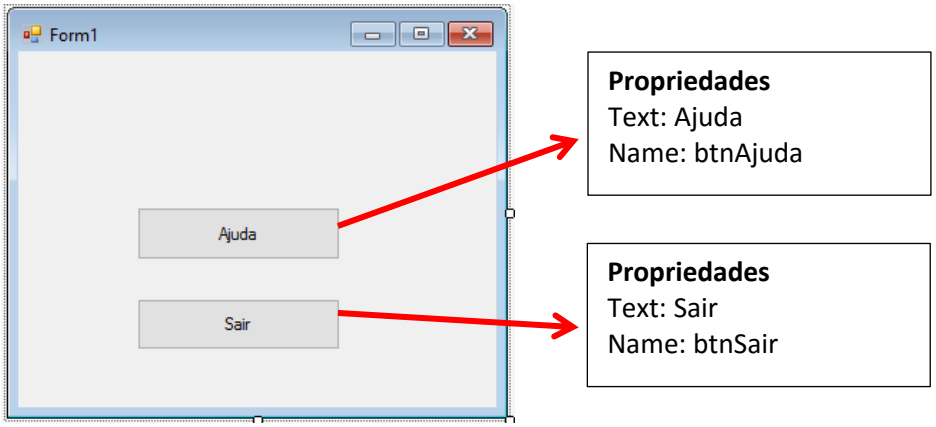
Como exemplo, para exibir uma caixa de mensagem assim que o usuário clicar sobre o botão temos é necessário a declaração do seguinte método manipulador do evento click:



Quando um evento e criado com a ajuda do VisualStudio (com dois cliques sobre o componente ou na barra de propriedades e eventos) o método é nomeado seguindo o padrã *NomeDoControle_NomeDoEvento* no nosso exemplo o controle é o button1 e o evento é o Click.

Os manipuladores de eventos recebem duas referências a objetos. A primeira para o objeto que lançou o evento (sender), e a segunda é uma referencia para um objeto argumentos de evento (e). O argumento (e) é um tipo de EventArgs, uma classe base que contêm informações de evento.

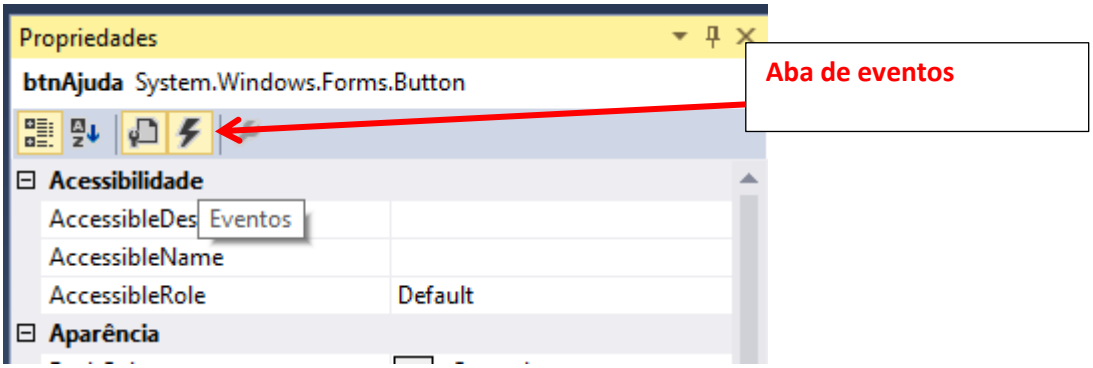
O programador pode criar um método manipulador de eventos e registra-ló para mais de um componente de controle. Como exemplo, vamos adicionar um outro botão ao formulário anterio e renomea-lós com Ajuda e Sair.



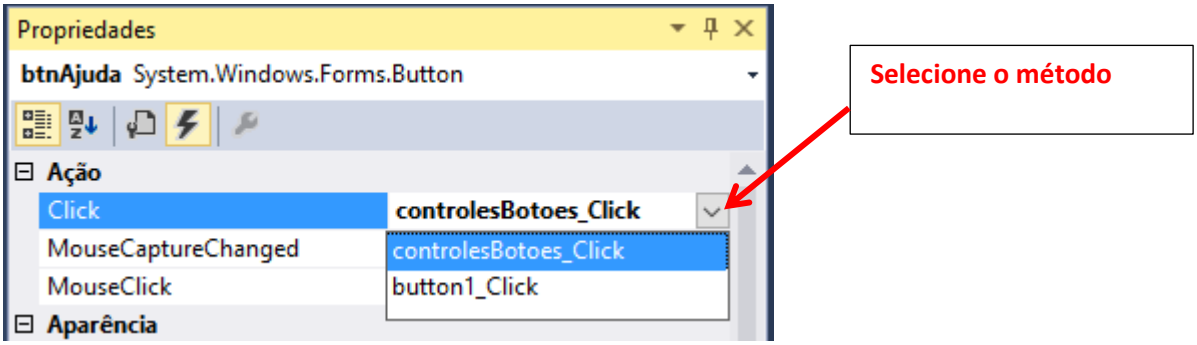
Com o código do nosso manipulador eventos abaixo queremos saber qual botão foi acionado pelo usuário.

```
2 references
private void controlesBotoes_Click(object sender, EventArgs e)
{
    if(sender == btnAjuda)
    {
        MessageBox.Show("Você clicou no Ajuda");
    }else
        if(sender == btnSair)
        {
            MessageBox.Show("Você clicou no Sair");
            Application.Exit();
        }
}
```

Para registrar esse manipulador para os botões, selecionamos o botão Ajuda e na barra de propriedades utilizaremos a aba de eventos.



Para o evento Click, registramos o método que implementamos **controleBotoes_Click**.



O mesmo deve ser feito para o botão Sair.

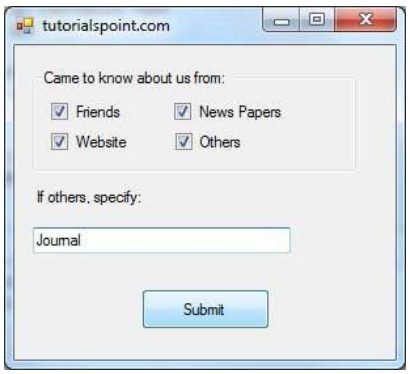
Componentes CheckBox e RadioButton

Os componentes CheckBox e RadioButtons são ditos botões de estado que podem esta no estado de ligado/desligado ou verdadeiro/falso. Os RadioButtons geralmente aparecem agrupados e apenas um deles no gurpo pode estra selecionado em dado momento.

Exemplo:

Já quanto aos CheckBox, não há restrições sobre a quantidade de caixas de seleção o usuário pode marcar.

Exemplo:



As propriedades e eventos comuns a estes componentes são listadas na tabela abaixo.

Prof. Wolley W. Silva

Propriedades	Descrição
Checked	Indica se a CheckBox ou RadioButton foi marcada.
CheckState	Indica se o estado atual dos componentes, marcado ou desmarcado.
Text	Texto exibido à direta dos componentes (rótulo)

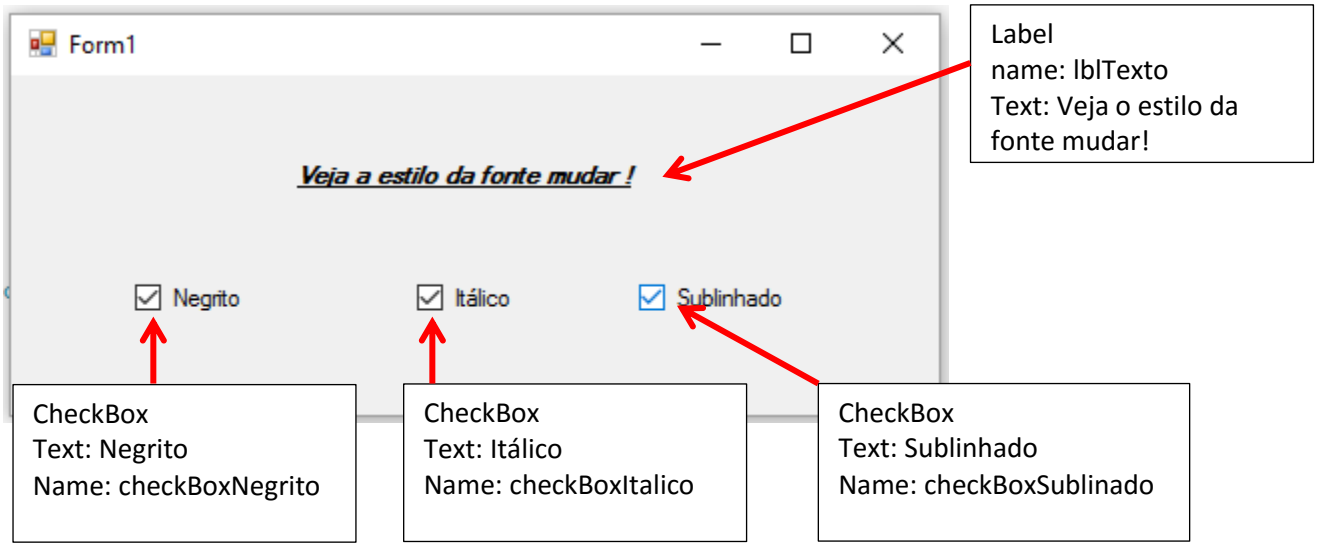
Eventos	Descrição
CheckedChange	Disparado sempre que os componentes são marcados ou desmarcados.
CheckedStateChange	Disparado quando a propriedade CheckState muda.

1. Aplicação de Exemplo – Trabalhando com CheckBox

Vamos implementar a seguinte aplicação abaixo que permite o usuário mudar o estilo da fonte de um label.

1.1. Crie um novo projeto chamado CheckBox.

1.2. Adicione os componentes na formulário padrão do projeto e altere suas propriedades com indicado abaixo.



1.3. Para o primeiro CheckBox, Negrito, adicione o evento CheckedChanged e implemente como o exemplo abaixo:

```
1 reference
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{


    lblTexto.Font = new Font(lblTexto.Font.Name,
                             lblTexto.Font.Size,
                             lblTexto.Font.Style ^ FontStyle.Bold);


}
```

1.4. Par ao segundo CheckBox, Itálico, adicione o mesmo evento CheckedChanged e implemente como o exemplo abaixo:

```
1 reference
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
    lblTexto.Font = new Font(lblTexto.Font.Name,
                             lblTexto.Font.Size,
                             lblTexto.Font.Style ^ FontStyle.Italic);
}
```

1.5. Para o terceiro CheckBox, Sublinhado, adicione o mesmo CheckedChanged e implemente como o exemplo abaixo:

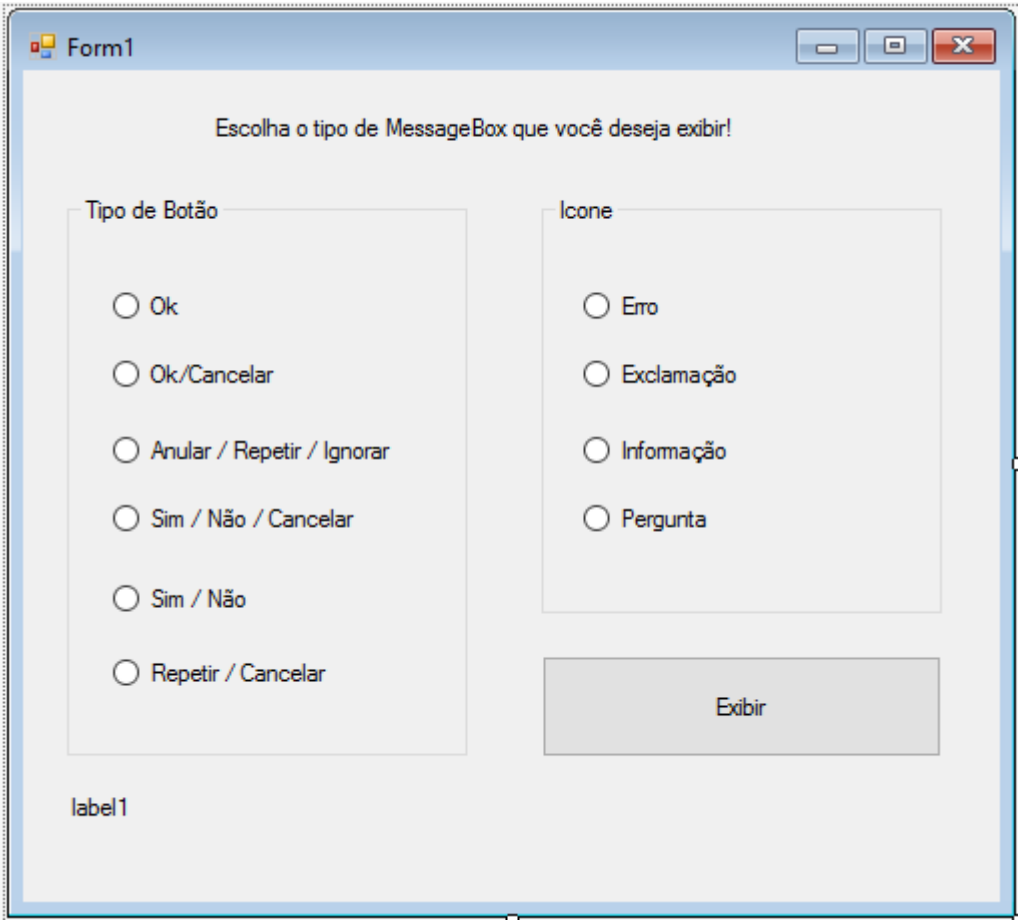
```
1 reference
private void checkBoxSublinado_CheckedChanged(object sender, EventArgs e)
{
    lblTexto.Font = new Font(lblTexto.Font.Name,
                             lblTexto.Font.Size,
                             lblTexto.Font.Style ^ FontStyle.Underline);
}
```

1.6. Execute e teste o funcionando da aplicação.

2. Aplicação de Exemplo – Trabalhando com RadioButton

2.1. Crie um novo projeto chamado RadioButton

A próxima aplicação utiliza botões de rádio para selecionar as opções de uma MessageBox. O usuário seleciona os atributos que deseja e então pressiona o botão exibir.



2.2. Monte o formulário com os componente de acordo com as propriedades descritas abaixo.

O componente GroupBox é utilizada para separar dois grupos de botão o Tipo de Botão e o Icone, assim apenas uma opção de cada grupo poderá ser escolhida.

Os botões do grupo **Tipo de Botão** possuem as seguintes propriedades:

Botão	Propriedades
Ok	Text: Ok Name: rbOk
Ok/Cancela	Text: Ok/Cancela Name: rbOkCancela
Anular / Repetir / Ignorar	Text: Anular / Repetir / Ignorar Name: rbAnularRepedirIgnorar
Sim / Não / Cancela	Text: Sim / Não / Cancela Name: rbSimNaoCancela
Sim / Não	Text: Sim / Não Name: rbSimNao
Repetir / Cancelar	Text: Repetir / Cancelar Name: rbRepetirCancelar

Os botões do grupo **Icone** possuem as seguintes propriedades:

Botão	Propriedades
Erro	Text: Erro Name: rbErro
Exclamação	Text: Exclamação Name: rbExclamacao
Informação	Text: Informação Name: rbInformacao
Pergunta	Text: Pergunta Name: rbPergunta

2.3. Abra o editor de código fonte do formulário e declare como atributo um MessageBoxButtons e um MessageBoxIcon.

```
11  namespace RadioButton
12  {
        3 references
13      public partial class Form1 : Form
14      {
15
16          //declara como atibuto um tipo de botao e de icone
17          private MessageBoxButtons tipoBotao = MessageBoxButtons.OK;
18          private MessageBoxIcon tipoIcone = MessageBoxIcon.Error;
19
```

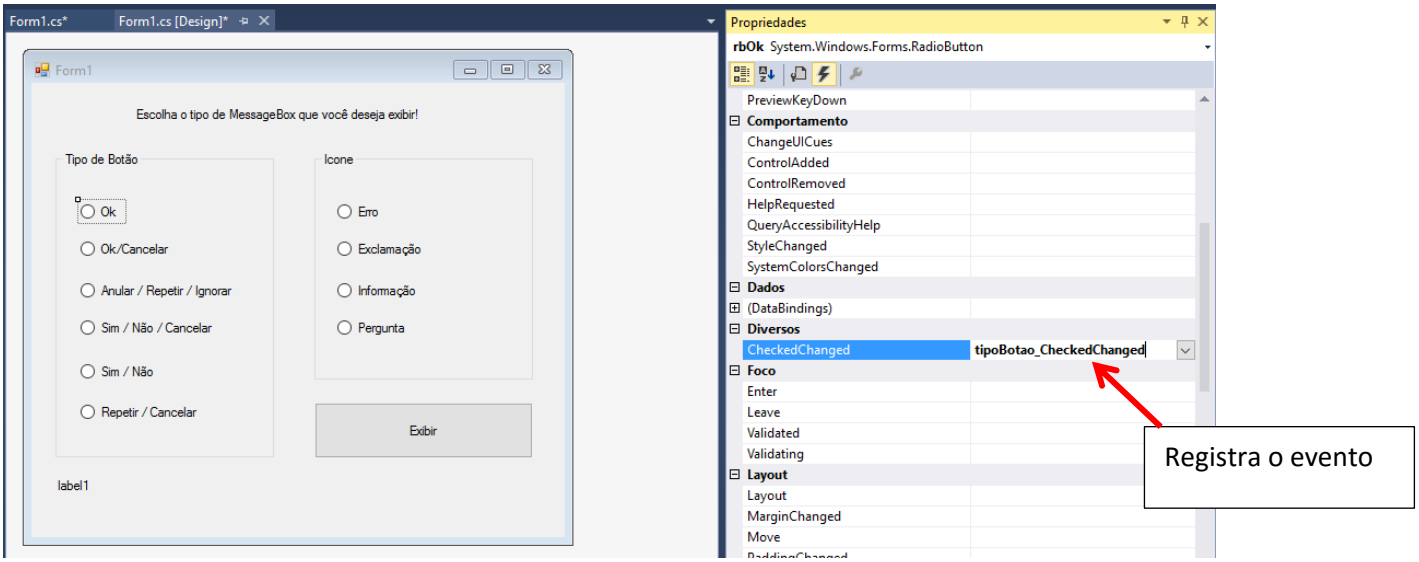
2.4. Ainda no editor de código fonte do formulário, implemente um método manipulador para o evento ChekedChanged que será registrado para todos os radiobuttons do grupo Tipo de Botão.

```
6 references
private void tipoBotao_CheckedChanged(object sender, EventArgs e)
{

    if (sender == rbOk)
    {
        tipoBotao = MessageBoxButtons.OK;
    }else
        if (sender == rbOkCancela)
        {
            tipoBotao = MessageBoxButtons.OKCancel;
        } else
            if (sender == rbAnularRepetirIgnorar)
            {
                tipoBotao = MessageBoxButtons.AbortRetryIgnore;
            }else
                if (sender == rbSimNaoCancelar)
                {
                    tipoBotao = MessageBoxButtons.YesNoCancel;
                }else
                    if (sender == rbSimNao)
                    {
                        tipoBotao = MessageBoxButtons.YesNo;
                    }else
                        tipoBotao = MessageBoxButtons.RetryCancel;


}
```

2.5. Registre este método para todos os radiobutton do grupo Tipo de Botão através da barra de propriedade, no evento CheckedChanged.



2.6. Volte para o editor de código fonte do formulário e implemente o método manipulador de evento dos botões do segundo grupo Icone.

```
4 references
55  private void tipoIcone_ChekedChanged(object sender, EventArgs e)
56  {
57      if (sender == rbErro)
58      {
59          tipoIcone = MessageBoxIcon.Error;
60      }
61      else
62          if (sender == rbExclamacao)
63          {
64              tipoIcone = MessageBoxIcon.Exclamation;
65          }
66          else
67              if (sender == rbInformacao)
68              {
69                  tipoIcone = MessageBoxIcon.Information;
70              }
71              else
72                  tipoIcone = MessageBoxIcon.Question;
73  }
74
```

2.7. Registre este método para todos os botões radiobutton do grupo Icone.

2.8. Crie um evento Click para o botão Exibir e implemente como o exemplo abaixo:

```
1 reference
75  private void btnExibir_Click(object sender, EventArgs e)
76  {
77      //cria a mensagem de acordo com as selecoes
78      DialogResult resposta = MessageBox.Show("Caixa de Mensagem personalizada",
79                                              "Mensagem personalizada",
80                                              tipoBotao,
81                                              tipoIcone);
82      switch(resposta)
83      {
84          case DialogResult.OK:
85              lbSelecao.Text = "OK foi selecionado";
86              break;
87          case DialogResult.Cancel:
88              lbSelecao.Text = "Cancelar foi selecionado";
89              break;
90          case DialogResult.Abort:
91              lbSelecao.Text = "Anular foi selecionado";
92              break;
93          case DialogResult.Retry:
94              lbSelecao.Text = "Repetir foi selecionado";
95              break;
96          case DialogResult.Ignore:
97              lbSelecao.Text = "Ignorar foi selecionado";
98              break;
99          case DialogResult.Yes:
100             lbSelecao.Text = "Sim foi selecionado";
101             break;
102         case DialogResult.No:
103             lbSelecao.Text="Não foi selecionado";
104             break;
105     }
106 }
```

2.9. Execute e teste a aplicação