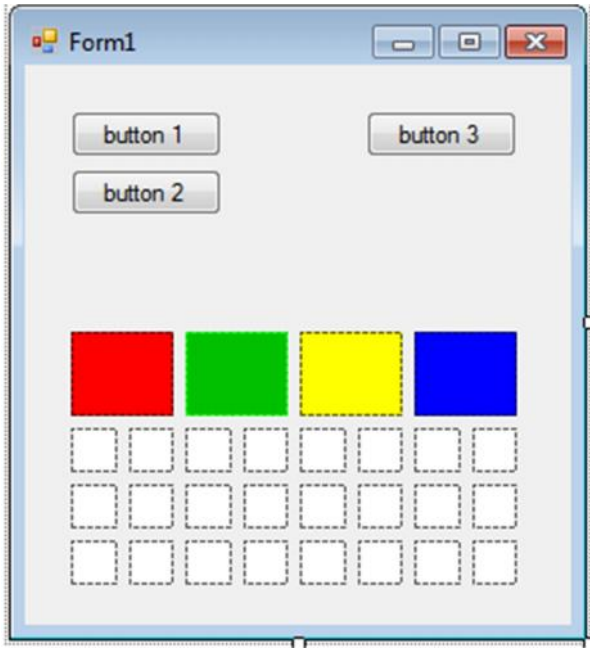
Eventos no C#

Afinal, depois de alguma pesquisa, os eventos são tão simples como no VB.NET.

Comece por criar uma form com 3 botões e 21 paineis:



1. Comece por criar 3 botões, quatro painéis de cor e 24 pequenos painéis, igual ao exemplo. Defina todas as propriedades "Name" como:
    - btn1, btn2 e btn3, panelBlue, panelRed, panelYellow e panelGren e panel1 a panel24 nos mais pequenos.
    - Mude os "Text" dos botões e as "Backcolor" dos painéis de cor. Mude as propriedades "borderAtyle" dos painéis pequenos para "none".

2. **Eventos padrão ou principais ou por defeito.**
    - Se der um duplo clique num qualquer objeto entrará para a codificação do seu evento padrão (principal). E isso faz-se numa função, que no c# e no java se chamam "métodos":
    - Experimente no botão btn1 e depois num painel no panelBlue.
        - No botão o evento padrão é o clique e abre o respetivo método:
          `private void btn1_Click(object sender, EventArgs e)`
            - tem o nome do botão e do evento "Click
            - private significa que só é acessível nesta form.
            - Void já sabe, é o retorno do método (função)
            - Aceita dois argumentos, nos seus parâmetros.
        - No painel o evento padrão é o paint.
          `private void panelBlue_Paint(object sender, EventArgs e)`
            - tem o nome do painel e do evento "Paint"
            - O resto é igual

    - Portanto, todos os objetos têm um evento principal, diferente dos outros objetos e um duplo clique cria o respetivo método, onde podemos codificar as ações que queremos que aconteçam no sequencia desse clique.

- Vamos escrever o código dos eventos padrão do btn1 e btn2
- Duplo clique no btn1 e btn2:

  - Método do btn1

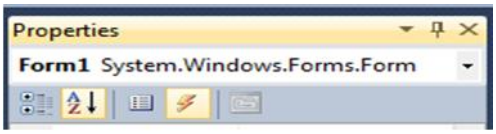
```
private void btn1_Click(object sender, EventArgs e)
{
    MessageBox.Show("olá click");
}
```

  - Método do btn2

```
private void btn2_Click(object sender, EventArgs e)
{
    btn3.Text = "Clica-me";
   // este evento apenas muda o nome do btn3
}
```

- Teste:

3. Eventos alternativos: Queremos fazer uma pequena palete de cores para que o utilizador click num painel e com essa ação, mudar a cor de fundo do btn1. Mas para isso precisamos do evento click e não do Paint que é o evento padrão dos painéis, Como fazer?... Simples:
   - 1º selecionar o objeto no DESIGN (na form). Pode ser o painel mais à esquerda. O azul.
   - 2º escolher o evento a codificar.
     - Na janela das propriedades, em cima. há vários icons;
     - Interessam dois:
       - um com uma lista
       - outro com um raio.



     - A lista contém as propriedades que já conhece.
     - O raio, contém os eventos disponíveis para o objeto selecionado.
       - Se fez o que pedi e selecionou o peinel azul, pode Verificar que a lista de eventos tem o evento **PAINT** pré selecionado.
       - E tem até um nome associado. É precisamente o do método criado com o duplo clique de há pouco.

   - Localize o evento "Click" na sua lista de eventos.
   - Dê-lhe duplo clique e entrará para o seu método onde poderá codificar o que pretende que ele faça, ou seja, que o btn1 receba a cor do painel.

```
private void panelBlue_Click(object sender, EventArgs e)
{
    btn1.BackColor = Color.Blue;
}
```

     - Teste.
   - Vá agora à lista de eventos do painel azul, e observe o nome do método que ficou associado ao evento click. Trata-se do método **panelBlue_Click()** . Cada vez que clica nesse painel, este método vai ser executado.
   - Repita o procedimento para os eventos “click” dos restantes painéis, e faça-os mudar a backColor do btn1 para a cor respetiva.
   - Teste e verifique cada um dos nomes dos métodos registados nos respetivos eventos “click”.

4. Exercício: otimização do código anterior:
   O exercício anterior acabou com 4 métodos, cada um referente ao evento click dos 4 painéis de cor. Vamos avançar e substitui-los por um único método. Observe os métodos do evento click dos painéis.
   Note que o 1º parâmetro de cada daqueles métodos é do tipo **Object** e chama-se **sender**. Isto significa que ao clicar no objeto, ele envia uma cópia de si mesmo para o respetivo método. Bom, então se temos o objeto disponível dentro do método, também significa que podemos aceder às suas propriedades, logo à cor do painel.

   - **Passo1**: Vamos começar por escrever um método que sirva todos os 4 painéis.
     - **Precisamos de um nome mais comum**: panelCor_Click
     - **A acessibilidade é a mesma**: private
     - **O retorno é o mesmo**: void:
     - **Os parâmetros são os mesmos**: (object sender, EventArgs e), pois continuamos a precisar de receber no 1º uma cópia do objeto que dispara o evento click. O 2º veremos nas aulas de setembro.

```
private void panelCor_Click(object sender, EventArgs e)
{
    btn1.BackColor = ((Panel)sender).BackColor;
}
```

     - **Uma só linha: Btn1 recebe a cor do objeto que disparou o evento**
     - **Notar em: ((Panel)sender)**
     - **É a 1ª execução que ocorre e converte o "sender" que é do tipo object, para o tipo Panel.**
       - **Isto é necessário porque object é um tipo que aceita qualquer objeto, todos os objetos, e o problema é que os objetos específicos, como o botão ou o menu, têm propriedades específicas diferentes do painel.**
       - **Usando o tipo Object, definimos que aceitamos qualquer objeto específico naquele 1º parâmetro. Como sabemos que vai ser um painel, para aceder às suas propriedades, só temos de converter e isso faz-se assim: (Panel)sender.**
       - **Este tipo de conversão chama-se "cast", colocamos o tipo de conversão (Panel) antes da var a converter "sender". Da mesma forma Podemos converter um float para int:**
         float a;
         int b = (int)a;
     - **Só falta perceber porque é que há dois parenteses: ((Panel)sender)**
       - **É apenas para forçar que a conversão se execute primeiro, antes de aceder às suas propriedades, caso contrário a prioridade da execução seria aceder às propriedades e só depois é que convertia. Ou seja estaria a aceder à propriedades de object e não de um panel. Pode experimentar tirar os parenteses exteriores para ver.**
     - **Prioridade de execução:**
       - **O código desta linha executa primeiro tudo o que está à direita do sinal de atribuição "=".**
       - **Na direita executa primeiro o que está no mais centro dos parenteses e evolui para fora.**
       - **Só depois é que atribui o resultado à parte esquerda.**
       - **Ou seja: converte o object sender para Panel e acede à propriedade Backcolor que a atribui à propriedade BackColor do botão btn1. Simples não é?**

   - **Passo2**: Falta agora dizer ao evento "click" de cada painel que deve executar este novo método e não os anteriormente criados.
     - **Passar ao modo design e clicar no painel azul.**
     - **Localizar na lista de eventos deste painel, o evento click, que terá associado o método btnAzul_Click.**
     - **Abrir a optionBox do lado direito e selecionar o novo método que já deverá estar disponível na lista.**
     - **Testar o painel azul**
     - **Repetir o procedimento nos restantes painéis e Testar.**
   - **Comentar os 4 métodos originais, que já não são precisos.**

5. Exercício: Efeito 3D em cadeia
Usando a técnica anterior de **múltiplos disparos de eventos com um só método,** vamos usar 24 pequenos painéis para criar um efeito 3D: queremos que a borda de cada painel passe para relevo baixo 3D quando o rato passa sobre ele e passe a borda simples quando o rato sair da sua zona. Isto criará um efeito 3D interessante ao passar com o rato sobre os vários painéis. Para isto vamos precisar de 2 eventos: MouseEnter e MouseLeave para disparar 2 métodos genéricos: um para quando o rato entra na área de cada painel e outro para detetar quando o rato sai dessa área específica.
Pressupondo que já tem os 24 painéis criados e as suas propriedades normalizadas, a ideia é criar primeiro os dois métodos genéricos e depois associa-los de uma só vez aos 2 eventos de todos os 24 painéis.

- **1 – Criar o método MouseEnter para o panel1:**
  - **Selecionar o panel1 e duplo click no avento MouseEnter**
  - **Alterar o nome do método para painel3D_MouseEnter**
  - **O resto fica igual**
  - **Uma só linha de código:**

```
((Panel)sender).BorderStyle = BorderStyle.Fixed3D;
```

- **2 – Criar o método MouseLeave para o panel1:**
  - **Selecionar o panel1 e duplo click no avento MouseLeave**
  - **Alterar o nome do método para painel3D_MouseLeave**
  - **O resto fica igual**
  - **Uma só linha de código:**

```
((Panel)sender).BorderStyle = BorderStyle.FixedSingle;
```

- **3 – Associar de uma só vez estes métodos aos 24 painéis: Modo design**
  - **Usando o rato, selecionar todos estes pequenos paineis de uma só vez;**
  - **Localizar o método MouseEnter e na dropBox, selecionar o método painel3D_MouseEnter.**
  - **Localizar o método MouseLeave e na dropBox, selecionar o método painel3D_MouseLeave.**

Testar:
Notar que apesar de termos 24 painéis, todos os seus eventos MouseEnter e MouseLeave têm que estar associados a estes dois métodos: panel3D_MouseEnter() e panel3D_MouseLeave(), caso contrário não funciona.
Perceba a poupança código e de trabalho que obteve ao concentrar num só método todos as ações daqueles objetos todos. Um código assim torna-se muito mais fácil de manter (alterar e evoluir), porque só tem de um fazer num só método evitando erros de alterar num método e esquecer-se de um fazer noutros.
Questão: podíamos otimizar ainda mais este código? Sim, através de um só método que tivesse um if para detetar a propriedade borderStyle a agir em conformidade.

6. Exercício: Botão que foge:
Neste exercício vamos criar uma pequena diversão: O btn2 tem já um evento click que executa um método com a função de alterar o nome do btn3 para "Clica-me". Bom agora vamos criar um evento e um método para a fuga do botão.

- Localize o evento MouseHover do btn3 e duplo clique:
  - Este evento o método que lhe estiver associado Sempre que o rato entra no espaço do botão,
  - O método simplesmente calculará novos valores para as propriedades top e left do botão: são as distancias ao topo e à esquerda da form. Mais, vamos calcular estes valores aleatoriamente.

```
private void btn3_MouseHover(object sender, EventArgs e)
{
    Random rand = new Random();          // novo objeto aleatório criado
    btn3.Top = rand.Next() % 200;        // Nova distancia ao topo
    btn3.Left = rand.Next() % 200;       // Nova distancia à esquerda
}
```

7. Exercício: Evento de teclas:
   No caso seguinte o evento é disparado quando uma tecla é carregada, mas queremos associa-lo ao botão btn1. Esquisito? pode ser, mas há vantagens.
   - 1 - Em design Selecionar o btn1 e escolher o evento keyDown - > duplo clique. Observar o 2º parâmetro do método: O tipo é "keyargumentArgs" e não o EventArgs. Isto significa que apesar do objeto ser um botão, o evento keyDown irá ouvir as teclas: (key listener). Isto pode ser útil, por exemplo, se quisermos escrever texto numa de várias caixas de texto, apenas se um determinado objeto, como um botão ou um painel, estiver ativo.
   - Escrever o seguinte código

```
private void btn1_KeyDown(object sender, KeyEventArgs e)
{
    MessageBox.Show("olá btn1 keydown diz que a tecla foi:" + " " + e.KeyCode);
}
```

Podemos, em vez desta mensagem, por o texto a ser escrito numa textBox...
Podemos, associar o método ao evento keydown da form, em vez do botão...

A partir daqui, podemos o que quisermos.