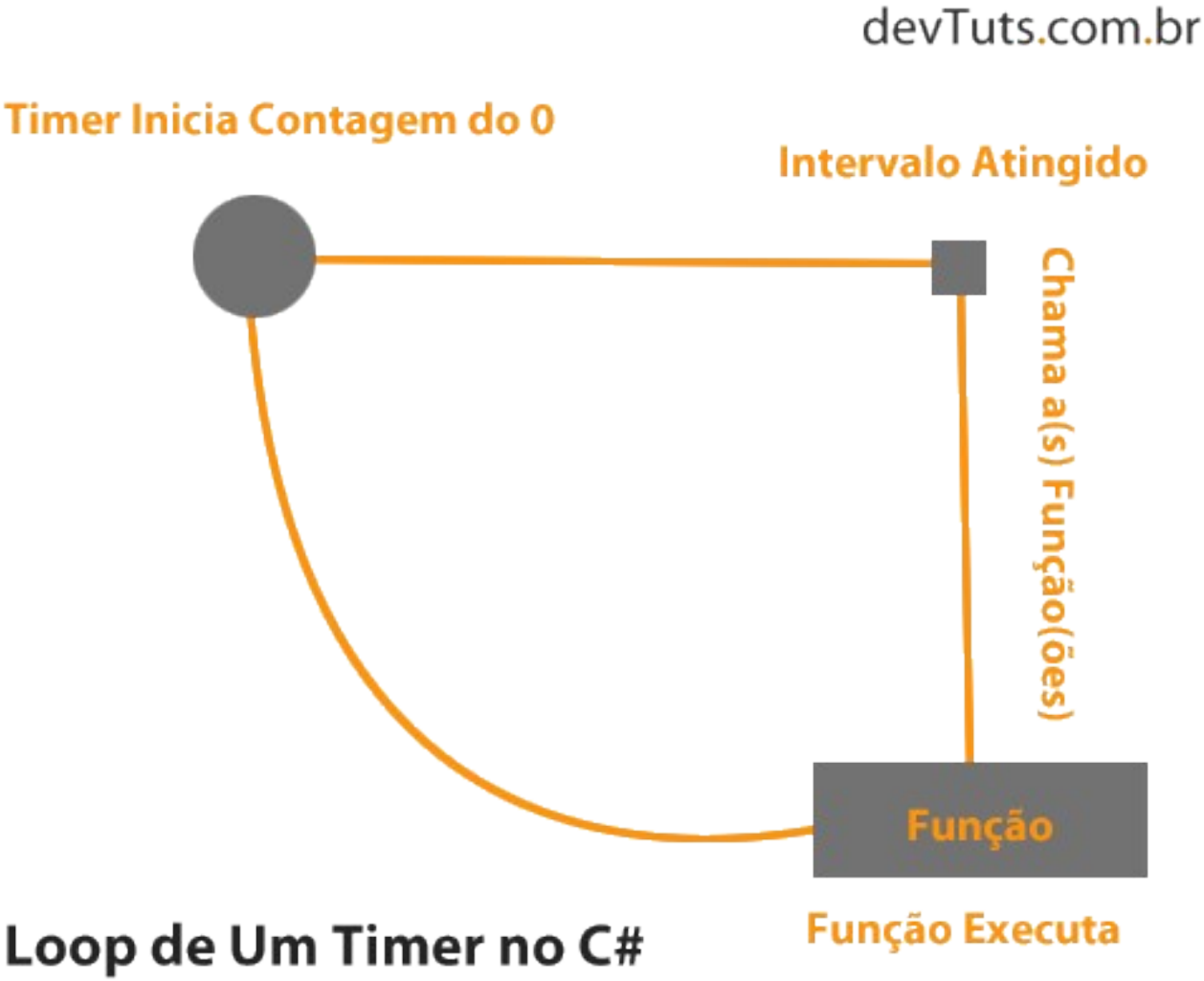
PROGRAMAÇÃO C#
TIMERS

Prof. Alexandre Krohn

FUNCIONAMENTO



IMPLEMENTAÇÃO

.NET Framework fornece 3 classes

- System.Windows.Forms.Timer
  - Utilizado somente com Windows Forms; outro código na mesma thread pode atrasar ou mesmo bloquear os eventos Tick.
- System.Timers.Timer
  - Uma thread é criada quando a classe timer é instanciada (isolada da thread onde está sendo executado o programa); thread-safe
- System.Threading.Timer
  - Não é thread-safe, mas permite interagir com outras threads usando técnicas padrões

IMPLEMENTAÇÃO

	System.Windows.Forms	System.Timers	System.Threading
Eventos Timer roda em qual thread?	UI thread	UI ou Thread de trabalho	Thread de trabalho
Instancias são thread-safe?	Não	Sim	Não
Requer Windows Forms?	Sim	Não	Não
Preciso nos eventos?	Não	Sim*	Sim*
Evento inicial do timer pode ser agendado?	Não	Não	Sim
Classe suporta herança?	Sim	Sim	Não
***Dependendo da disponibilidade dos recursos do sistema (por exemplo, worker threads)**			

SYSTEM.WINDOWS.FORMS.TIMER

SYSTEM.WINDOWS.FORMS.TIMER

Este timer é otimizado para uso em aplicativos Windows Forms e deve ser usado em uma janela

- Limitado a uma precisão de 55 ms

Membros

- bool Enabled
- int Interval
- Start()
- Stop()
- Tick
  - Evento

SYSTEM.WINDOWS.FORMS.TIMER

Criando um timer

```
System.Windows.Forms.Timer meuTimer =
new System.Windows.Forms.Timer();
```

Definindo o intervalo de tempo entre os eventos (em ms)

```
meuTimer.Interval = 2000; // 2 segs
```

SYSTEM.WINDOWS.FORMS.TIMER

Definir a função que deve ser chamada

```
meuTimer.Tick +=
    new EventHandler(meuTimer_Tick);
```

- Note que duplo TAB após digitar += produz o seguinte código

```
void meuTimer_Tick(object sender, EventArgs e)
{
    throw new NotImplementedException();
}
```

SYSTEM.WINDOWS.FORMS.TIMER

Ativando o timer

```
meuTimer.Enabled = true;
```

ou

```
meuTimer.Start();
```

Desativando o timer

```
meuTimer.Enabled = false;
```

ou

```
meuTimer.Stop();
```

SYSTEM.TIMERS .TIMER

SYSTEM.TIMERS.TIMER

Possui maior precisão

Membros

- bool Enabled
- bool AutoReset
- int Interval
- Start()
- Stop()
- Elapsed
  - Evento

SYSTEM.TIMERS.TIMER

Criando e inicializando um timer

```
System.Timers.Timer meuTimer = new
System.Timers.Timer(2000);
```

ou

```
System.Timers.Timer meuTimer = new
System.Timers.Timer();
meuTimer.Interval = 2000;
```

SYSTEM.TIMERS.TIMER

Definir se o intervalo deve ser repetido (default = **`true`**)

- Executar somente uma vez

```
meuTimer.AutoReset = false;
```

- Repetir continuamente (até ser desabilitado)

```
meuTimer.AutoReset = true;
```

SYSTEM.TIMERS.TIMER

Definir a função que deve ser chamada

```
meuTimer.Elapsed += new
System.Timers.ElapsedEventHandler(meuTimer_Elapse
d);
```

- Lembre-se do duplo TAB após digitar +=

```
void corridaTimer_Elapsed(object sender,
   System.Timers.ElapsedEventArgs e)
{
    throw new NotImplementedException();
}
```

e.SignalTime obtém a hora do evento

SYSTEM.TIMERS.TIMER

Ativando o timer

```
meuTimer.Enabled = true;
```

ou

```
meuTimer.Start();
```

Desativando o timer

```
meuTimer.Enabled = false;
```

ou

```
meuTimer.Stop();
```

SYSTEM.TIMERS.TIMER

Acesso aos controles do Windows Forms não é inerentemente seguro para thread

- *Cross-thread operation not valid: Control accessed from a thread other than the thread it was created on*

Solução 1

- Dizer para a thread do timer chame o evento **`Elapsed`** na própria thread da interface

```
meuTimer.SynchronizingObject = this;
```

SYSTEM.TIMERS.TIMER: SOLUÇÃO 2

Chamar o método **`Invoke`**, o qual executará uma função na thread onde o controle foi criado

- Sintaxe
  - Para métodos SEM parâmetros
    - **`Invoke(delegate)`**
  - Para métodos COM parâmetros
    - **`Invoke(delegate, object [])`**

- **`Delegate`**
  - Tipo de referência que define uma assinatura a um método
  - Quando instanciada, um **`delegate`** contém um ou mais métodos
    - Essencialmente um apontador de funções orientado para objetos.

SYSTEM.TIMERS.TIMER: SOLUÇÃO 2

Passos

1. Criar o método que realiza a alteração

```
private void AtualizaComponente(int info)
{
  tbMensagem.Text = info;
}
```

2. Criar o delegate do método (mesma assinatura)

```
public delegate void AtualizaComponenteDel(int info);
```

3. Realizar a chamada do método Invoke (com definição do delegate)

```
tbMensagem.Invoke(new
 AtualizaComponenteDel(AtualizaComponente),
new object[]{120});
```

TRABALHANDO COM TIMER

Cuidado com eventos reentrantes

- Tempo de processamento do evento é maior que o intervalo de tempo entre chamadas
- Problemas com variáveis acessadas por múltiplas threads