Capítulo 10 – Eventos

ISEL/LEIC – Inverno 2011/2012

2.º ano, 2.º Semestre

Nuno Leite

AVE: Ambientes Virtuais de Execução – Semestre de Inverno 2011/12

http://www.deetc.isel.ipl.pt/programacao/ave/

Eventos

- Os Delegates suportam o mecanismo de um objecto poder invocar um método que foi registado no *delegate*
- O .NET suporta a construção de *eventos* (palavra chave *event* no C#) para simplificar a utilização de delegates
- Os eventos são muito utilizados em aplicações baseadas em GUIs
- Ver entrada “Events and Delegates” (e itens relacionados) no MSDN

Implementação de Eventos no MSIL

- public event DelegateType eventName;

- O compilador de C# gera:
  - **private** DelegateType eventName
  - 2 métodos para registo e desregisto no evento
    - add_eventName
    - remove_eventName
  - Membro evento com referências para os métodos

Redefinição do comportamento de registo e desregisto de *observers*

- Para que serve?
  - Sempre que queiramos um comportamento alternativo, por exemplo implementar mecanismos de sincronização *thread-safe*, não suportado completamente na implementação existente por omissão
  - Por questões de optimização de memória versus desempenho
    - Usado num tipo que define muitos eventos

- Como fazer?
  - Definir explicitamente os métodos de adição e remoção de *observers*
  - Definir um campo *delegate* do mesmo tipo que o evento

Exemplo

```
// Especificação do protótipo da função Handler
delegate int Feedback(int value);
// Implementação explícita
class Counter {
        private Feedback handlers;
        public event Feedback FeedbackEvent {
                add { handlers += value; }
                remove { handlers -= value; }
        }
        public void AddHandler(Feedback f) { FeedbackEvent += f; }
        public void RemoveHandler(Feedback f) { FeedbackEvent -= f; }
        public void NotifyHandlers(int n) {
                Console.WriteLine( handlers(n) ); Console.WriteLine(); }
        // Executa handlers por cada iteração de from a to.
        public void DoIt(int from, int to) {
             for (int i = from; i <= to; i++)
                NotifyHandlers(i);     }
}
```

Desenhar um tipo que define muitos eventos

- O tipo System.Windows.Forms.Control define cerca de 70 eventos
  - Se fosse usada a implementação por omissão gerada pelo compilador (que gera os métodos de acesso add e remove, mais um delegate por cada evento)
    - Seriam gerados cerca de 70 conjuntos destes por cada instância de Control
    - Muitos dos eventos provavelmente são usados raramente o que implica um desperdício de memória

Desenhar um tipo que define muitos eventos

- Solução mais eficiente em termos de memória:
  - Redefinir explicitamente os métodos add e remove de cada evento e usar um só campo do tipo *delegate* para armazenar *todos* os eventos (armazenados na _invocationList)
    - De notar que os eventos devem ter o mesmo tipo de delegate na assinatura para poderem ser associados
  - Vantagem: memória ocupada é reduzida
  - Desvantagem: invocação de cada delegate é mais lenta dado que é necessário pesquisar na lista de *delegates*
  - Optimização: usar dicionário que associa tipo de evento a lista de *delegates* que processam o evento