Visão Geral do C#

por Mauro Sant'Anna
MVP Brasil

msdn Fórum Access

O que é C#?

- Um linguagem de programação recente
- Lançada em conjunto com a plataforma .NET
- Completamente nova, sem carregar o "peso" de compatibilidade com versões anteriores
- Moderna, robusta, orientada a objetos e componentes

msdn Fórum Access

C# - A linguagem

- Inspirada no C++
- Desenvolvido em conjunto com o .NET por Anders Hejlsberg
- Atraente a desenvolvedores de outras linguagens que não o VB (C/C++, Pascal etc)
- Usada pela Microsoft para codificar boa parte das novidades da plataforma .NET

msdn Fórum Access

Por que uma nova linguagem?

- O "equilíbrio" que era bom nos anos 70/80 (C) e 80/90 (C++) não é bom hoje
- Os novos hardwares têm novas exigências
- A Internet apresenta novos desafios

msdn Fórum Access

Problemas com outras linguagens

- O gerenciamento de memória (ponteiros/malloc/free) é um bom exemplo:
  - Acesso a ponteiros não alocados causam instabilidade
  - Sujeito a "Vazamento" de memória
- Os problemas acima são desagradáveis em um desktop, mas catastróficos em máquinas que ficam ligadas meses a fio

msdn Fórum Access

Vantagens do C#

- Suporta componentes diretamente
- Tudo em um arquivo (sem .H, .LIB, .IDL, .TLB)
- Gerenciamento de memória automático com "coletor de lixo"

msdn Fórum Access

OOP

- Formato executável (.EXE e .DLL)
- OOP: suporta herança e polimorfismo de classes
- Compatível com outras linguagens

msdn Fórum Access

C# - Baseado no C/C++

- Baseado no C++ quando possível:
  - Declaração de variáveis
  - Declaração de funções
  - Boa parte dos operadores (inclusive +=, ++, ||, &&, !, !=, == etc)
  - Blocos com {}
  - Loops (for, while, do) + foreach

msdn Fórum Access

Diferenças do C++

- Apenas “managed code”
- Tipos, biblioteca de runtime e demais características do .NET Framework
- Não tem argumentos default, union, #include – compilação condicional (#ifdef, #define) presentes

msdn Fórum Access

Diferença do C++ (Cont.)

- Sobrecarga de operadores e operadores de conversão existem, mas com outra sintaxe
- Um único operador . para . -> ::
- Declaração e implementação do código em um só lugar
- Chamadas à API do Windows e ponteiros apenas em código *unsafe*
- Código *unsafe* permite maior performance e integração, sem sair da linguagem

msdn Fórum Access

Além do C++: Idéias

- Linguagem orientada a componentes
- *Tudo* é visto como um objeto
- Robustez
- Fácil documentação
- Preservação do investimento

Idéias: Componentes

- Além do OOP tradicional
- Propriedades, métodos e eventos como conceitos de 1ª categoria
- Attributes e reflections
- Compilação direta de fonte a .EXE ou .DLL, sem intermediários como .OBJ ou .LIB

Tudo é objeto: vatangens e problemas

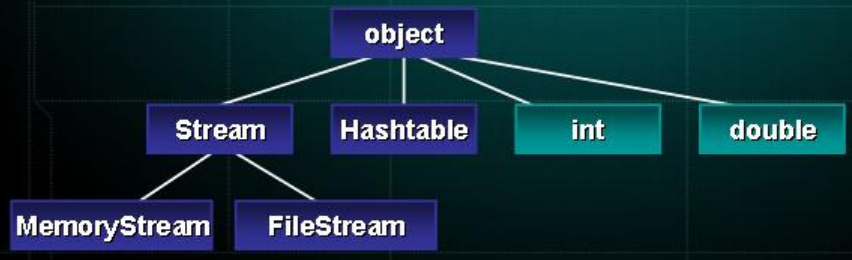
- Problemas tradicionais:
  - Smalltalk, Lisp: Tudo é objeto, mas custo de performance é grande
  - C++, Java, Delphi: Tipos primitivos são "mágicos"e não-OOP

Tudo é objeto: a solução

- C# unifica os tipos sem custo de performance
  - Coleções funcionam com todos os tipos
- *struct* e *enum*, além de inteiros, ponto flutuante e decimal são tipos por valor (baratos)

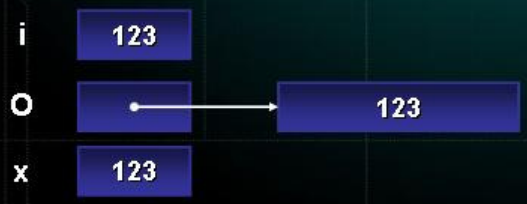
Idéias: Tudo é objeto

- Tudo pode ser visto como um *object*
  - Todos tipos "herdam" de *object*

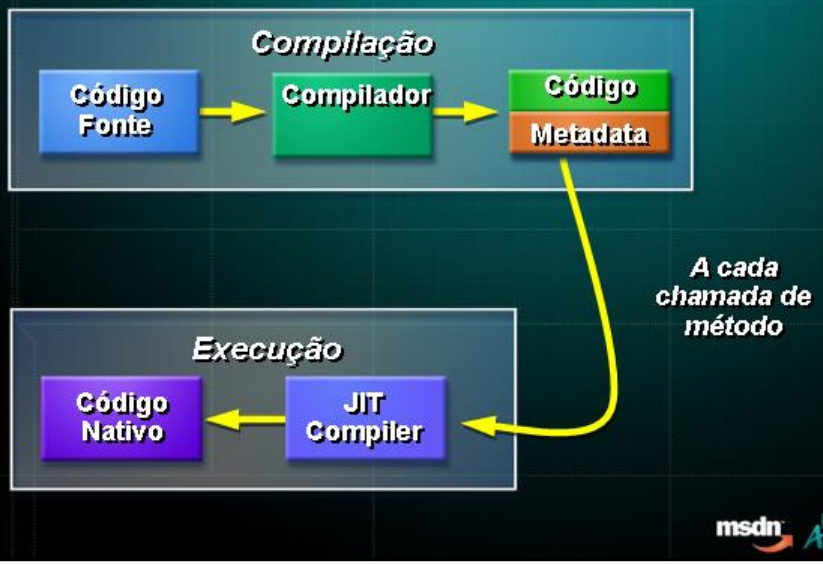


Exemplo de boxing

```
int i = 123;          // Tipo por valor
object O;             // Tipo por referência
O = i;                // Causa "boxing"
string S;
S = O.ToString()      // Chama via O
int x;
x = (int) O;          // Faz "unboxing"
```



Compilação e Execução



MSIL

- Independência de CPU
- Verificações de segurança
- Alta eficiência do código final
- Mantido em cache; a compilação só ocorre na primeira execução

Idéias: Robustez

- Gerenciamento de memória automático com "Coleta de lixo" (GC):
  - Robustez e facilidade de uso
- Exceptions: mecanismo único de tratamento de erros
- Tipos seguros: o sistema de tipos jamais pode ser violado; todos os "casts" são verificados
- Faixas de arrays são sempre validadas
- Intenção do programador deve ser clara

msdn Fórum Access

Tratamento de exceções

```
Try
{
  //..Código que possa gerar uma exceção..
}Catch(Exception ex)
{
  //Aqui é capturado a exceção pela variavel ex do tipo
   Exception
}Finally
{
  //Este bloco sempre será executado, mesmo que seja
   gerado uma exceção.
}
```

msdn Fórum Access

Comentários

```
//Comentário de apenas uma linha

/*Comentário em blocos
de várias linhas*/
```

C#: case-sensitive

```
string strvar = "Primeira String";

string strVar = "Segunda String";
```

Tipos pré-definidos

- Referenciados: object, string
- Com sinal: sbyte, short, int, long
- Sem sinal: byte, ushort, uint, ulong
- Caractere: char
- Ponto flutuante: float, double, decimal
- Lógico: bool

msdn Fórum Access

Operadores

Operadores Aritméticos:

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto

Operadores de Atribuição:

Operador	Descrição	Exemplo
=	Atribuição Simples	Int num = 5;
+=	Atribuição Aditiva	num += 6; //num = num + 6
-=	Atribuição Subtrativa	num -= 9; //num = num – 9
*=	Atribuição Multiplicativa	num *= 2; //num = num * 2
/=	Atribuição de Divisão	num /= 4; //num = num / 7
%=	Atribuição de módulo	num %= 2; //num = num % 2

msdn Fórum Access

Operadores (Cont.)

Operadores Relacionais:

Operador	Descrição	Exemplo
==	Igualdade	if (num == 5) //num é igual a 5;
>	Maior	if (num > 0) //num é maior que 0;
<	Menor	if (num < 10) //num é menor que 10;
<=	Menor Igual	if (num <= 8) //num é menor ou igual a 8;
>=	Manior Igual	if (num >= 10) //num é maior ou igual a 10;
!=	Diferente	if (num != 0) //num é diferente de 0;

Operadores Lógicos:

Operador	Descrição	Exemplo
&&	E	if (num > 5 && num < 10) //num é maior que 5 e menor que 10;
||	OU	if (num > 10 || num < 5) //num é maior que 10 ou é menor que 5;

msdn Fórum Access

Operadores (Cont.)

Operadores de Incremento e Decremento:

Operador	Descrição	Exemplo
++	Incremento	i++ //é o mesmo que (i = i + 1);
--	Decremento	i-- //é o mesmo que (i = i – 1);

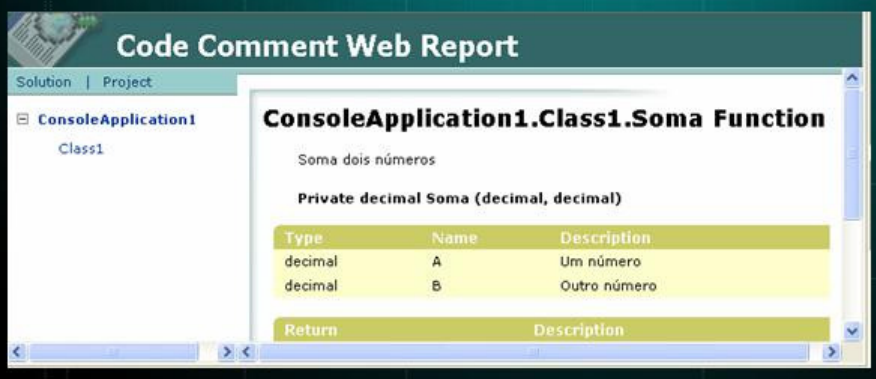
msdn Fórum Access

Idéias: Fácil documentação

```
/// <summary>
/// Soma dois números
/// </summary>
/// <param name="A">Um número</param>
/// <param name="B">Outro número</param>
/// <returns></returns>

decimal Soma(decimal A, decimal B)
{
    return A + B;
}
```

Página Web com documentação



Novidade na versão 2.0

- Suporte a Generics
- Nullable Types
- Iterators
- Partial Classes
- Métodos Anônimos

Idéias: Preservar investimento

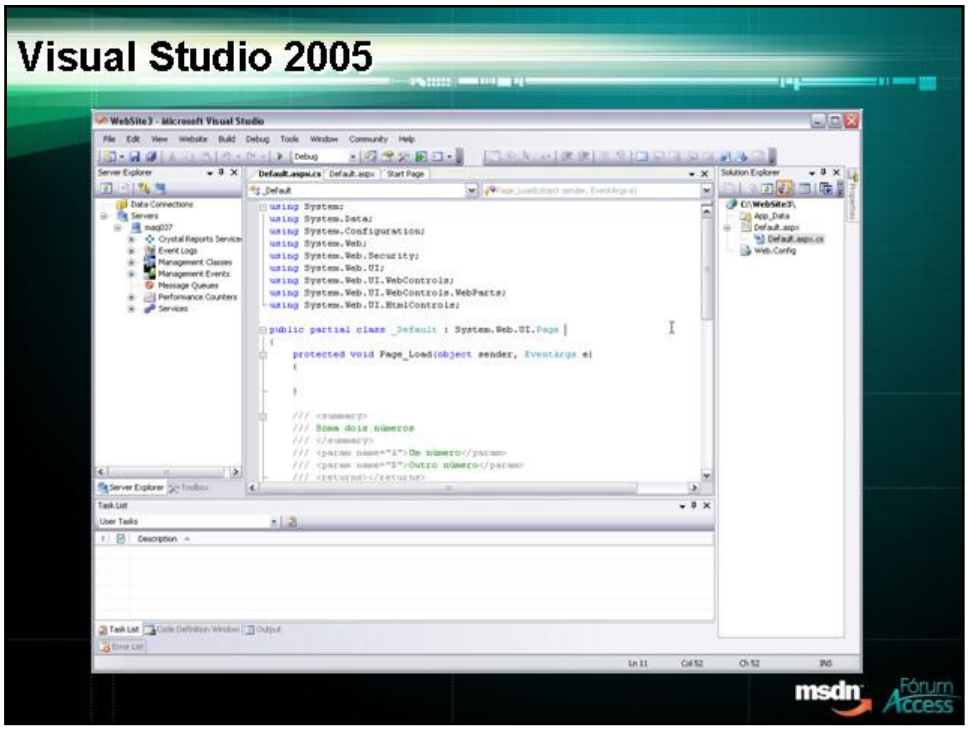
- Baseado no C++
- Muito fácil de aprender para quem conhece C++ ou Java
- Pode ser misturada com código em outras linguagens (C++, VB etc)
- Boa integração com COM/COM+
- Chama DLLs
- Suporta XML, Bancos de dados, SOAP

Alô Mundo

```
public class Alo
{
  public static int Main(string[ ] args)
  {
    System.Console.WriteLine("Alo, mundo\n");
    return 0;
  }
}
```

Padronização

- C#, núcleo do sistema de runtime e da .NET Framework
- Padrão ECMA (http://www.ecma-international.org/) e ISO
- Rotor
  - Windows XP, BSD Unix e Mac OS
  - Com fontes
  - Fins não comerciais


Visual Studio 2005
WebSite3 - Microsoft Visual Studio
msdn
Fórum Access