Ambientes Visuais

Inicialmente, apenas especialistas utilizavam os computadores, sendo que os primeiros desenvolvidos ocupavam grandes áreas e tinham um poder de processamento reduzido. Porém, a contínua evolução e popularização dos computadores provocaram redução dos custos e do tamanho de equipamentos de informática, tornando os sistemas computadorizados acessíveis a um número cada vez maior de usuários. Consequentemente, aumentou também a preocupação com o componente dos sistemas computadorizados que o qual o usuário tem contato, ou seja, a interface. Atualmente, a qualidade da interface influencia a aceitação ou não aceitação dos sistemas computadorizados.

Foram então desenvolvidos os ambientes visuais de desenvolvimento para facilitar a programação de aplicações, aumentando a produtividade do programador e permitindo o desenvolvimento de aplicações com interfaces gráficas amigáveis para interação com o usuário.

Os ambientes visuais de desenvolvimento são também conhecidos como ferramentas de desenvolvimento rápido de aplicações (*RAD - Rapid Application Developement*). Alguns exemplos de ambientes visuais são Visual Studio, Delphi, Eclipse, NetBeans, etc.

O desenvolvimento de ambientes visuais é geralmente baseado em um paradigma conhecido como *paradigma orientado a eventos*. De uma forma resumida, um paradigma de desenvolvimento define como os desenvolvedores enxergam o problema que deve ser resolvido e a solução para este problema (quais abstrações são consideradas no processo de desenvolvimento).

Quando o paradigma é *orientado a eventos*, o foco central do processo de desenvolvimento está relacionado às seguintes questões: "**Quais eventos o usuário do meu aplicativo pode gerar?**" e "**Quais destes eventos o meu aplicativo deve tratar?**". A partir da resposta destas questões, o código para tratamento dos eventos é implementado dentro de procedimentos e funções específicas.

Existem outros paradigmas que podem ser utilizados no desenvolvimento de ambientes visuais e que auxiliam o desenvolvimento de aplicativos mais confiáveis, organizados e com alto nível de reuso, como o *paradigma orientado a objetos*, que aproxima o mundo real do

mundo virtual. A ideia fundamental é organizar o software através de uma coleção de objetos que possuem estrutura (dados/atributos) e comportamento (métodos).

Introdução a Plataforma .Net FrameWork

A Plataforma Microsoft.NET oferece uma alternativa de ambiente para produzir e executar aplicações web, rodando-as em PCs, micros de mão e outros dispositivos, como telefones celulares. O plano da Microsoft é tornar a infraestrutura dessa plataforma amplamente disponível. Tanto que ela já pode ser baixada em seu site e deverá fazer parte das próximas versões do Windows.

A Plataforma .NET é também a principal arma com a qual a Microsoft tenta marcar posição no concorridíssimo mercado dos Serviços Web (Web Services) - nome dado a programas ou componentes que devem ser utilizados na Internet.

Estes serviços on-line são a pedra de toque da Internet, tal como os estrategistas das grandes empresas a imaginam num futuro próximo.

Por meio de serviços web, empresas trocarão informações e farão negócios. Aplicações que rodam num local poderão usar módulos localizados num servidor remoto, consolidando um modelo de computação distribuída. Residentes em servidores web, esses serviços podem fornecer produtos finais - por exemplo, documentos e informações - ou desempenhar tarefas específicas, como realizar cálculos e autenticar transações. Espera-se, assim, que os sites operem de forma integrada, gerando benefícios para empresas e indivíduos. Na essência, essa visão dos serviços web é hoje compartilhada por grandes nomes como IBM, Sun e Oracle, e todos têm iniciativas nessa área.

Uma característica central da Plataforma .NET é aderir aos padrões da Internet sem abrir mão de procedimentos já consagrados no Windows. Para isso conta com o Visual Studio.NET, suíte de programação definida pela Microsoft como "especialmente voltada para a rápida construção e integração de Web Services".

O produto incorpora as linguagens Visual Basic, Visual C++ e Visual C# ("CSharp), todas com sobrenome .NET. Linguagens tradicionais, as duas primeiras sofreram ajustes para a nova plataforma, enquanto o C# começa do zero.

TERMOS DA PLATAFORMA

CLR - Sigla de Common Language Runtime. Base comum a todas as linguagens .NET, o CLR é o ambiente que gerencia a execução de código escrito em qualquer linguagem. Faz parte do Framework.

FRAMEWORK - É o modelo da plataforma .NET para construir, instalar e rodar qualquer aplicação, no desktop ou na Internet. Para executar um programa .NET, é preciso ter o Framework instalado.

IDE COMPARTILHADO - Ambiente integrado de programação (Integrated Development Environment) do Visual Studio.NET. Diferentes linguagens usam o mesmo editor de código e depurador e compilam executáveis na linguagem MSIL. Além das linguagens da Microsoft, já há mais de 20 outras (Perl, Cobol, Pascal, etc) que podem usar esse ambiente.

MSIL - Microsoft Intermediate Language. Quando se compila uma aplicação .NET, ela é convertida para uma linguagem intermediária, a MSIL, um conjunto de instruções independentes de CPU. Na hora de executar o programa, um novo compilador, chamado Just-in-time (JIT) Compiler, o converte para o código nativo, ou seja, específico para o processador da máquina.

MANAGED CODE - Código administrado, ou seja, código escrito para rodar com o runtime do VS.NET. No VS.NET, somente o C++ produz programas que não dependem do runtime, o chamado Unmanaged code.

SOAP - Sigla de Simple Objetc Access Protocol, ou protocolo simples de acesso a objetos. O SOAP é um padrão aberto, baseado em XML, criado pela Microsoft, Ariba e IBM para padronizar a transferência de dados entre aplicações. Pode ser usado em combinação com vários outros protocolos comuns da Internet, como HTTP e SMTP.

UDDI - Iniciais de Universal Description, Discovery and Integration, é uma espécie de páginas amarelas para web services. Na UDDI, empresas expõem seus serviços para que outras possam utilizá-los.

WEB SERVICES - programa completo ou componente de software residente num servidor web.

XML - Sigla de Extensible Markup Language, o XML é uma linguagem baseada em tags semelhante ao HTML. Sua principal característica é a extensibilidade. Quem emite um

documento XML pode criar tags personalizadas, que são explicadas num documento anexo, que tem extensão XSD.

XSD - Sigla de XML Schema Definition. Arquivo associado a um documento XML que descreve e valida aos dados no documento. Assim como as linguagens de programação, os XSDs aceitam dados de diferentes tipos, como números, data e moeda.

XML WEB SERVICES - Blocos fundamentais para a criação de sistemas de computação distribuída na Internet. Um serviço web é uma porção de código localizada num servidor web e que pode ser utilizada por uma aplicação qualquer. O web service pode produzir documentos ou procedimentos. Uma das características centrais dos web services é serem baseados em padrões abertos.

WSDL - Web Service Description Language. Submetida à W3C - o órgão padronizador da Internet. A linguagem WSDL define regras baseadas em XML para descrever serviços web.Relembrar conceitos e recursos básicos apresentados nas aulas anteriores e explorar reutilização de código através da implementação de procedimento e função.

Introdução ao C#

A Microsoft define o C# como a principal linguagem de programação para uso da tecnologia **.NET**. Por ser uma derivação da linguagem **C++**, sem as suas limitações, e é uma linguagem bastante simples de se implementar.

Tipo de Dados

- **Boolean** – Tipo lógico

Este tipo aceita apenas dois valores: false (falso) ou true (verdadeiro).

Exemplo:

```
bool ativo = true;
bool inativo = false;
```

Alguns tipos inteiros, tamanho e seus intervalos:

Tipo	bits	Intervalo
sbyte	8	**-128** até **127**
byte	8	**0** até **255**
short	16	**-32768** até **32767**
ushort	16	**0** até **65535**
int	32	**-2147483648** até **2147483647**
uint	32	**0** até **4294967295**
long	64	**-9223372036854775808** até **9223372036854775807**
ulong	64	**0** até **18446744073709551615**
char	16	**0** até **65535**

Tipos Reais (pontos flutuantes e tipos decimais):

Tipo	bits	Precisão	Intervalo
float	32	7 dígitos	**1.5×10^{-45}** até **1.5×10^{38}**
double	64	15-16 dígitos	**5.0×10^{-324}** até **1.7×10^{308}**
decimal	128	28-29 decimal	**1.0×10^{-28}** até **7.9×10^{28}**

Operadores:

Categoria	Operador(es)	Associação
Primário	(x) x.y f(x) a[x] x++ x— new typeof sizeof checked unchecked	**left**
Unário	+ - ! ~ ++x --x (T)x	**left**
Multiplicidade	* / %	**left**
Aditivo	+ -	**left**
Substituição	<< >>	**left**
Relacional	< > <= >= is	**left**
Igualdade	== !=	**right**
AND Condicional	&&	**left**
OR Condicional	||	**left**
XOR Condicional	^	**left**

Vetores

Trabalharvetores é muito similar na maioria das linguagens de programação. No C# é simples declarar e atribuir valores a vetores.

A sintaxe para a declaração de um array é bem simples, coloca-se o tipo desejado e em seguida os colchetes abrindo e fechando, o nome da variável e a alocação de tamanho do vetor.

Exemplos de declaração:

```
//Exemplo1:
//Declaração
string[] vetNome = new string[3];
//Atribuição
vetNome[0] = “Ana”;
vetNome[1] = “Pedro”;
vetNome[2] = “Maria”;
```

```
//Exemplo2
//Declaração com atribuição
string[] vetNome = {“Ana”, “Pedro”, “Maria”};
```

Comandos Condicionais

if ... else

A condicional mais usada em linguagens de programação é o **if (se)** por ser fácil sua manipulação. A sintaxe desta condicional é:

```
if (condições)
{
    comandos;
}
else if (condições)
{
    comandos;
}
else
{
    comandos;
}
```

switch ... case

Este é um tipo de condicional que verifica várias expressões para uma única comparação. Sua sintaxe é:

```
switch (variável ou propriedade)
{
    case valor1:
        comandos;
        break;

    case valor2:
        comandos;
        break;

    default:
        comandos;
}
```

Comandos de Repetição

Existem vários tipos de comandos de repetição e eles são:

while

Sintaxe:

```
while (condição)
{
    comandos;
}
```

do ... while

O comando **do...while** constrói um laço semelhante ao do **while**, mas ele só verifica a condição depois de executar as expressões.

Sintaxe:

```
do
{
    comandos;
}while (condição);
```

for

Este comando de repetição, diferente do anterior que executa o procedimento várias vezes até que uma condição seja verdadeira, aqui será executado um número previamente determinado de vezes.

Sintaxe:

```
for (variável = valor_inicial, condição, incremento)
{
    comandos;
}
```

for each

Já neste comando de repetição podemos varrer todos os elementos de um vetor seja de objeto ou de tipos primitivos.

Sintaxe:

```
for each (variável_do_tipo in vetor)
{
    comandos;
}
```