PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.COM

EDIÇÃO #22 - NOVEMBRO 2009 ISSN 1647-0710

COMPUTAÇÃO EM GRID

LUA 2ª PARTE : LINGUAGEM DE PROGRAMAÇÃO

O PRÓXIMO PADRÃO DE C++

CAKEPHP SCRIPT BAKE

índice

equipa PROGRAMAR

coordenadores
Joel Ramos
Pedro Abreu

editor
António Silva

capa
Sérgio Alves

redacção
Augusto Manzano
Bruno Oliveira
Francisco Almeida
Jorge Paulino
Luís Dias

equipa de revisão
Bruno Oliveira
Fernando Martins
Miguel Rentes

equipa de divulgação
David Ferreira

contacto
revistaprogramar
@portugal-a-programar.org

website
www.revista-programar.info

issn
1647-0710

Portugal, um país que esquece...

Antes de nos debruçarmos sobre o problema em questão neste editorial , gostaríamos de pedir desculpa aos nossos leitores pelo atraso significativo desta edição. Tentaremos que o mesmo não comprometa a próxima edição; de outra forma, as circunstâncias pouco favoráveis em que a equipa da revista se encontra poderão forçar uma alteração na periodicidade da revista, de modo a evitar os recorrentes atrasos de lançamento.

Passemos agora ao assunto principal do editorial. A revista – desde o início que o nosso projecto tem tornado claro esse objectivo – tem um público-alvo que se alarga a todos os leitores de alguma forma relacionados com a língua portuguesa. Se por um lado, com esta admitida posição de abertura em relação à participação e divulgação fora de Portugal, seria de esperar que a comunidade interveniente neste projecto fosse algo homogénea, podemos também considerar que seria natural que a revista, sendo os criadores e posteriores membros do staff portugueses, fosse também ela maioritariamente composta por portugueses.

«A verdade é que, enquanto aumentamos e alargamos a revista a outras paragens, a participação lusa deveria, no mínimo, manter-se (mantendo-se também, é claro, a natural renovação de colaboradores).»

É o que tem acontecido, ao longo dos já vários anos que conta este projecto: é bem visível que a maioria dos artigos foram escritos por portugueses. A tendência tem, no entanto, sentido alguma regressão nos últimos tempos, com a participação portuguesa a reduzir a olhos vistos. Mas será Portugal um país que esquece? Talvez sim, talvez não. A verdade é que, enquanto aumentamos e alargamos a revista a outras paragens, a participação lusa deveria, no mínimo, manter-se (mantendo-se também, é claro, a natural renovação de colaboradores).

Por outro lado, realçamos a enorme quantidade de feedback positivo que temos recebido do Brasil, e principalmente o espírito proactivo que se tem manifestado por parte dos leitores e redactores de além-mar. Os novos artigos, sugestões e emails de apoio vindos do Brasil que têm chegado frequentemente ao email da revista, orgulham, como líderes e voluntários deste projecto, todo o nosso staff, e lamentamos apenas que a participação portuguesa não tenha, nos últimos tempos, estado ao mesmo nível. Aos novos colaboradores brasileiros, o nosso agradecimento; aos leitores que acham que trariam algo de novo ao envolver-se na revista, a nossa esperança de que o façam. Artigos e sugestões, aceitam-se e querem-se, venham eles de qualquer dos lados do Atlântico.

Joel Ramos e Pedro Abreu

Disponiblização do código fonte do Chrome OS

http://www.chromium.org/chromium-os
http://googleblog.blogspot.com/2009/07/introducing-google-chrome-os.html

O sistema operativo que está a ser trabalhado pela Google, pretende basear-se na famosa utilização da Nuvem, onde as aplicações e muitos documentos estão guardados em servidores e o utilizador lhes pode aceder de qualquer local com acesso à internet.

Lançamento do Windows Seven

Foi lançado, no fim do mês de Outubro o sucessor do Windows Vista, já apoiado por várias companhias, como a Acer, que afirma que: "O Windows 7 vai ajudar a melhorar a má reputação que o Windows tem, quando comparado com o Mac OS".

http://windows.microsoft.com/pt-PT/windows7/products/home

GO! A nova linguagem de Progaramação da Google

http://golang.org/

De notar que após o seu lançamento surgiram dados sobre a existência de uma outra linguagem com o mesmo nome, como se pode ver no link a seguir indicado (http://code.google.com/p/go/issues/detail?id=9), e como tal o seu nome poderá ainda alterado.

Silverlight 4 chega em 2010, e a versão Beta já foi lançada

A aposta da Microsoft, concorrente directa do Adobe Flash, vai chegar no primeiro semestre de 2010. O Silverlight 4 suporta o Google Chrome, entre outras novidades.

http://silverlight.net/

CodeBits - Sapo

Já está marcada a data para a terceira edição do Sapo Codebits, que volta a promover a criatividade e talento, procurando este ano reforçar o conceito de comunidade e colaboração. Nos dias 3, 4 e 5 de Dezembro é este o espírito que vai assentar arraiais na Cordoaria, em Lisboa.

O formato do Codebits mantém-se, com as conferências e apresentações iniciais a serem seguidas de um concurso de programação e hacking que dura 24 horas e de onde saem as ideias que vão depois ser "esmiuçadas" e votadas para escolher um vencedor.

A agenda (http://codebits.eu/s/calendar) já está fechada e contempla cerca de 40 apresentações de 45 minutos, entre oradores convidados e propostas da comunidade que Celso Martinho garantiu serem de elevado interesse e qualidade.

Durante os três dias não falta também espaço de animação, com eventos nocturnos onde pontuam o já "tradicional" Quiz, bandas de música e concursos espontâneos de Guitar Hero. A comida e bebidas são "à borla" e a cada participante é entregue um kit de boas vindas com os

Escolha de cores utilizando Reflection

As cores podem tornar uma aplicação mais agradável e a possibilidade de o utilizador escolher a sua própria combinação de cores, é muitas vezes uma realidade, principalmente nas aplicação comerciais.

Baseando-se nesta permissa, este código pertende mostrar com isto é possível e como é relativamente simples de implementar, usando o Namespace System.Reflection, podendo depois ser adaptado, por exemplo, a um Custom Control, onde o utilizador faria a sua selecção.

```
Imports System.Reflection

Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        ' Cria uma ListView que irá listar os nomes e mostrar as cores
        Dim lv As New ListView
        Me.Controls.Add(lv)
        With lv
            .View = View.Details
            .Dock = DockStyle.Fill
            .Columns.Add("Cores")
            .Columns.Add("Nomes")
        End With

        ' Preenche a ListView com todas as cores pré-definidas
        For Each c As ColorEntry In ColorList.GetColors()
            Dim row As New ListViewItem With {.UseItemStyleForSubItems = False}
            row.BackColor = c.Color
            row.SubItems.Add(c.Name)
            lv.Items.Add(row)
        Next
        lv.AutoResizeColumns(ColumnHeaderAutoResizeStyle.HeaderSize)
    End Sub
End Class

''' <summary>
''' Classe que irá guardar a definição de cada cor
''' </summary>
Public Class ColorEntry
    Public Sub New(ByVal color As Color, ByVal name As String)
        m_color = color
        m_name = name
    End Sub
```

```
    Private m_color As Color = Color.Empty
    ' Propriedade para a Cor
    Public ReadOnly Property Color() As Color
        Get
            Return m_color
        End Get
    End Property

    Private m_name As String = String.Empty
    ' Propriedade para o Nome
    Public ReadOnly Property Name() As String
        Get
            Return m_name
        End Get
    End Property

End Class

''' <summary>
''' Classe que irá criar uma Lista de Cores
''' </summary>
Public Class ColorList
    ''' <summary>
    ''' Preenche a lista de cores com base nos nomes
    ''' </summary>
    Public Shared Function GetColors() As ColorEntry()
        Dim getColorNames As New List(Of String)
        getColorNames.AddRange(ColorNames())

        Dim listColor As New List(Of ColorEntry)

        For Each name As String In getColorNames
            Dim entry As New ColorEntry(Color.FromName(name), name)
            listColor.Add(entry)
        Next
        Return listColor.ToArray()
    End Function

    ''' <summary>
    ''' Lista os nomes de todas as cores
    ''' </summary>
    Private Shared Function ColorNames() As List(Of String)
        Dim listNames As New List(Of String)
        Dim tmpColor As Color = Color.White
        Dim colorType As Type = tmpColor.GetType
        ' Usa Reflection para ir buscar o nome da cor
        For Each propInfo As PropertyInfo In _
                colorType.GetProperties( _
                BindingFlags.Static Or _
                BindingFlags.DeclaredOnly Or _
                BindingFlags.Public)
            listNames.Add(propInfo.Name)
        Next propInfo
        Return listNames.ToList()
    End Function
End Class
```

Que é muitas formas... Uma Crítica sobre a POO

Há alguma discussão em torno da forma mais correta de descrever o conceito em que um objeto pode assumir várias formas de comportamento lógico computacional, previamente definido e codificado. Alguns profissionais da área de desenvolvimento de software com orientação a objetos denominam este conceito com a palavra poliformismo (usada num contexto mais oficioso), outros usam a palavra polimorfismo (usada num contexto mais oficial). Não querendo polemizar o tema, mas sim de apresentar uma explicação que seja mais plausível para melhor entendimento do real conceito que a programação orientada a objetos faz sobre o fato de um objeto poder assumir várias formas e de também justificar o motivo que nesta obra o termo é referenciado como poliformismo e não polimorfismo.

O termo poliformismo é um substantivo masculino que representa a qualidade ou estado de ser capaz de assumir diferentes formas (Dicionário Houaiss), usado para representar conceitos associados as áreas de botânica, química, genética ou zoologia (Dicionário Aurélio e Houaiss), acrescenta-se ainda a área de fisioquímica (Dicionário Houaiss). Assim sendo deve-se considerar que no contexto: botânico refere-se ao fato de existir órgãos ou plantas com diversas formas; químico refere-se ao fenômeno apresentado por substâncias que cristalizam em diferentes sistemas; genético refere-se a ocorrência simultânea, na população, de genomas que apresentam variações nos alelos de um mesmo lócus, resultando em diferentes fenótipos, cada um com uma freqüência determinada; zoológico refere-se a propriedade que certas espécies de animais têm em apresentar formas diferentes de acordo com a função que desempenham em seu habitat (Dicionário Aurélio) e fisioquímico refere-se a propriedade que alguns elementos químicos possuem de se apresentar com formas e propriedades físicas diferentes, como densidade, organização espacial ou condutividade elétrica, como ocorre com o grafite e o diamante que são formas alotrópicas do carbono (Dicionário Houaiss).

Tomando a palavra poliformismo e separando-a em poli (muitos) e morfismo que é um sinônimo masculino usado na área matemática, mais precisamente na álgebra moderna, que representa o conceito de um conjunto que aplicado sobre outro conjunto mantém as operações definidas em ambos os conjuntos. Perceba que se esta idéia for aplicada sobre aos conceitos de programação orientada a objetos ter-se-á então algo semelhante e muito próximo ao conceito de herança. Na prática usual o termo polimorfismo está relacionado a algo adquirir durante sua evolução várias formas, algumas vezes contrárias a sua forma natural, gerando até certas deficiências e deformidades.

Considerando-se que o conceito de programação orientada a objetos surgiu na Noruega por volta de meados da década de 1960 e que a partir de então ocorreram muitas formas de interpretar seus conceitos iniciais é fácil entender a confusão que se faz com alguns termos. Deve-se tomar o cuidado de considerar que o termo polimorfismo é aplicado e assim deve ser no contexto das áreas anteriormente comentadas. No entanto, esta palavra não deve ser usada no contexto técnico da programação orientada a objetos. No idioma inglês o termo analogamente usado é polymorphic, onde poly é a tradução literal de poli (são cognatos) e possui em inglês e português o mesmo significado, ou seja, muitos, mas a palavra morphic em seu significado mias simples em inglês significa forma (Compact Oxford English Dictionary of Current English). Assim sendo, morphic em inglês é relativo a forma, mórfico em português é relativo a Morfeu (deus dos sonhos e do sono) ou então relativo a morfina na área de farmacologia (Dicionário Houaiss). Desta forma, as palavras morphic e mórfico são um caso de falso-cognatos e não podem ser possuir o mesmo significado nos idiomas inglês e português.

Perceba que em inglês o termo polymorphism representa exatamente a idéia de um objeto possuir várias formas de comportamento, mas uma tentativa de tradução literal do termo em inglês para o português como polimorfismo faz com que a idéia original contida na palavra seja perdida dando outra interpretação, daí a sugestão aqui em se fazer uso do termo poliformismo.

Dentro do conceito de programação orientada a objetos, um objeto quando definido dentro dos rigores algorítmicos não pode de forma alguma assumir durante seu processamento várias formas indiscriminadas, a não ser formas previamente definidas e esperadas. Um objeto (ou classe) pode possuir múltiplas formas de aplicação, mas estas várias formas são sempre previsíveis e possuem como característica o fato de serem definidas, com a devida antecedência de suas execuções, daí o motivo de não se fazer uso do termo polimorfismo (que pode ocorrer sem a intervenção de terceiros, de forma natural), mas sim utilizar o termo poliformismo (não existente nos dicionários, mas nem sempre um termo técnico é encontrado nos

dicionários por ser uma descrição com propósito bem definido) no sentido de indicar que um mesmo objeto pode possuir muitas formas (formas previamente definidas) de uso, ou seja, ser um elemento próximo ao conceito de multiforme ou aquele que se apresenta sob numerosas formas.
Como dito inicialmente não é objetivo polemizar esta discussão, mas o de proporcionar uma luz sobre sua interpretação e também apresentar uma justificativa do motivo de estar sendo usado nesta obra o termo poliformismo quando tantos outros usam polimorfismo.
Existe muita coisa estranha por ai, as quais ninguém questiona. Só para ilustrar um pouco, vai ai algumas pérolas da área da computação: deletar ao invés de apagar; clicar ao invés de acionar; aspas simples ao invés de apóstrofos, teclar (ou tc) ao invés de simplesmente escrever, digitar ao invés de datilografar, naum ao invés de não, falow onde ninguém fala absolutamente nada e tantas outras.

Augusto Manzano, São Paulo, Brasil

Computação em Grid

Introdução

Assentando fundamentalmente em conceitos de computação paralela e distribuída, a computação em Grid, assim como outras tecnologias que a precederam, foi inicialmente criada para as necessidades computacionais que os cientistas necessitavam para os seus projectos.

A ciência é fortemente baseada em computação e a capacidade de processamento dos computadores actuais, apesar da sua evolução exponencial, continua a revelar-se insuficiente para os objectivos pretendidos pela ciência, como tal a necessidade de partilhar recursos para atingir um objectivo comum, tornou-se uma estratégia fundamental tanto na ciência como a nível comercial.

Não só os computadores evoluíram exponencialmente, mas também as áreas relacionadas com estes obtiveram uma evolução bastante considerável. As redes são um bom exemplo, passando de velocidades na unidade de medida dos megabytes para os gigabytes.

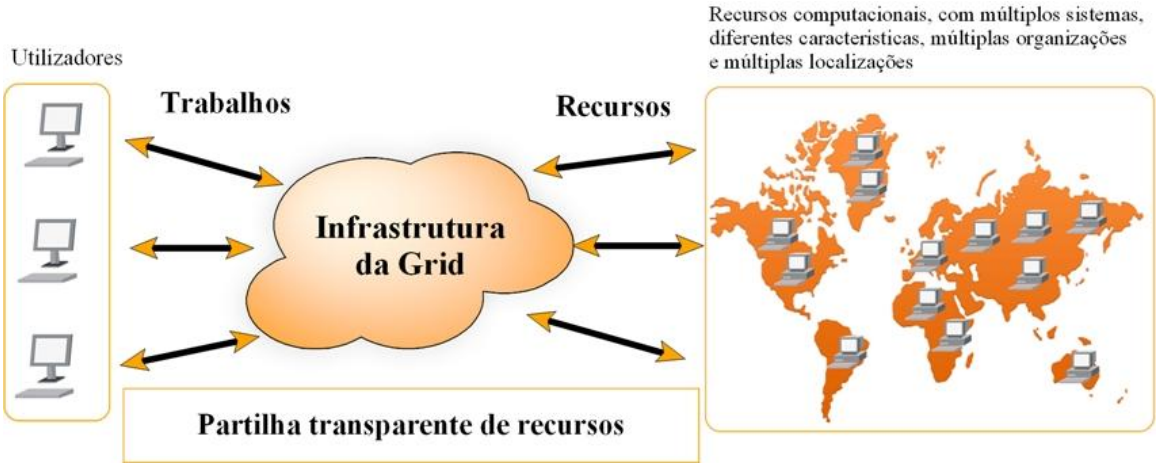
De modo a explorar ainda mais este progresso, o termo Grid oferece um potencial significado para superar obstáculos para problemas ou questões que dificultam o progresso. O conceito assenta na base da cooperação entre máquinas, ou seja, várias máquinas contribuem com os seus recursos para que no conjunto das mesmas, os recursos agregados consigam oferecer garantias em termos de performance.

Estes recursos fornecidos à Grid, podem ser locais, ou recorrendo à internet, abranger uma grande área geográfica. Este tipo de cooperação em grande escala possibilita um enorme potencial em termos computacionais e permite formas de trabalho conjuntas que à alguns anos atrás se considerava impossível.

Um ponto explorado pelo paradigma de Grid, é que as máquinas que um utilizador comum utiliza para as suas tarefas, possuem na sua maioria recursos em excesso, em relação com o que os utilizadores realmente necessitam. Em diversas situações, apenas é utilizada uma pequena parte da capacidade da máquina, o que implica que grande parte da capacidade da máquina se encontre desperdiçada. É também neste desperdício computacional que a computação em Grid tenta obter benefícios, utilizando recursos que a máquina não necessita para uma determinada função que esteja a desempenhar num dado instante. Este tipo de aproveitamento faz com que as máquinas que integram a Grid possuam uma grande versatilidade, e ao contrário de um cluster (um cluster é formado por um conjunto de computadores, que utiliza um tipo especial de sistema operativo, muitas vezes é construído a partir de computadores convencionais (personal computers), que são ligados em rede e comunicam através do sistema, trabalhando como se fossem uma única máquina de grande porte), as máquinas não existem única e exclusivamente para servir o paradigma em que estão inseridas, podendo realizar as tarefas comuns que os utilizadores necessitam, mas aproveitando sempre que possível os recursos não explorados.

A heterogeneidade do conceito de Grid não se limita ao tipo de tarefas que as máquinas podem processar, mas também ao tipo de ambientes em que estas operam, ou seja, existe uma abstracção de arquitectura e software existente em cada máquina, fazendo com que independentemente das suas características cooperem em conformidade sobre uma tarefa em comum.

Grids em grande escala são intrinsecamente distribuídas, heterogéneas e dinâmicas. O seu conceito passa por fornecer grandes capacidades de processamento,



armazenamento e acesso a dispositivos independentemente da sua localização geográfica.

Os utilizadores de uma Grid podem aceder a um simples recurso (um computador, unidade de armazenamento, etc.), ou utilizar vários recursos agregados e coordenados como se trata-se de um computador virtual. Os recursos de uma Grid possuem várias formas de configuração, podendo, por exemplo integrar apenas a Grid quando a máquina se encontra inactiva, não alterando a experiência de um utilizador comum que esteja a usufruir da máquina.

O paradigma da computação em Grid tem como objectivo a integração transparente de recursos de computação que podem pertencer a organizações independentes, encapsulando as suas especificidades. Desta forma, grandes infra-estruturas de computação podem ser criadas a partir de recursos dispersos, que surgem aos utilizadores como um único sistema.

A internet disponibiliza uma camada de serviços que torna possível a partilha de informação independentemente da sua localização. A grid faz algo semelhante permitindo a partilha de recursos computacionais de forma transparente independentemente das suas características e localização.

As implementações de uma Grid normalmente funcionam através da introdução de uma nova camada de software entre as aplicações e a infra-estrutura "física" composta pelo computador, sistema operativo e redes. Assim os utilizadores interagem com os recursos computacionais através desta camada de software que esconde a complexidade e diversidade da infra-estrutura, oferecendo uma interface uniforme para acesso aos recursos disponibilizados. Esta camada de software é designada de Middleware.

O Middleware de uma Grid permite ao utilizador aceder de forma transparente a recursos de computação disponíveis na infra-estrutura de uma Grid, evitando a complexidade de lidar com os detalhes de cada computador.

Vantagens e desvantagens

A computação em Grid é indiscutivelmente uma tecnologia que oferece diversos benefícios na sua utilização. Com a computação em Grid não é necessário despender grandes quantidades de dinheiro num super computador centralizado, podendo utilizar vários computadores dispersos que realizam um trabalho em conjunto. A eficiência de recursos é também uma vantagem considerável, pois para além de o computador poder executar outras tarefas, são aproveitadas as capacidades não utilizadas de uma máquina. Outra possibilidade é utilizar a máquina sobre determinadas condições ou políticas, como por exemplo, pode ser definido que uma máquina só executa trabalhos se o computador se encontrar sem interacção humana sobe um determinado período de tempo.

Outra vantagem significativa é a fiabilidade deste tipo de abordagem, pois estes ambientes são modulares e descentralizados, ou seja não contêm pontos únicos de falha. No caso de uma das máquinas constituintes falhar, o sistema trata de alocar outros recursos e reiniciar aquele trabalho numa outra máquina, sem que para isso ocorra perda de dados.

A adição e a remoção de recursos é um processo muito simples. As máquinas podem ser adicionadas ou removidas da Grid on-the-fly (termo geralmente utilizado para descrever alterações que são efectuados durante o tempo de execução de uma determinada actividade), o que permite uma excelente escabilidade. Os trabalhos podem ser executados paralelamente nas diversas máquinas e os ambientes de Grid estão extremamente bem preparados para executar trabalhos que podem ser repartidos em diversas tarefas mais pequenos e posteriormente executados paralelamente em diversas máquinas.

Diversas áreas estão actualmente a tirar partido da computação em Grid, como a ensino, medicina e aplicações financeiras.

Utilização de Grids

Diferentes organizações necessitam de Grids por diferentes razões. Existem vários tipos de Grids operacionais por todo o mundo:

• Grids nacionais são mantidas por um país. Este tipo de Grid é muito útil em situações de emergência como terramotos ou ataques terroristas. Pode também suportar investigações científicas, estudos meteorológicos e ambientais.

• Grids de projecto são criadas para trabalhos sobre um objectivo específico. São tipicamente construídas para partilhar recursos por um tempo limitado e desenhadas para corresponder às necessidades de grupos de investigação.

• Grids privadas são normalmente chamadas de Grids locais ou Intra-grids, e são utilizadas por instituições como

hospitais ou escolas. Este tipo de Grid é geralmente pequena e gerida centralmente.

• Goodwill Grids são criadas quando voluntários efectuam a doação dos recursos não utilizados da sua máquina, para que assim possam contribuir para a resolução de problemas mundiais como a investigação do cancro.

• Grids peer-to-peer baseiam-se na filosofia "de dar para receber", onde os utilizadores partilham informação com outros utilizadores da Grid

• Cloud-like grids permitem aos utilizadores alugar recursos computacionais pagando aos donos dos computadores que cedem esses recursos. Os serviços oferecidos pela Amazon (http://aws.amazon.com/s3/) e Google (http://www.google.org/powermeter/) são bons exemplos deste tipo de Grids.

Exemplos de Grids

• Compute Against Cancer (http://www.computeagainstcancer.org/) – Este projecto utiliza os ciclos de processamento não utilizados dos processadores dos voluntários para estudar a estrutura e o comportamento de células cancerígenas.

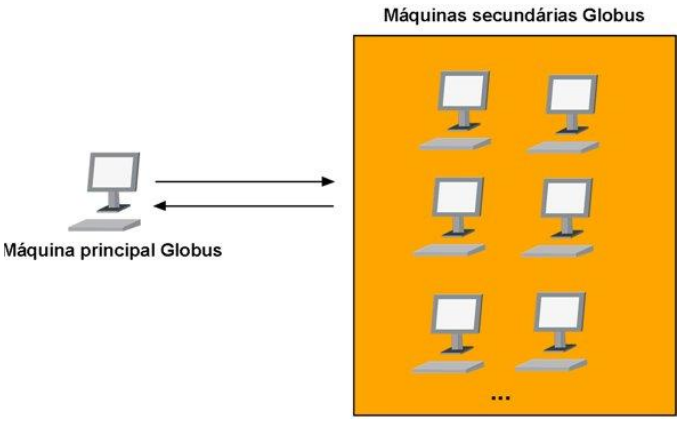
• SETI@home (http://setiathome.ssl.berkeley.edu/) – Este projecto utiliza a computação voluntária para procurar vida inteligente noutros planetas.

• TeraGrid (http://www.teragrid.org)– Sistema de Grid que providencia uma poderosa infra-estrutura para investigação científica. Em 2004 a TeraGrid já possuía 20 teraflops de computação e 1 peta byte de armazenamento distribuído.

Globus Toolkit

O Globus Toolkit é um projecto Open Source que disponibiliza um conjunto de serviços e bibliotecas que suportam Grids e aplicações para Grids. O conjunto de ferramentas inclui software para segurança, informação da infra-estrutura, gestão de recursos, gestão de informação, comunicação, detecção de falhas e portabilidade. O Globus é uma camada de Middleware para Grids com um conjunto de componentes que podem ser utilizados independentemente ou em conjunto no desenvolvimento de aplicações.

Para cada componente, são definidos protocolos e APIs em linguagem C e Java. Uma tremenda variedade de serviços de alto nível, como ferramentas e aplicações têm vindo a ser implementadas com base nestes componentes básicos. Actualmente na versão 4, o Globus Toolkit 4 (GT4) encontra-se não só direccionado no ramo da ciência mas também no ramo comercial.



Desvantagens e desvantagens

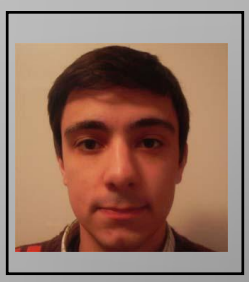
Torna-se importante realçar que o Globus Toolkit inclui muitas outras bibliotecas e serviços destinados a facilitar o desenvolvimento de aplicações para Grids, não só através de Web Services Java mas também recorrendo ás linguagens de programação C e Python. Uma grande desvantagem é a falta de suporte para a plataforma Windows e a falta de suporte em geral para os sistemas Unix mais recentes, sendo necessário proceder à compilação da versão source1, o que muitas vezes se revela uma tarefa demorada e complicada.

Conclusões

Um factor importante a considerar é que tipo de trabalhos pode justificar o uso de uma Grid. Com certeza que executar um trabalho numa Grid que demore apenas um segundo num computador dito "comum" não irá retornar qualquer tipo de benefício. Aquando da distribuição dos trabalhos, é necessário processar muitos outros serviços, como a validação do ficheiro a executar, selecção das máquinas apropriadas, envio do trabalho para a respectiva máquina, etc. Este tipo de tarefas necessárias ao funcionamento do Middleware só efectivamente retornam algum tipo de benefício se o trabalho em questão necessitar de poder computacional considerável. Podemos imaginar um caso muito simples: supondo que era pretendido executar 10 trabalhos que demoram, cada um, 15 minutos a executar sequencialmente numa só máquina. Teoricamente eram necessárias 2 horas e meia (10 tarefas x 15 minutos) para executar este conjunto de trabalhos. No caso de estes trabalhos executarem numa Grid computacional com pelo menos 10 máquinas disponíveis, era possível enviar cada um destes trabalhos para uma máquina diferente, resultando numa execução em paralelo destes 10 trabalhos. Em

condições normais reduziríamos o tempo de 2 horas e meia para 15 minutos, que teriam de acréscimo o tempo de execução dos serviços do Middleware. Este exemplo enquadra-se num cenário em que todas as máquinas envolvidas possuem aproximadamente o mesmo poder computacional.
Se fosse pretendida a execução destas 10 tarefas em apenas 15 minutos numa só máquina, era necessário possuir uma máquina 10 vezes superior ao nível computacional da máquina inicialmente utilizada. No entanto esta máquina com 10 vezes mais processamento não custa só 10 vezes mais que a máquina original. Em termos de custos, a Grid oferece-se uma solução mais económica. Ainda neste exemplo mencionado, se fosse possível dividir estes 10 trabalhos em x sub-trabalhos cada, por exemplo 3 sub-trabalhos, e assumindo que teríamos máquinas suficientes para a execução (neste caso 30) então o tempo de execução deste trabalho iria reduzir, neste exemplo, 3 vezes o tempo, ou seja para 5 minutos.
É possível afirmar que o ganho de computação recorrendo a uma Grid computacional oferece diversos benefícios. Afirma-se como uma solução de baixo custo, que faz uso do trabalho cooperativo entre máquinas existentes numa organização para aumentar o poder processamento disponível, sem que para isso seja necessário investir grandes recursos monetários na aquisição de super computadores.

SOBRE O AUTOR



Bruno Oliveira é estudante do 3o ano de EngenhariaInformática da ESTGF, bem como Trabalhador Independente na área de programação Web. Tem especial interesse pela área de Base de Dados e programação orientada a objectos.

bruno.oliveira@portugal-a-programar.org

Bruno Oliveira

O próximo padrão de C++

```
// Novo código em C++ (Baseado em
Stroustrup, 2006)

template<class T> using Vec =
vector<T,My_alloc<T>>;
Vec<double> v = { 2.3, 1.2, 6.7, 4.5
};
sort(v);
for(auto p = v.begin(); p!=v.end();
++p)
    cout << *p << endl;
decltype(&v) v_ptr = &v;
```

Quem quer que já tenha programado em C++, decerto já possui conhecimento suficiente da linguagem e das bibliotecas padrão para ter familiaridade com os seus múltiplos paradigmas. Objectivamente, o C++ é uma linguagem complexa e muito abrangente: permite facilmente programação procedimental (que muitos chamam programar a la C), permite o uso de polimorfismo, tanto dinâmico como estático, e finalmente, permite também metaprogramação. Na realidade, C++ pode ser igualmente utilizado tanto por iniciados como por peritos, possui um certo grau de flexibilidade, uma sintaxe por vezes particularmente dúbia, e de uma curva de aprendizagem desproporcionalmente íngreme para os programadores mais ambiciosos.

São as imperfeições, as limitações, e as potencialidades sub-aproveitadas da linguagem que levam a Comissão do Padrão de C++ a reformá-la quando necessário. Entre as figuras mais proeminentes desta Comissão, encontra-se por exemplo, o autor da linguagem, Bjarne Stroustrup, bem como diversos profissionais de tecnologias de informação que trabalham com, ou para, diversas empresas que ao longo dos anos têm promovido a linguagem e proposto extensões. Para tornar o C++ uma linguagem mais intuitiva e de mais simples aprendizagem, e ao mesmo tempo manter compatibilidade com código existente, eis algumas das muitas interessantes alterações e novas funcionalidades que nos esperam.

Alterações na sintaxe e inferência de tipos

O C++ é invariavelmente alvo de críticas devido à sua sintaxe e regras de semântica complicadas. No próximo padrão recomendado, haverá algumas novidades significativas e outras de menor impacto. Para começar, podemos mencionar o famoso exemplo de código dado por Stroustrup, que qualquer compilador que respeite o padrão de 2003 rejeita, tornado correcto no próximo padrão. Uma novidade menos importante é a desambiguidade na sintaxe de templates. Ao processar vector<T,My_alloc<T», um bom compilador deverá conseguir discernir entre o operador de right shift » e templates compostos. Na verdade, já existiam muitos compiladores com esta extensão, pelo que incorporar este comportamento nos requerimentos padrão de C++ não deverá constituir qualquer surpresa.

Uma novidade menos importante é a desambiguidade na sintaxe de templates. Ao processar vector<T,My_alloc<T», um bom compilador deverá conseguir discernir entre o operador de right shift » e templates compostos. Na verdade, já existiam muitos compiladores com esta extensão, pelo que incorporar este comportamento nos requerimentos padrão de C++ não deverá constituir qualquer surpresa.

A própria linha de código em que notamos esta pequena conveniência mostra outra bem mais interessante: passará a ser possível usar designações alternativas parametrizadas para templates, da mesma forma que nos habituámos a fazer uso do typedef (que não permitia qualquer forma de parametrização). A palavra-chave using permitirá simplificar a sintaxe para templates de classes que normalmente seriam demasiados complicados e que dificultariam a manutenção do código. Neste exemplo, vector<T,My_alloc<T» passa a poder declarar-se usando Vec<T>. Da mesma forma, using passa a poder ser utilizado em substituição da antiga sintaxe com typedef.

A linha seguinte mostra-nos uma inicialização de um objecto vector que faz lembrar a inicialização de vectores estáticos. No novo padrão, a chaveta {} deverá ser utilizada em detrimento do parêntesis na chamada de construtores de objectos. O motivo por trás desta adição é a necessidade de permitir ao programador diferenciar melhor construtores, funções, e operadores de conversão (que até agora, partilhavam o uso indiscriminado dos parêntesis). Será também possível, através da nova classe STL list_initializer<>, atribuir a qualquer classe a capacidade de inicialização através de vector estático. Note-se que isto já é possível com a ajuda de algumas bibliotecas independentes (por exemplo, a boost utiliza este estilo de construtores em algumas das suas classes), mas o método virá a tornar-se parte da STL.

Finalmente, e ainda parte do mesmo exemplo, é a sintaxe simplificada na declaração do iterator do ciclo for. Com o próximo padrão, será possível contar com o compilador para

deduzir o tipo das variáveis, com as palavras-chave decltype e auto. Ao declarar uma variável usando auto, o compilador verá no lado direito da expressão qual o tipo de variável que o programador tenciona usar. Claro que em circunstâncias dúbias se poderá ainda obter erros, mas em casos convenientes, como o que é dado no exemplo, é escusado estar a escrever explicitamente o tipo do iterador que se espera. Quem quer que esteja habituado a utilizar iteradores com a STL, reconhece de imediato as vantagens da declaração automática de variáveis. O exemplo que se segue mostra-nos também a nova sintaxe alternativa para o ciclo for, com inferência de iterador. Este novo tipo de ciclo for deverá funcionar com vectores estáticos (como o do exemplo) ou contentores STL.

```
int my_array[5] = {1, 2, 3, 4, 5};
for(int &x : my_array) // Equivalente
a um for_each
{
  x *= 2;
}
```

Este novo tipo de ciclo for deverá funcionar com vectores estáticos (como o do exemplo) ou contentores STL.
Alterações a uniões, classes e tipos enumerados

No próximo padrão de C++, as uniões terão regras mais relaxadas de declaração. Por enquanto, não é permitido utilizar dentro de uniões tipos que envolvam construtores não triviais (i.e., construtores de classes com mais de uma variável). No próximo padrão, isto deverá ser possível, tal como exemplificado no seguinte código.

```
class Ponto
{
private:
  double x = 0.0;
  double y = 0.0;
public:
  Ponto() = default; // Construtor
inferido por omissão
  Ponto(double xx, double yy) : x(xx),
y(yy) {}
  Ponto(double dd) : Ponto(dd,dd) {}
  // Operador new proibido
  void *operator new(std::size_t) =
delete;
};

class Coordenada : public Ponto
{
public:
  using Ponto::Ponto; // Construtores
por herança
}
```

```
union exemplo // União com construtor
não trivial
{
  double r;
  Ponto p;
};

enum class Ordinal
{
  primeiro = 1,
  segundo,
  terceiro
};
```

Também patente no exemplo dado acima é a classe Ponto, que mostra novidades significativas que irão tornar a vida mais fácil a muitos programadores no futuro. Utilizando o novo padrão, as classes terão à sua disposição novas facilidades para a sua definição. A primeira novidade é a introdução de inicialização na declaração dos membros da classe. Os membros x e y podem ser inicializados dentro da sua declaração na classe, sem necessidade de utilizar sequer valores de omissão nos construtores. De forma semelhante, igualando um construtor à palavra-chave default, pode-se definir um construtor de omissão como "construtor óbvio de omissão", que dispensa definição. Por oposição, pode-se também proibir certos construtores, operadores, ou destrutores, igualando-os a delete (no exemplo, proíbe-se a alocação dinâmica de objectos Ponto, proibindo o operador new).

Também interessante é a nova capacidade de delegar construtores, definindo um construtor em termos de outro construtor (à semelhança do que já acontece em Java ou C#). Outra nova mecânica de construção, exemplificada na classe Coordenada, é a herança de construção. Através da palavra-chave using, indicamos ao compilador que a classe Coordenada replica todos os métodos de construção de Ponto. Note-se que tirando partido deste estilo de construção, não se podem definir quaisquer outros construtores.

Finalmente, os tipos enumerados passarão a ser mais distintos de inteiros. Adicionando a palavra-chave class à declaração de um tipo enumerado (como exemplificado em Ordinal), podemos também explicitar outros tipos inteiros, tais como long ou char, e mesmo caso seja baseado em int, dizemos ao compilador que qualquer conversão entre inteiros e o novo tipo enumerado declarado é proibida. Isto evita, em particular, os problemas da conversão implícita entre inteiros e tipos enumerados. As constantes das classes enumeradas passarão a habitar no seu próprio namespace (no caso do exemplo, Ordinal::primeiro, Ordinal::segundo, etc.). Note-se que este reforço do tipo é, tal como maior parte das adições à linguagem, opcional.

Novo operador de sufixo

Uma nova adição à linguagem é o novo operador de sufixo, ou melhor, a possibilidade de sobrecarregar o operador de sufixo. O operador de sufixo permite definir símbolos literais para definição automática de tipos ou para unidades arbitrárias. A STL fará uso imediato desta funcionalidade no tipo std::complex. No entanto, a definição de símbolos literais pode também ser utilizada em cálculos científicos, tornando código numérico mais fácil de escrever, ler e gerir. O exemplo seguinte mostra como esta facilidade ajuda a reduzir complexidade na sintaxe.

```
template<typename T>
std::complex<T> operator «» i(T&
numero)
{
  return std::complex<T>(T(),numero);
}

// Exprimir velocidade em unidades de c
double operator «» c(double numero)
{
  return numero*299792458.0; // S.I.
}

//...

const std::complex<double> o = 1.0 +
5.2i;
const double veloc = 0.98c;//
Velocidade
```

Semântica de movimento

Não só se beneficia de escrita mais simples, como também de mais ferramentas para tornar os nossos programas mais eficientes. Variáveis temporárias são a maior contribuição para o excesso de reserva de memória num programa de C++. E isto ainda acontece numa era em que a esmagadora maioria de processadores existentes tem intruções para mover porções arbitrárias de memória directamente entre endereços distintos. Até agora, pudemos contar com compiladores que façam optimizações inteligentes em determinadas circunstâncias para tirar proveito do movimento de variáveis. Com o novo padrão, é possível declarar argumentos de funções ou variáveis usadas temporáriamente, com o prefixo &&. Desta forma, indica-se ao compilador quando deve forçar o movimento de variáveis. Com esta nova semântica, classes de C++ poderão beneficiar de construtores de movimento no futuro, efectivamente transformando objectos, mas com um mínimo de intervenção por parte do programador. Esta semântica pode ser melhor explicada através de um simples exemplo que utiliza a célebre função swap:

```
// Esta função força o movimento de
objectos
template<typename T>
typename
std::remove_reference<T>::type&&
move(T&& a)
{
    return a;
}

class objecto
{
  // ...
  public:
    objecto() = default;
    objecto(const objecto &o); //
cópia
    objecto(objecto &&o); // movimento
};

template<typename T>
void swap(T &a, T &b)
{
  T tmp(move(a)); // move o conteúdo
de a para tmp
  a = move(b); // move o conteúdo de b
para a
  b = move(tmp); // move o conteúdo de
tmp para b
}
```

A função swap do exemplo dado permite efectuar permutação sem cópia, e utiliza um método auxiliar move que força o movimento de variáveis, independentemente de serem chamadas por referência. Note-se que apesar de poupar recursos como visado, este swap baseado em movimentos não é completamente seguro, nem é sempre desejável. Como é fácil imaginar, a semântica de movimento é desejada para optimizações em algoritmos, e o seu desuso potencialmente cria mais problemas que aqueles que supostamente resolve.

Novidades para templates

No próximo padrão, os templates verão a sua utilidade expandida por intermédio dos templates variádicos: da mesma forma que existem funções variádicas, com um número de argumentos que não está pré-definido, templates variádicos aceitam qualquer número, ou tipo, de argumentos.

Tirando partido desta nova funcionalidade, é possível criar templates de funções variádicas, ou mais interessante ainda, com templates de expressões, que abrem mais

possibilidades em metaprogramação com templates. Até agora, o limite imposto pela pré-determinação da parâmetros de templates forçava autores de bibliotecas que façam uso extenso de templates a recorrer a truques de macros e a repetir código para esconder esses limites ao programador. Com templates variádicos, o próprio compilador gera templates com parâmetros repetidos, à medida que estes sejam necessários. De notar que esta iteracão é de um grau diferente da iteração de classes e métodos que a metaprogramação com templates utiliza, e que como tal, permitirá reduzir a quantidade de código necessário.

Uma grande adição ao mecanismo de templates, com um grande impacto prático, seria o novo mecanismo de conceitos, que teria melhorado drásticamente as mensagens de erro obtidas com o mero uso da STL. Infelizmente, a implementação de conceitos gerou grandes complicações de retrocompatibilidade na sua implementação, e como resultado, estes foram adiados para outra ocasião.

Adições à Standard Template Library

A biblioteca oficial do padrão tem acompanhado as restantes adições, desde a criação dos templates. Com o próximo padrão, a STL verá uma colecção de novas classes, adaptadas de bibliotecas independentes (um grande contribuidor é a biblioteca boost), que serão muito úteis. Por exemplo:

• std::thread: Classe que permite lançar uma thread, e que necessita de uma classe funcional (i.e., uma classe com um operador de parêntesis) para definir a sua execução. Para gerir recursos partilhados entre operações entre threads, haverá mutexes (std::mutex, por exemplo), ou caso seja necessário código de extrema eficiência, a STL terá também tipos atómicos para o efeito.

• Tipos atómicos: uma série de classes auxiliares desenhadas para multithreading. Operações, que partilham recursos no tempo entre threads, e que mediante violações de acesso à memória, resultam em mais erros de compilação que erros de execução, poupando assim tempo no teste de software.

• std::tuple: a extensão de maps (pares ordenados de variáveis, em que a primeira é índice da segunda) a várias variáveis. Tal como maps, tuples são ordenados automaticamente. Tornado possível graças a templates variáticos.

• std::regex: Expressões regulares, algo que também já se implementara inúmeras vezes em bibliotecas independentes. A STL finalmente fornecerá a programadores uma implementação padrão de uso simples.

Conclusão

Outras novidades que não mencionámos aqui, mas também importantes para o próximo padrão do C++ incluem: um novo símbolo para o apontador nulo: nullptr. O nullptr tem o seu próprio tipo definido, que permite eliminar muitas ambiguidades, fontes de erros no código, e claro, é mais elegante do que escrever NULL; funções Lambda para programação funcional em C++; modificadores de sobrecarga de métodos em classes derivadas; expressões constantes (constexpr) e literais Unicode para strings.

O novo padrão de C++, até agora apelidado C++0x (uma indicação de que o prazo para a sua definição seria o final da década) terá efectivamente o seu draft final concluído no final de 2009 (à custa de sacrificar algumas das suas inovações mais ambiciosas), mas só será submetido à ISO na primeira metade de 2010. Em média, os padrões oficiais ISO levam cerca de um ano a ser aprovados, o que levará o próximo padrão a finalmente tornar-se "oficial" no início de 2011. Felizmente, os compiladores mais populares (tais como Visual C++, GCC e Borland C++) já incluem há algum tempo as alterações de drafts já conclusivos, como por exemplo o TR1 (Technical Report 1).

Como uma linguagem de programação de sistemas de interesse académico e tecnológico, o C++ evolui a um passo contido, mas de forma mais ou menos consistente com as aplicações que lhe foram delegadas ao longo do tempo. Claro está, a linguagem não existe num vácuo, e a evolução de outras linguagens também influencia a sua. Apesar desta influência, o novo padrão de C++ ainda não adoptará garbage collection, por motivos de complexidade em manter novo código gerido compatível com código determinista, e a gestão automática de memória opcional.

Mais importantes que as alterações drásticas à linguagem, a comissão do padrão de C++ optou por refinamentos à sintaxe que permitem simplificar a escrita e a leitura de código, bem como inclusões importantes à STL. Melhoramentos estes que decerto beneficiarão programadores independentemente do seu grau de destreza com a linguagem.

Referências

• The C++ Standards Committee
(http://www.open-std.org/jtc1/sc22/wg21/)

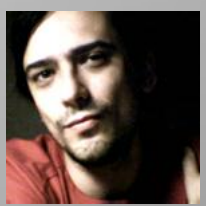
• A Brief Introduction to Rvalue References
(http://www2.research.att.com/~ bs/C++0xFAQ.html#rval)

• The C++0x "remove concepts" decision
(http://www.ddj.com/cpp/218600111;jsessionid=4KNM1QVDB4FGTQE1GHPSKHWATMY32JVN)

• An overview of the upcoming C++ (C++0x) standard
(http://www.youtube.com/watch?v=JffvCivHEHU)

• Wikipedia:C++0x
(http://en.wikipedia.org/wiki/C%2B%2Box)

SOBRE O AUTOR



Francisco Almeida foi Licenciado em Engenharia Física pela Universidade de Lisboa em 2003 e é agora um aluno finalista de Doutoramento de Física pela Universidade Católica de Lovaina, Bélgica. Para além de Ciência, tem interesse em Programação Orientada por Objectos e Algoritmia. A sua linguagem de programação preferida sempre foi o C++, mas também considera Java, C# e D muito interessantes.

francisco.almeida@portugal-a-programar.org

Francisco Almeida

LUA - Linguagem de Programação - Parte 2

No primeiro artigo desta série foram apresentadas informações sobre o uso inicial da linguagem de programação Lua. Foram então abordados pontos relacionados com o desenvolvimento da linguagem, sua aquisição e exemplos de programação sequencial. Neste artigo será enfatizada a programação com tomada de decisão.

Decisões e Condições

Na sua forma mais ampla, decisão é o acto ou efeito de decidir, de deliberar, de julgar, de tomar uma decisão.

Para uma decisão ser tomada esta necessita estar calcada sobre uma proposição, que na esfera da área da computação é tratada como sendo uma condição.

A condição é por sua natureza própria o estado ou a situação de algo. Dentro da esfera computacional pode-se dizer que condição é a *relação lógica entre os elementos* que formam a proposição para a tomada de uma decisão e que a resposta da tomada de uma decisão poderá ser *verdadeira* ou *falsa*.

A relação lógica entre esses elementos ocorre sob dois aspectos:

- Variável versus variável;
- Variável versus constante.

A forma para se estabelecer uma relação lógica permitida é conseguida com o uso de operadores relacionais.

Operadores Relacionais

Os operadores relacionais são ferramentas responsáveis por permitir a definição da relação entre elementos que formam uma condição, à qual será utilizada para a execução do processo de tomada de uma decisão.

A tabela seguinte apresenta os operadores relacionais que podem ser utilizados com a linguagem Lua.

Simbolo	Significado
=	igual a
<>	diferente de
>	maior que
<	menor que
>=	maior ou igual que
<=	menor ou igual que

Os operadores relacionais possuem a mesma prioridade. Assim sendo, não há necessidade de se preocupar em alterar a sua prioridade quando a utilizar numa expressão lógica.

Desvio Condicional Simples

Para se fazer uso de tomada de decisão com desvio condicional simples é necessário fazer uso de instrução:

```
if (condição) then [bloco] end
```

Que possui a sintaxe:

```
if(condição) then
 bloco de acção
end
 instruções executadas após
 condição ser falsa ou após a execução
do bloco de acção
 quando a condição for
 verdadeira.
```

Se condição for verdadeira, então serão executadas todas as instruções definidas na área de bloco de acção que estiver entre **if (condição) then** e **end**. Se condição for falsa o bloco de acção não será executado transferindo a execução do programa para após end.

No sentido de exemplificar este tipo de acção, o programa seguinte efectua a leitura de dois valores inteiros, e apresenta os valores por ordem crescente. Considere então o seguinte código:

```
if(condição) then
 bloco de acção
end
instruções executadas após
condição ser falsa ou após a execução
do bloco de acção
quando a condição for
verdadeira. -- inicio do programa
COND1
 io.write("Entre 1o. valor: ")
 A = io.read("*number")
 io.write("Entre 2o. valor: ")
 B = io.read("*number")
 if (A > B) then
    A, B = B, A
 end
 io.write("Os valores sao: ");
 print(A .. " e " .. B);
```

Pode escrever o código do programa anterior num editor de texto da sua preferência, grave-o com o nome **condicao01.lua** e execute-o através da linha de comandos: **lua 5.1 condicao01.lua**.

Desvio Condicional Composto

Para se utilizar a tomada de decisão com desvio condicional composto é necessário utilizar a instrução:

```
if (condição) then [bloco1] then
```

Que possui a sintaxe:

```
if(condição) then
 bloco de acção 1
else
 bloco de acção 2
end
 instruções executadas após
 condição ser falsa ou após a execução
do bloco de acção
 quando a condição for
 verdadeira.
```

Se a condição for verdadeira, então serão executadas todas as instruções definidas na área de bloco de acção que estiver entre if (condição) then e else, ou seja, as instruções definidas no bloco de acção 1. Caso a condição seja falsa serão executas todas as instruções que estejam definidas na área de bloco de acção que esteja contida entre else e end, ou seja, as instruções que estejam definidas no bloco de acção 2. Após a execução de um desvio condicional composto a execução do programa continua após end.

No sentido de exemplificar este tipo de acção o programa seguinte efectua a leitura de duas notas escolares, calcula a média aritmética das notas e apresenta a mensagem "Aprovado" caso a condição seja maior ou igual a 5. Se a condição for menor que 5 apresenta a mensagem "Reprovado". Junto de cada mensagem será apresentado o valor da média.

```
-- inicio do programa COND2
  io.write("Entre 1a. nota: ")
  N1 = io.read("*number")
  io.write("Entre 2a. nota: ")
  N2 = io.read("*number")
  MD = (N1 + N2) / 2
  if (MD >= 5) then
    io.write("Aprovado, ")
  else
    io.write("Reprovado, ")
  end
  print(string.format("%5.2f", MD))
-- fim do programa COND2
```

Em seguida escreva o código de programa anterior num editor de texto de sua preferência, gravando-o com o nome **condicao02.lua**. Posteriormente execute-o com a linha de comando **lua 5.1 condicao02.lua**.

Desvio Condicional Sequencial

Para se fazer uso de tomada de decisão com desvio condicional sequencial é necessário utilizar a instrução:

```
if (condição) then [bloco1] elseif
(condição2) then [bloco2] ... elseif
(condição N) then [blocoN] else [bloco
```

Que possui a sintaxe:

```
if (condição1) then
 bloco de acção 1
elseif (condição2) then
 bloco de acção 2
 (¦)
elseif (condiçãoN) then
 bloco de acção N
else
 bloco de acção N + 1
end
instruções executadas após
condição sequencial a ser
executada
```

Se condição1 for verdadeira, então serão executadas as instruções que estejam posicionadas na área de bloco de acção entre **if (condição1) then** e **elseif (condição2) then**, ou seja, as instruções definidas no bloco de acção 1. Se condição2 for falsa, então serão executadas as instruções que estejam posicionadas na área de bloco de acção entre **if (condição2) then** e **elseif (condiçãoN) then** e assim sucessivamente. Não sendo as condições verdadeiras, serão executadas as instruções existentes entre **else** e **end** definidas como bloco de acção N+1.

No sentido de exemplificar este tipo de acção o programa seguinte efectua a leitura de um valor numérico entre 1 e 12, correspondente ao nome do mês do ano de um calendário.

```
-- inicio do programa COND3
  io.write("Entre valor: ")
  N = io.read("*number")
  if (N == 1) then
    print("Janeiro")
  elseif (N == 2) then
    print("Fevereiro")
  elseif (N == 3) then
    print("Marco")
  elseif (N == 4) then
    print("Abril")
  elseif (N == 5) then
    print("Maio")
  elseif (N == 6) then
    print("Junho")
  elseif (N == 7) then
    print("Julho")
  elseif (N == 8) then
    print("Agosto")
  elseif (N == 9) then
    print("Setembro")
  elseif (N == 10) then
    print("Outubro")
  elseif (N == 11) then
    print("Novembro")
  elseif (N == 12) then
    print("Dezembro")
  else
    print("Mes invalido")
  end
-- fim do programa COND3
```

Operadores Lógicos

Existem ocasiões em que é necessário trabalhar com o relacionamento de duas ou mais condições ao mesmo tempo para a tomada de uma única decisão. Para esses casos é necessário proceder à utilização dos operadores lógicos.

Os operadores lógicos utilizados com a linguagem Lua são três: **and** (operador lógico de conjunção), or (operador lógico de disjunção) e **not** (operador lógico de negação). Em alguns casos, o uso de operadores lógicos evita a utilização de muitas condições **if...then** encadeados.

O nível de prioridade entre operadores lógicos obedece a seguinte ordem: **or**, **and** e **not**.

O operador lógico de conjunção **and** é utilizado quando dois relacionamentos lógicos de uma decisão necessitam ser verdadeiros para obter um resultado lógico verdadeiro; caso contrário, o resultado do valor lógico retornado será falso. A tabela-verdade para o operador lógico **and**, considerando o uso de duas condições, pode retornar um dos seguintes resultados lógicos:

Condição 1	Condição 2	Resultado
Falsa	Falsa	Falso
Verdadeira	Falsa	Verdadeiro
Falsa	Verdadeira	Verdadeiro
Verdadeira	Verdadeira	Verdadeiro

O operador **or** faz com que o resultado lógico seja verdadeiro quando pelo menos uma das condições envolvidas na decisão for verdadeira, fornecendo um resultado lógico verdadeiro.

No sentido de exemplificar este tipo de acção o programa seguinte efectua a leitura do sexo biológico de um ser humano e apresenta uma mensagem informando se o sexo fornecido é ou não válido..

```
-- inicio do programa COND5
  io.write("Entre seu sexo: ")
  S = io.read()
  if (S == "m") or (S == "f") then
    print("Sexo valido")
  else
    print("Sexo invalido")
  end
-- fim do programa COND 5
```

O exemplo acima demonstra a utilização do operador lógico **or**, que somente permite a apresentação da mensagem "**Sexo válido**", caso o valor fornecido para a variável S seja **m** ou **f**. Qualquer outro valor fornecido apresentará a mensagem "**Sexo invalido**".

O operador lógico **not** é utilizado quando se necessita de estabelecer a inversão do valor de um determinado resultado lógico de uma decisão. É possível obter valores: não verdadeiro e não falso. O operador lógico **not** inverte o resultado lógico de uma condição. A tabela-verdade para o operador lógico **not**, que é utilizado posteriormente a uma condição, pode retornar um dos seguintes resultados lógicos:

Condição	Resultado
Falsa	Não Falso
Verdadeira	Não Verdadeiro

O operador **not** faz com o resultado lógico de uma determinada decisão seja invertido.

No sentido de exemplificar este tipo de acção o programa seguinte efectua a leitura de um valor numérico e apresenta o valor, informado caso este valor não seja menor que 3.

```
-- inicio do programa COND6
  io.write("Entre um numero: ")
  NUMERO = io.read("*number")
  if not (NUMERO <= 3) then
    print( NUMERO)
  end
-- fim do programa COND6
```

Conclusão

Neste artigo foi dado ênfase nas acções de processamento lógico baseado em tomada de decisão por meio de desvio condicional simples, composto e sequencial, além do uso dos operadores relacionais e lógicos.

No próximo artigo será tratado as questões relacionadas ao uso dos laços de repetição.

SOBRE O AUTOR

Natural da Cidade de São Paulo, Augusto Manzano tem experiência em ensino e desenvolvimento de programação de software desde 1986. É professor da rede federal de ensino no Brasil, no Instituto Federal de Educação, Ciência e Tecnologia. É também autor, possuindo na sua carreira várias obras publicadas na área da computação.

augusto.manzano@portugal-a-programar.org

Augusto Manzano

CakePHP - Script Bake

Num mercado de trabalho globalizado devemos utilizar ferramentas que optimizem o nosso tempo de trabalho,principalmente nas rotinas mais repetitivas como por exemplo as CRUD (create, retrieve, update e delete). Neste contexto, o framework CakePHP conta com um óptimo gerador de código dentro do padrão MVC (model/view/controller).

Este artigo demonstra como utilizar o script "bake" e como personalizar o seu comportamento.

Pré-requisitos
1. Servidor Apache + MySQL + PHP a correr (sugestão: XAMPP http://www.apachefriends.org/)

2. Cópia do CakePHP instalada (Versão utilizada neste artigo: 1.2.5 http://cakephp.org/)

3. Conhecimento das convenções utilizadas pelo Cake e o padrão de projeto MVC (http://book.cakephp.org/)

4. Estar cansado de criar formulários para entrada de dados partindo do zero.

Colocando as mãos na massa

Antes de iniciarmos vamos configurar uma variável de ambiente para facilitar o trabalho.

1. Clique com o botão direito do mouse em "Meu Computador" e escolha "Propriedades".
2. Na guia "Avançado" clique no botão "Variáveis de ambiente".
3. Procure pela Variável "Path" e em seguida clique no botão "Editar".
4. Acrescente ao final, separado por ponto e vírgula, o caminho para seu diretório do PHP. No meu caso que tenho o XAMPPLITE instalado: "c:\xampplite\php".

A configuração desta variável de ambiente não é obrigatória mas evita que os scripts do cake não encontrem o php.

Testando o script BAKE

1. Acesse ao prompt de comando do Windows pelo botão Iniciar / Executar. Digite cmd e pressione ENTER.

2. Digite "cd c:\xampplite\htdocs\cake_1.2.5\cake\console" para acessar a pasta de scripts do cake. (lembre-se de substituir "xampplite" e "cake_1.2.5" pela sua pasta onde está localizado o Apache e a sua versão do Cake respectivamente)

3. Execute então o script BAKE digitando "cake bake" e pressionando ENTER. Se tudo correr bem você deverá ver a seguinte mensagem:

```
Welcome to CakePHP v1.2.4.8284 Console
---------------------------------------
---
App : console
Path:
C:\xampplite\htdocs\cake_1.2.5\cake\con
sole
---------------------------------------
---
What is the full path for this app
including the
app directory name?
Example:
C:\xampplite\htdocs\cake_1.2.5\cake\con
sole\
myapp
[C:\xampplite\htdocs\cake_1.2.5\cake\co
nsole\
myapp] >
```

Termine o script com CTRL + C.

Base de dados

Como exemplo, utilizaremos uma Base de dados de tarefas do tipo "To Do List" onde as tarefas podem ser categorizadas. Para isso, a nossa Base de dados "todo" terá duas tabelas: "tasks" e "categories". Utilize o PhpMyAdmin com os comandos SQL abaixo para criar as tabelas:

```
CREATE TABLE IF NOT EXISTS
`categories` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`title` varchar(20) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT
```

```
CHARSET=latin1 AUTO_INCREMENT=4 ;
INSERT INTO `categories` (`id`,
`title`)
VALUES
(1, 'Profissional'),
(2, 'Estudo'),
(3, 'Pessoal');

CREATE TABLE IF NOT EXISTS `tasks` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`title` varchar(50) NOT NULL,
`created` datetime NOT NULL,
`modified` datetime NOT NULL,
`category_id` int(11) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT
CHARSET=latin1 AUTO_INCREMENT=4 ;
INSERT INTO `tasks` (`id`, `title`,
`created`,
`modified`, `category_id`) VALUES
(1, 'Terminar artigo sobre CakePHP',
'2009-10-
14 17:06:24', '2009-10-14 17:06:24',
1),
(2, 'Terminar leitura do livo de
Ajax', '2009-10-
14 17:06:24', '2009-10-14 17:06:24',
2),
(3, 'Cortar cabelo', '2009-10-14
17:06:24',
'2009-10-14 17:06:24', 3);
```

```
---------------------------------------
Database Configuration:
---------------------------------------
Name:
[default] >
Driver:
(db2/firebird/mssql/mysql/mysqli/odbc/o
racle/po
stgres/sqlite/sybase)
[mysql] >
Persistent Connection? (y/n)
[n] >
Database Host:
[localhost] >
Port?
[n] >
User:
[root] >
Password:
>
The password you supplied was empty.
Use an
empty password? (y/n)
[n] > y
Database Name:
[cake] > todo
Table Prefix?
[n] >
Table encoding?
[n] >
```

Pasta do aplicativo

Em seguida, vamos criar a pasta do aplicativo usando o script BAKE. Repita os dois primeiros passos do item "Testando o script BAKE". Após isso, digite: "cake bake -app todo" e responda à pergunta "What is the full path for this app including the app directory name?" informando o caminho completo para a pasta do novo aplicativo. No nosso caso: "c:\xampplite\htdocs\todo". Neste ponto o bake informa que irá copiar o esqueleto de um aplicativo padrão do cake para a sua nova pasta indicada. Confirme a operação respondendo "y" para a pergunta: "Look okay? (y/n/q)" ,e "n" para a pergunta "Do you want verbose output? (y/n)".

Na próxima etapa, o bake irá perguntar sobre a sua Base de dados. Responda de acordo com as suas configurações. As respostas abaixo valem para uma Base de dados MySQL a correr localmente:

nome = "todo"
usuário = "root" e sem senha.

Para finalizar esta etapa responda "y" para a pergunta "Look okay? (y/n)" e "n" para "Do you wish to add another database configuration?". O bake irá atualizar o arquivo "database.php" que fica dentro da pasta "config" de seu aplicativo recém-criado. Neste ponto (para os curiosos que ficam a olhar para o forno a cada 5 minutos para ver se o bolo está pronto) você já pode visualizar o aplicativo gerado pelo bake apontando seu browser para o endereço http://localhost/todo/. Será exibido o conteúdo da página estática home.ctp com alguns alertas e instruções de modificações.

Criando os modelos

Nossa próxima etapa será criar os modelos(models). Execute o script bake informando o aplicativo: "cake bake –app c:\xampplite\htdocs\todo".O bake exibirá o seu menu principal conforme mostrado na página seguinte:

```
Welcome to CakePHP v1.2.4.8284 Console
---------------------------------------
App : todo
Path: c:\xampplite\htdocs\todo
---------------------------------------
Interactive Bake Shell
---------------------------------------
[D]atabase Configuration
[M]odel
[V]iew
[C]ontroller
[P]roject
[Q]uit
What would you like to Bake?
(D/M/V/C/P/Q)
>
```

Escolha "M" para criar os modelos. Na próxima tela o bake irá perguntar qual classe de modelo que deseja criar de acordo com as tabelas da Base de dados.

```
---------------------------------------
Bake Model
Path: c:\xampplite\htdocs\todo\models\
---------------------------------------
Possible Models based on your current
database:
1. Category
2. Task
Enter a number from the list above,
type in the
name of another model, or 'q' to
exit
[q] >
```

Digite 1 para escolher o modelo "Category" (note que o Bake já sugeriu o nome de sua classe

modelo no singular a partir do nome da tabela no plural "Categories"). Em seguida o script bake irá perguntar se deseja informar algum critério de validação para os campos do modelo ("Would you like to supply validation criteria for the fields in your model? (y/n)"). Responda "y" e o bake irá apresentar uma lista numerada com as opções de validação para cada campo do modelo e sugerindo uma opção padrão de acordo com o tipo de campo. Para o campo "id" aceite o valor padrão sugerido como campo sem validação (29 – "Do not do any validation on the field"). Para o campo "title" aceite também o valor padrão sugerido como campo não vazio (20 – "notEmpty").

Na próxima etapa o script irá perguntar se você deseja definir as associações deste modelo com outros. Responda "y". O script já sugere "Category hasMany Task?" baseado nas convenções de campos para a Base de dados. No nosso caso, a tabela Tasks possui a coluna category_id e o bake já associa automaticamente à coluna id da tabela Categories. Responda "y". Responda "n" para as outras associações sugeridas e confirme a criação do modelo (arquivo "category.php" na pasta models do seu aplicativo).

Repita a operação para o modelo Task (2). Informando na parte de associações que "Task belongsTo Category". Neste ponto você pode (na verdade deve!) inspecionar o código gerado abrindo a pasta models de seu aplicativo. Edite os arquivos "category.php" e "task.php" com o seu editor preferido e estude o código gerado.

```
<?php
class Category extends AppModel {
    var $name = 'Category';
    var $validate = array(
        'title' => array('notempty')
    );
    //The Associations below have been
created
    with all possible keys, those that
are not needed
    can be removed
    var $hasMany = array(
        'Task' => array(
        'className' => 'Task',
        'foreignKey' => 'category_id',
        'dependent' => false,
        'conditions' => '',
        'fields' => '',
        'order' => '',
        'limit' => '',
        'offset' => '',
        'exclusive' => '',
        'finderQuery' => '',
        'counterQuery' => ''
        )
    );
}
```

Perceba também que foram criados dentro da pasta "tests\fixtures" e "tests\cases\models" do seu aplicativo, os arquivos para geração de testes. Assunto para outro artigo no futuro.

Criando os controladores

De volta ao menu principal do scrip Bake, escolha a opção "C". Mais uma vez, o bake investiga as suas tabelas da Base de dados e de acordo com as convenções do Cake os seguintes controladores são sugeridos:

```
---------------------------------------
Bake Controller Path       :
c:\xampplite\htdocs\todo\controllers\
---------------------------------------
Possible Controllers based on your
current
database:
1. Categories
2. Tasks
Enter a number from the list above,
type in the name of another
controller, or 'q' to exit
[q] >
```

Escolha 1 para criar o controlador "Categories". E responda as perguntas conforme abaixo:

```
---------------------------------------
Baking CategoriesController
---------------------------------------
Would you like to build your
controller
interactively? (y/n)
[y] > n
Would you like to include some basic
class
methods (index(), add(), view(), edit
())? (y/n)
[y] > y
Would you like to create the methods
for admin
routing? (y/n)
[y] > n
---------------------------------------
The following controller will be
created:
---------------------------------------
Controller Name: Categories
---------------------------------------
Look okay? (y/n)
[y] > y
```

Ao responder "n" na primeira pergunta ("Would you like to build your controller interactively?") você deixa o script gerar tudo automaticamente.

Responder "y" na segunda pergunta ("Would you like to include some basic class methods (index(), add(), view(), edit())?") fará com que o script adicione estes métodos ao seu controlador.

Responder "n" na terceira pergunta ("Would you like to create the methods for admin routing?") o que impede que o script gere os métodos de administração. Esses métodos servem para os utilizadores autenticados no sistema, assunto que foge do escopo deste artigo.

O script cria então o arquivo "categories_controller.php" na pasta "controllers" do seu aplicativo. Abra este arquivo e veja os métodos índex, add, view, edit e delete criados automaticamente. Perceba que os métodos view, edit e delete recebem o argumento "id" como parâmetro pois tratam respectivamente da visualização, alteração e exclusão de uma única categoria já cadastrada. O método índex seleciona todas as categorias e repassa para a visão de mesmo nome. O método add inclui umanova categoria.

Repita a operação para a geração de código do controlador de tarefas (TasksController).

Criando as visões

Escolha a opção "V" no menu principal do script Bake para criar as visões e responda conforme abaixo demonstrado:

```
---------------------------------------
Bake View
Path: c:\xampplite\htdocs\todo\views\
---------------------------------------
Possible Controllers based on your
current
database:
1. Categories
2. Tasks
Enter a number from the list above,
type in the
name of another controller, or '
q' to exit
[q] > 1
Would you like to create some
scaffolded views
(index, add, view, edit) for this
controller?
NOTE: Before doing so, you'll need to
create
your controller and model classes (
including associated models). (y/n)
[n] > y
Would you like to create the views for
admin
routing? (y/n)
[y] > n
```

Quando responde "y" para a primeira pergunta (Would you like to create some scaffolded views (index, add, view, edit) for this controller?) informa ao bake que ele deve criar as visões index, add, view e edit para os métodos do

controlador gerado anteriormente.

Note que não existe uma visão para a exclusão pois é apenas uma pergunta de confirmação e em seguida redireciona para o index. Perceba também o alerta que o Bake fornece, informando que antes de criar as visões é necessário que existam o modelo e o controlador correspondente. Responder "n" para a segunda pergunta (Would you like to create the views for admin routing?) impede a criação das visões correspondentes aos métodos administrativos que não foram criados no controlador.

Repita a operação para as visões de Tasks e...corra para olhar o forno!

Saboreando os resultados

Neste ponto você pode apontar o seu browser para o endereço http://localhost/todo/tasks/ e avaliar os resultados. O método índex do controlador (arquivo tasks_controller.php da pasta controllers) será executado e os dados a serem exibidos são enviados para a visão (arquivo index.ctp da pasta views/tasks). Perceba que a junção com a tabela categories ocorre por utilizarmos o campo "title", um dos campos "automágicos" das convenções do Cake. Outros campos "automágicos" são o created e o updated (do tipo datetime) que guardam respectivamente a data de criação do registro e sua atualização.

Experimente os formulários criando uma nova tarefa (New Task) ou editando uma existente (Action Edit). Veja que os formulários criados pelo Cake já possuem a caixa de seleção pronta com as categorias. Tente incluir uma tarefa sem descrição (title) e veja a validação funcionar.
Interessante também notar que ao listar as categorias (List Categories) e em seguida visualizar uma delas (Action View), você obtém a lista das tarefas relacionadas (Related Tasks).

A cereja no topo do bolo

Na minha opinião só vale a pena utilizar um gerador de código caso possua uma arquitetura que permita interferir no código gerado. O script Bake permite criar templates para que você altere seu comportamento padrão na geração das visões de forma relativamente fácil.
Copie os 4 arquivos (form.ctp, home.ctp, index.ctp e view.ctp) que ficam na pasta cake\console\libs\templates\views para a pasta app\vendors\shells\templates\views. Alterando estes arquivos copiados terá controle sobre como o bake gera as suas visões. Como exercício traduza as mensagens do inglês para o português, em seguida gere novamente as visões e veja o resultado.

Conclusão

O script bake oferece boa flexibilidade para personalizar as visões (views), mas um ponto fraco no código gerado está nos dois arquivos add.ctp e edit.ctp que possuem o mesmo código de formulário. Já o código gerado para os modelos (models) é aceitável. Outro ponto fraco está na geração do código dos controladores que acaba também por ser redundante. Neste caso sugiro que crie uma classe pai de onde todas as classes de controle de seu projeto possam se basear através de herança.

O objetivo deste artigo é despertar a curiosidade dos programadores PHP para o Cake. O código gerado pelo script Bake é um bom ponto de partida para conhecer este framework. A análise do código gerado que com certeza serve como material de estudo junto com o manual do Cake online.

Bom apetite!

SOBRE O AUTOR

Licenciado em Análise de Sistemas e atuando como desenvolvedor desde 1988, Luís Dias possui interesse especial em tecnologias opensource. Natural da cidade de Niterói, Rio de Janeiro e atualmente residindo em Portugal na cidade do Porto, vem atuando como desenvolvedor independente utilizando CakePHP e Joomla em diversos projetos. Nas horas vagas toca bateria em bandas de rock e é fã de cinema.

Luís Dias

Queres participar na Revista PROGRAMAR? Queres integrar este projecto, escrever artigos e ajudar a tornar esta revista num marco da programação nacional?

Vai a

www.revista-programar.info

para mais informações em como participar,
ou então contacta-nos por

revistaprogramar
@portugal-a-programar.org

Precisamos do apoio de todos para tornar este projecto ainda maior...

contamos com a tua ajuda!

Equipa PROGRAMAR

Um projecto Portugal-a-Programar.org