

PROGRAMAR

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

EDIÇÃO #29- JUNHO 2011

ISSN 1647-0710



CONTROLO DE VERSÕES
PARA PEQUENOS E GRANDES
PROJECTOS

A PROGRAMAR

MICROSOFT

BIZTALK SERVER
AOS OLHOS DOS PROGRAMADORES

INTRODUÇÃO

CLOUD COMPUTING
E PLATAFORMA WINDOWS AZURE

LUA

LINGUAGEM DE PROGRAMAÇÃO
PARTE 9

MEF

MANAGED EXTENSIBILITY
FRAMEWORK

VIM

O EDITOR
DE TEXTO

COLUNAS

OPENXML
SDK

VISUAL (NOT) BASIC

FAZER MAL
= RÁPIDO?

CORE DUMP

COMUNIDADES

CERTIFICAÇÕES
MICROSOFT

NETPONTO

EQUIPA PROGRAMAR

Coordenadores

António Silva
Fernando Martins

Editor

Igor Nunes

Design

Sérgio Alves (@scorpion_blood)

Redacção

Augusto Manzano, Caio Proiete,
Fernando Martins, Fábio Domingos,
Nuno Godinho, Pedro Martins,
Ricardo Rodrigues, Sandro Pereira,
Sara Silva

Staff

António Santos, Fábio Domingos,
Jorge Paulino, Marco Marques

Contacto

revistaprogramar@portugal-a-programar.org

Website

<http://www.revista-programar.info>

ISSN

1647-0710

Hacktivismo

Segundo a Wikipédia a palavra hacktivismo vem da junção entre *Hack* e *Activism* (*Hacktivism* em inglês), e nos últimos tempos tem sido bastante usado por muitos. Em cerca de um mês e meio a Sony foi atacada pelo menos quatro vezes, e pelo menos inicialmente o suposto motivo era o hacktivismo, e a Sony chegou mesmo a acusar o grupo Anonymous referiram que nada tinham a ver com o roubo de informação de clientes da Sony.

Ainda no ano passado este grupo de Hackers e Activistas tinha sido notícia pelos ataques perpetrados contra servidores das empresas que tinham retirado o apoio ao Wikileaks, como a Amazon, Visa, MasterCard, PayPal, até o banco suíço PostFinance onde estava a conta de doações do Wikileaks depois de ter cancelado a conta de Julian Assange foi atacado. Também membros do senado Norte-Americano e do Ministério Público Sueco foram atacados com a mesma razão.

Recentemente veio a público que o Departamento de Defesa Norte-Americano pretende avançar com a retaliação aos ataques cibernéticos com armas convencionais, que poderão ir até ao lançamento de mísseis. Sim, leu bem, em último caso poderão ser lançados mísseis contra este tipo de criminosos. Mas a grande questão, é como terão eles a certeza da identidade do criminoso, e quantas serão as vítimas inocentes. Todos nós conhecemos maneiras mais ou menos eficazes de ocultar a nossa identidade completamente na Internet, ou pelo menos tornar a sua descoberta muito mais difícil. Mas o Departamento de Defesa Norte-Americano parece ter uma plena confiança nas suas capacidades para detectar estas origens, o problema será quando surgirem as mortes de inocentes...

O primeiro grande ataque conhecido, foi o worm anti-nuclear **WANK**, em 1989, que atingiu agências americanas como a NASA, o Departamento de Energia, entre outros. Mas não é só lá fora que isto acontece, em 1997, um grupo português atacou sites da Indonésia, incluindo sites governamentais e militares, pelo facto de a Indonésia ter invadido Timor Leste. Esse mesmo ataque foi repetido um ano mais tarde, em 1998. Este foi sem dúvida um dos primeiros grandes ataques massivos na história da internet.

Será que no futuro em vez de armas nucleares ou termonucleares, teremos que temer a Internet por ser a melhor arma de destruição de países? Ou nunca chegaremos a este ponto?

NR: Quero desde já agradecer ao Igor Nunes por ter aceite o cargo de editor da Revista PROGRAMAR, e desejo-lhe um óptimo trabalho.

António Silva

TEMA DE CAPA

- 6 **Git - Controlo de Versões para Pequenos e Grandes Projectos**
Um artigo que descreve funcionalidades do Git, bem como as principais diferenças entre os sistemas de controlo de versões centralizados e distribuídos. **Caio Proiete**

A PROGRAMAR

- 24 **Lua – Linguagem de Programação (Parte 9)**
A continuação de um excelente artigo sobre Lua, uma linguagem de programação pouco conhecida. Nesta nona parte descubra tudo sobre o operador lógico XOR e os Módulos. **Augusto Manzano**
- 28 **Introdução ao Cloud Computing e à Plataforma Windows Azure**
O que é o Cloud Computing? Para que serve? Em que ajuda? Descubra a resposta a esta e a outras questões relativas a esta renovação no mundo da computação que adveio da actual Virtualização em massa. **Nuno Godinho**
- 34 **Managed Extensibility Framework (MEF)**
Conheça a framework oficial de extensibilidade de aplicações para a Plataforma .NET que veio corrigir falhas na plataforma para construção de software extensível. **Ricardo Rodrigues**
- 37 **Microsoft BizTalk Server aos olhos dos programadores**
Muito se ouve falar sobre esta plataforma. Saiba agora quais são os benefícios da sua utilização no mercado de trabalho e em funcionalidades para o programador. **Sandro Pereira**
- 44 **O Editor de texto VIM**
Saiba os conceitos básicos deste editor de texto Open-Source para sistemas baseados em Unix, ferramenta inclusive capaz de realizar syntax-highlight de várias linguagens de programação. **Fábio Domingos**

COLUNAS

- 50 **CORE DUMP - Fazer mal = Rápido?**
Um artigo de opinião sobre o problema de não se criar as melhores soluções devido à falta de tempo no mundo da programação. Afinal, fazer mal é mais rápido que fazer bem? As consequências não se sobrepõem? **Fernando Martins**
- 51 **VISUAL (NOT) BASIC - Introdução ao OpenXML SDK**
Implemente este padrão ISO de arquivos (documentos, folhas de cálculo, entre outros) nas suas aplicações, em várias plataformas. Altere-os e visualize-os através de diversas aplicações. **Pedro Martins**

COMUNIDADES

- 58 **NetPonto - Certificações Microsoft**
Qual o percurso a percorrer para obter a sua Certificação Microsoft? Conheça as Certificações existentes, as hierarquias, os benefícios e como obtê-las. **Sara Silva**

EVENTOS

- 18 Jun. Reunião presencial Comunidade NetPonto - Lisboa
21 Jun. XII Encontro da Comunidade SQLPort
29 Jun. Windows Phone 7 “Mango” Dev Hub - Lisboa
09 Jul. Reunião presencial Comunidade NetPonto - Coimbra
22 Jul. IOI'2011 - 23ª Olimpíadas Internacionais de Informática
23 Jul. Reunião presencial Comunidade NetPonto - Lisboa

Para mais informações/eventos: http://bit.ly/PAP_Eventos

Redes da PT já estão preparadas para o IPv6

Um comunicado da PT refere que as redes que suportam o Meo, a TMN, e o Sapo ADSL já se encontram aptas a funcionar tanto em IPv4 (a norma que ainda domina nos endereços de Internet) como em IPv6. Para o processo de migração ficar concluído falta ainda proceder à adaptação para a versão 6 do IP dos terminais e equipamentos dos clientes da operadora. A PT definiu como objetivo a migração dos clientes empresariais para o IPv6 até ao final de 2011, mas não adianta qualquer data para adaptação dos equipamentos usados pelos clientes residenciais.

Uma vez que nem todos os intervenientes conseguem trabalhar à mesma velocidade, a rede da PT vai seguir a tendência mundial e passar a funcionar em dual stack que garante a compatibilidade com IPv4 e IPv6.

A migração para o IPv6 começou a ser trabalhada na PT há cerca de três anos, com o objetivo de acautelar o provável esgotamento dos 4,3 mil milhões endereços disponibilizados pela IPv4.

TMN testa rede 4G em Braga

A TMN iniciou no mês de Maio as primeiras demonstrações de redes da quarta geração de telemóveis (4G) em Braga.

A TMN optou por fazer a demonstração pública do 4G no Shopping Braga Parque. A demonstração, que deu a conhecer o potencial dos 150 Mbps do protocolo LTE (Long Term Evolution), abrange os serviços Meo Online, Meo Jogos, Videoconferências, e transmissões de vários vídeos em simultâneo. As demonstrações foram realizadas com tecnologias de rede fornecidas pela Nokia Siemens.

É a segunda vez que a TMN procede a demonstrações públicas das redes LTE: a primeira decorreu em Abril num centro comercial de Cascais.

Lançamento do Ubuntu 11.04

No fim do mês de Abril foi lançada a nova versão do Ubuntu, a Natty Narwhal (11.04), onde uma das suas principais novidades é a inclusão do Unity, que dá ao ambiente gráfico um aspecto renovado, correndo no entanto por cima do GNOME.

A versão 11.10 (Oneiric Ocelot) já está agendada para Outubro deste mesmo ano e a versão Alpha 1 que já foi lançada conta já com a implementação do GNOME 3 e com o Firefox 5 Beta. Conta também tal como a versão anterior, a 11.04, com o LibreOffice ao invés do OpenOffice.

Mais informação:

<https://wiki.ubuntu.com/OneiricOcelot/TechnicalOverview/Alpha1>

Microsoft apresenta Windows 8

Na conferência D9, na Califórnia, a Microsoft levantou o véu sobre a nova interface gráfica do Windows 8 que representa, talvez, a maior mudança na interface do sistema operativo da Microsoft. Ao contrário do que acontece agora, o ecrã principal apresenta agora uma série de "tiles", os tais retângulos com informação a ser atualizada em tempo real, tal como acontece com o Windows Phone 7.

A empresa de Redmond diz que o Windows 8 será compatível com todos os periféricos e dispositivos que neste momento correm no Windows 7, apesar de a interface ter sido redesenhada e repensada.

Ver o Vídeo: <http://bit.ly/iMPq0A>

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Git: Controlo de Versões para Pequenos e Grandes Projectos

O Git é um sistema de controlo de versões distribuído, gratuito e open-source, desenvolvido originalmente pelo Linus Torvalds, o criador do kernel do sistema operativo Linux, e actualmente é mantido pelo Junio Hamano juntamente com outros quase 300 colaboradores voluntários.



Comparado com outros sistemas de controlo de versões tradicionais, o Git diferencia-se por ser extremamente rápido, por simplificar o desenvolvimento de software de forma não-linear, onde podemos trabalhar em paralelo em diferentes funcionalidades das aplicações que desenvolvemos e então escolher quais funcionalidades devem fazer parte de cada versão da aplicação conforme o nosso fluxo de trabalho, e principalmente por ser um sistema distribuído bastante versátil e adequado para projectos de qualquer dimensão.

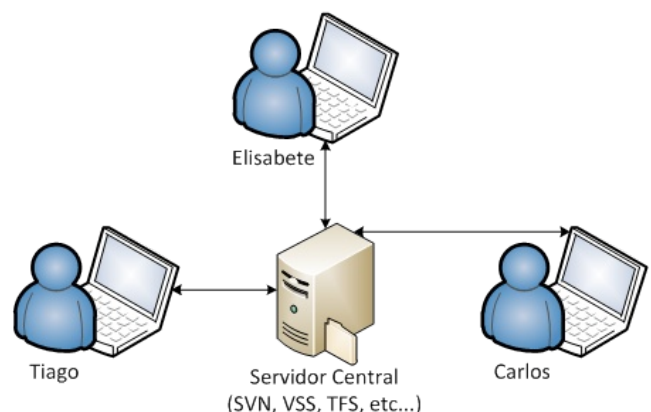
Outra grande vantagem do Git, é possuir versões para Windows, Linux e Mac OS X, o que facilita muito o controlo de versão de aplicações desenvolvidas em diferentes plataformas. Hoje em dia é muito comum uma mesma empresa desenvolver aplicações em .NET no Windows, outras em Java no Linux, e ainda outras para iPhone no Mac OSX, e poder utilizar uma única ferramenta para controlo de versão de todas as aplicações desenvolvidas na empresa, é excelente!

Neste artigo, vou explicar o funcionamento do Git e as principais diferenças entre os sistemas de controlo de versões centralizados e distribuídos, como instalar e configurar o Git no Windows, e os principais comandos que precisa conhecer para começar a utilizar o Git no dia-a-dia.

Centralizado vs Distribuído

O Subversion (SVN), o Visual Source Safe (VSS), o Team Foundation Server (TFS), e muitos outros, são todos sistemas de controlo de versões que funcionam de forma centralizada, isto é, existe sempre um computador central (servidor) que contém toda a história dos projectos com todas as inclusões e alterações que foram feitas pelos programadores desde o momento em que o projecto foi incluído no sistema de controlo de versões (geralmente no início do projecto) até a versão mais recente, e cada programador tem apenas *uma* versão da aplicação em seu próprio computador, normalmente a mais versão recente.

Dessa forma, os programadores estão sempre dependentes do servidor para guardar (*check-in*) alterações que desenvolvem em seus computadores, bem como para obter as alterações desenvolvidas por outros membros da equipa:



Assim, efectuar o controlo de versões com sistemas centralizados é geralmente mais lento, pois todas as operações (*check-in*, *check-out*, etc...) necessitam de comunicação com servidor central, que pode estar a apenas alguns metros de distância, como pode estar em outra cidade ou país. Além disso, torna-se impossível

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

controlar versões quando se está desconectado da rede onde encontra-se o servidor central, o que pode ser um problema quando é preciso trabalhar fora das instalações da empresa e não possibilidades de conectar-se remotamente à rede da empresa, por exemplo nas instalações de um cliente ou em viagens.

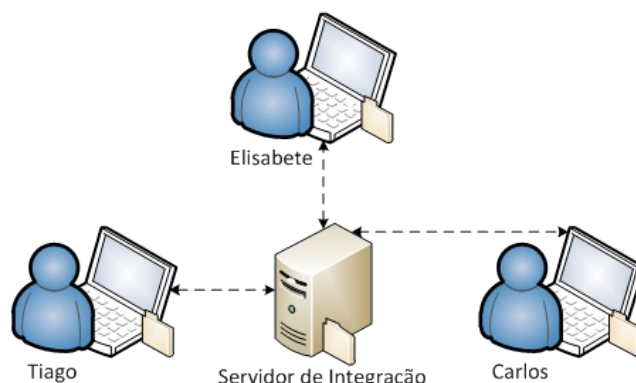
Controlo de Versão Distribuído

Num sistema de controlo de versão distribuído como o Git, não existe propriamente um “servidor central”. Cada programador tem uma cópia completa de toda a história dos projectos em sua própria máquina e pode controlar a versão de ficheiros e conteúdos de ficheiros localmente sem depender de um servidor e, somente quando achar apropriado, pode partilhar as suas alterações com outras pessoas da equipa e/ou receber alterações efectuadas por outros membros da equipa.

No Git, cada conjunto de alterações efectuadas em ficheiros ou conteúdos de ficheiros é chamado de “commit”, e conforme o programador implementa novas funcionalidades nas aplicações que está a desenvolver, vai efectuando diferentes commits que ficam guardados no seu próprio computador sem precisar depender de um servidor.

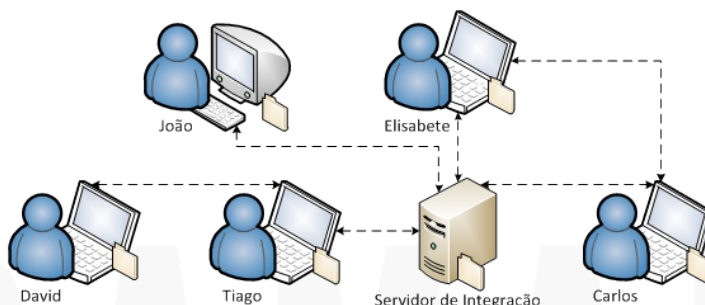
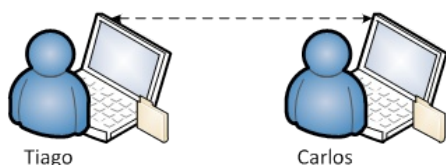
Uma vez que cada membro da equipa possui uma cópia integral de toda a história dos projectos, um programador que queira partilhar os seus commits recentes com outro colaborador, pode enviar directamente os commits para este colaborador (*push*), ou ainda, pode permitir que o outro colaborador obtenha os commits a partir de seu próprio computador (*pull*), tudo sem precisar comunicar-se com um servidor central.

No entanto, ao trabalhar em equipas com várias pessoas, ao invés de cada pessoa comunicar-se individualmente com outra para enviar/receber *commits*, é normal existir um sítio comum onde os elementos da equipa enviam os seus *commits* para que sejam partilhados com o restante da equipa, sendo este sítio comum normalmente conhecido como “servidor de integração”.



À primeira vista, este sítio comum se parece muito como o “servidor central” dos sistemas de controlo de versões centralizados, mas em realidade é bem diferente, pois é apenas mais um computador com uma cópia de toda a história dos projectos, assim como é o computador de qualquer outro membro da equipa. O servidor de integração é apenas uma convenção social entre os participantes da equipa, de forma a facilitar a partilha das alterações entre as várias pessoas.

O importante a destacar é que nenhum membro da equipa está directamente dependente ou conectado ao “servidor de integração”. Todos trabalham de forma desconectada em seus computadores, onde efectuam diversos commits que são gravados localmente, e só utilizam o “servidor de integração” para partilhar seus commits recentes com o restante da equipa, quando/se quiserem.



TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Dessa forma, as funcionalidades que devem ser partilhadas com toda a equipa, podem ser enviadas (*push*) para o “servidor de integração” e os membros da equipa podem obter (*pull*) essas alterações quando desejarem, assim como funcionalidades que não ainda não estão prontas para serem partilhadas com toda a equipa, podem ser partilhadas apenas entre as pessoas envolvidas no desenvolvimento dessas funcionalidades.

Importante: Algumas ferramentas de controlo de versões centralizado, como por exemplo o Team Foundation Server (TFS), permitem accionar uma opção para trabalhar de forma desconectada do servidor, para então mais tarde (quando for possível conectar-se ao servidor central) enviar as alterações. Isto não é a mesma coisa que trabalhar num sistema distribuído, pois neste caso não será possível efectuar commits no próprio computador, enquanto efectua diferentes alterações no código, ou seja, a ferramenta não permite gravar diferentes conjuntos de alterações (commits), para depois enviá-los para o servidor. Se trabalhou em funcionalidades diferentes, não conseguirá (facilmente) distinguir quais ficheiros/conteúdos correspondem a cada funcionalidade que trabalhou enquanto estava desconectado, e provavelmente irá enviar um único commit para o servidor central com todas as alterações, e estará a usar o sistema de controlo de versões de forma pouco eficiente.

Repositórios Git

Para controlar versões dos conteúdos, o Git utiliza o conceito de repositórios, onde cada repositório corresponde a uma pasta que pode conter ficheiros e sub-pastas também com ficheiros onde controla a versão de todos os conteúdos desta pasta e sub-pastas que existirem.

Uma pasta só considerada um repositório Git, após executarmos um comando específico (*init*) para que o Git inicialize a pasta como um repositório, e normalmente criamos um repositório Git para cada projecto que desenvolvemos, de forma a controlar as versões dos conteúdos de cada projecto separadamente.

Por exemplo, observe a estrutura de pastas abaixo:

```
C:\Projectos\  
----- NetPonto  
----- lib  
----- src  
----- NetPonto.sln  
----- ...  
----- doc  
----- Portugal-a-Programar  
----- lib  
----- src  
----- Portugal-a-Programar.sln  
----- ...  
----- doc
```

Qualquer uma destas pastas poderia ser transformada em um repositório Git de acordo com as necessidades do programador, mas normalmente controlamos as versões individuais de cada projecto, portanto faz sentido que a pasta `C:\Projectos\NetPonto\` seja um repositório Git e pasta `C:\Projectos\Portugal-a-Programar\`, ser outro repositório Git separado.

Instalação do Git no Windows

O site oficial do Git é o <http://git-scm.com>, onde pode efectuar o download da versão mais recente para a plataforma que desejar.

Para utilizar o Git no Windows, existem basicamente duas formas: Directamente, utilizando o [msysGit](#) ou através do [Cygwin](#).

O **msysGit** é uma versão desenvolvida especialmente para funcionar no Windows, e é a opção mais utilizada actualmente, enquanto o **Cygwin** é uma colecção de ferramentas e APIs que permite criar um ambiente com a mesma aparência e experiência do Linux, dentro do Windows, e então utilizar o Git através do **Cygwin**.

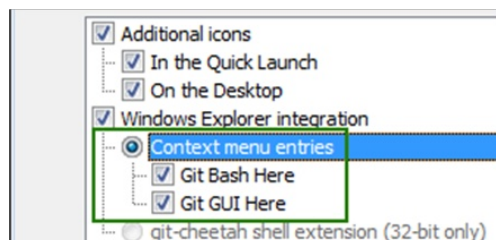
Neste artigo, vou demonstrar a instalação e configuração do **msysGit**, que no momento em que escrevo este texto está na versão **1.7.5.4**.

TEMA DE CAPA

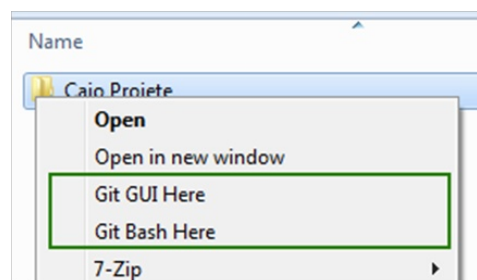
Git: Controlo de Versões para Pequenos e Grandes Projectos

Após efectuar o download **msysGit** e executar o instalador, recomendo efectuar a instalação mantendo as opções que já vem seleccionadas por padrão (Next, Next, Next ...), com excepção do passo 4, onde recomendo seleccionar também as opções “Context menu entries”, “Git Bash Here” e “GUI Bash Here” para adicionar duas entradas nos menus de contexto do Windows Explorer que permitem seleccionar uma pasta e executar as ferramentas do Git directamente na pasta seleccionada.

Passo 4 da instalação do Git

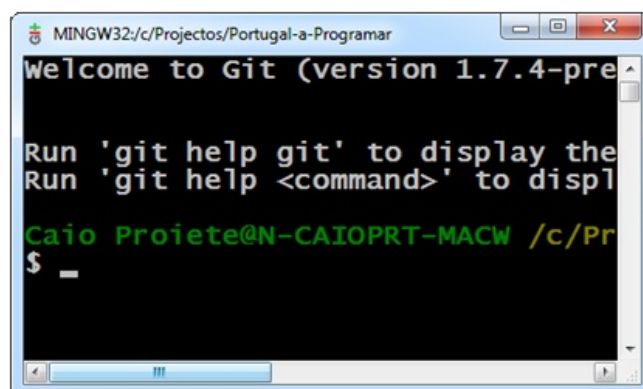
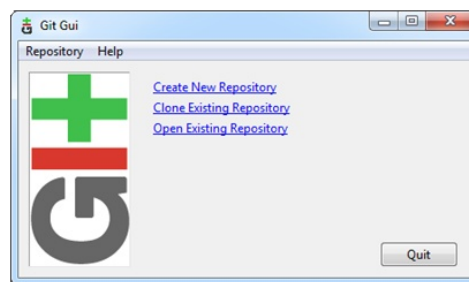


No Windows Explorer (após a instalação)



Principais Ferramentas do Git / msysGit

As principais ferramentas do Git são o **Git GUI** e o **Git Bash**. O **Git GUI**, como o nome indica, é uma interface gráfica que permite criar e gerir repositórios Git através de uma interface gráfica simples, e o **Git Bash** é uma aplicação de linha-de-comando, que também permite criar e gerir repositórios Git via linha-de-comando.

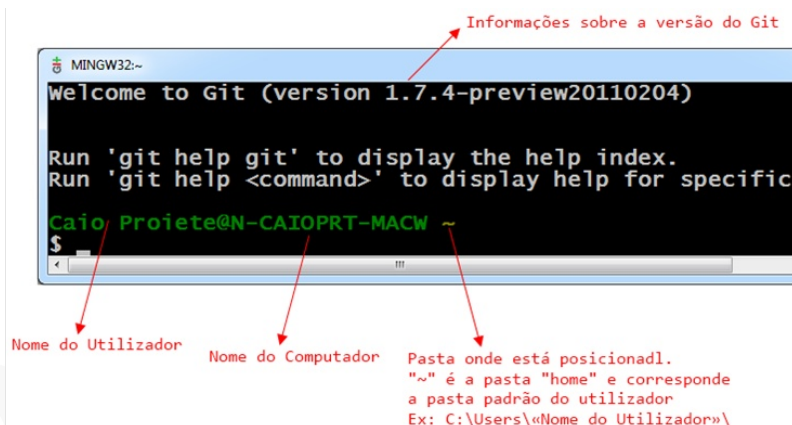


Enquanto o Git GUI permite realizar as principais tarefas do dia-a-dia, o Git Bash é a ferramenta que permite total controlo dos repositórios, e é também a ferramenta que utilizo para mostrar o funcionamento do Git neste artigo.

Configuração Inicial do Git

Após a instalação do msysGit, o próximo passo é configurar o nome e o e-mail que será utilizado por padrão, na criação dos repositórios e dos commits. É possível configurar estes parâmetros de forma global ou individualmente para cada repositório, e para isto utilizamos o comando “git config” em uma sessão do Git Bash, a ferramenta de mencionada acima e que após a instalação pode ser encontrada na pasta “Git” no menu iniciar do Windows.

Ao iniciar o Git Bash, irá visualizar um ecrã parecido com a figura abaixo:

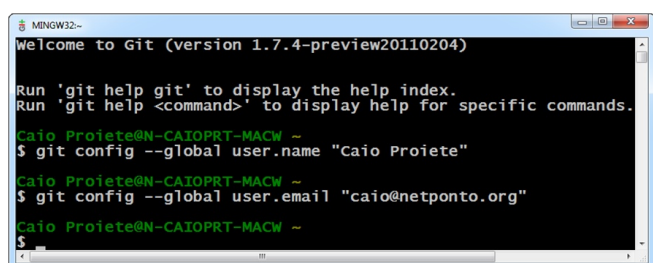


TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Os parâmetros globais a serem definidos são o `user.name` e o `user.email`, que correspondem ao nome e ao e-mail do utilizador, respectivamente, e para efectuar a configuração, basta executar os comandos abaixo, substituindo os valores entre aspas pelo nome e e-mail do utilizador:

```
$ git config --global user.name "Seu Nome"
$ git config --global user.email "seu@email.pt"
```



```
MINGW32~
Welcome to Git (version 1.7.4-preview20110204)
Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Caio Proiete@N-CAIOPRT-MACW ~
$ git config --global user.name "Caio Proiete"

Caio Proiete@N-CAIOPRT-MACW ~
$ git config --global user.email "caio@netponto.org"

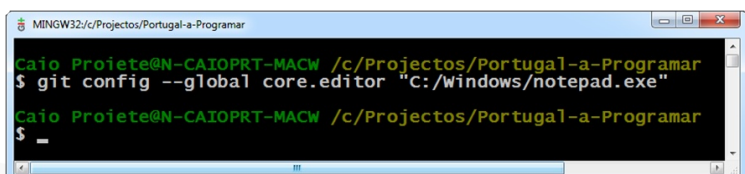
Caio Proiete@N-CAIOPRT-MACW ~
$
```

Além de configurar o nome e o e-mail do utilizador, é comum neste momento configurar também um parâmetro global para definir o editor de textos que deve ser utilizado pelo Git.

Por padrão, o Git utiliza o editor de textos [Vim](#), uma versão melhorada do editor [Vi](#) muito utilizado em ambientes Unix e que exige que o utilizador conheça as suas diferentes combinações de teclas para aceder cada uma das suas funcionalidades, o que pode tornar a utilização do Git mais complicada para quem não está acostumado com este editor, portanto uma opção é definir o bloco de notas do Windows (Notepad) como editor de texto padrão, ou outro de sua preferência (Notepad++, Textpad, etc...).

Para configurar o editor de textos, basta executar o comando abaixo, adaptando o caminho do executável do editor de textos desejado:

```
$ git config --global core.editor
"C:/Windows/notepad.exe"
```



```
MINGW32/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$ git config --global core.editor "C:/windows/notepad.exe"

Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$
```

Controlo de Versões com o Git

Criação de um Repositório Git

Depois de configurar o nome e o e-mail padrão, o próximo passo escolher a pasta onde irá criar um repositório Git, ou criá-la caso ainda não exista.

Para este exemplo, o repositório Git estará localizado na pasta `C:\Projectos\Portugal-a-Programar\`, e assumo que esta pasta ainda não exista, portanto podem ser criadas através do Windows Explorer se preferir, ou directamente através do Git Bash, utilizando alguns comandos existentes para o efeito:

- Criar a pasta `C:\Projectos`

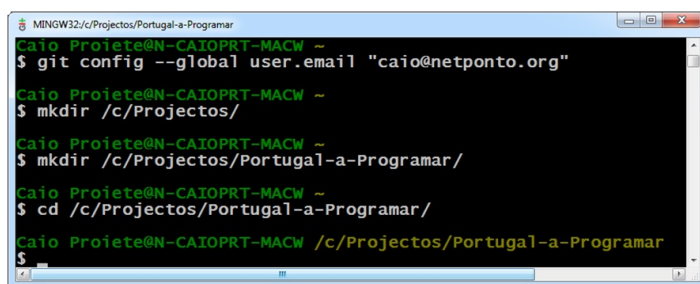
```
$ mkdir /c/Projectos/
```

- Criar a pasta `C:\Projectos\Portugal-a-Programar`

```
$ mkdir /c/Projectos/Portugal-a-Programar/
```

- Entrar na pasta `C:\Projectos\Portugal-a-Programar`

```
$ cd /c/Projectos/Portugal-a-Programar/
```



```
MINGW32/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW ~
$ git config --global user.email "caio@netponto.org"

Caio Proiete@N-CAIOPRT-MACW ~
$ mkdir /c/Projectos/

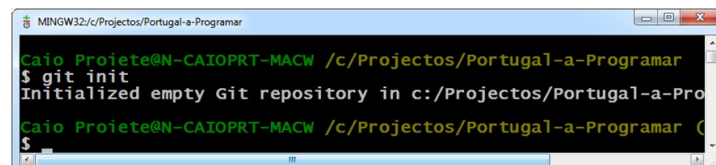
Caio Proiete@N-CAIOPRT-MACW ~
$ mkdir /c/Projectos/Portugal-a-Programar/

Caio Proiete@N-CAIOPRT-MACW ~
$ cd /c/Projectos/Portugal-a-Programar/

Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$
```

Uma vez posicionado na pasta desejada, para criar o repositório do Git deve utilizar o comando "git init", que essencialmente irá "preparar" a pasta actual para permitir o controlo de versões dos ficheiros e sub-pastas dentro deste repositório recém-criado.

```
$ git init
```



```
MINGW32/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$ git init
Initialized empty Git repository in c:/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$
```

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

O que este comando faz, em realidade é criar a estrutura do repositório Git em uma pasta oculta chamada “.git” dentro da pasta onde executou o comando. Neste exemplo, em “C:\Projectos\Portugal-a-Programar\.git”. É dentro desta pasta estarão, entre outras coisas, todos os commits.

A pasta onde criou o repositório Git, a partir de agora passa a ser chamada de “*working folder*” (pasta de trabalho).

Com o repositório Git criado, pode adicionar novos ficheiros, alterar ficheiros que já existiam, remover ficheiros, criar novas pastas, e etc... Para este exemplo, pode adicionar três ficheiros na pasta onde está o repositório Git com o seguinte nome/conteúdo:

Primeiro.txt

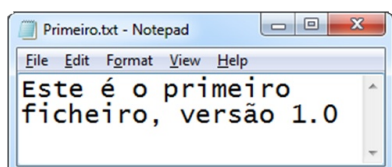
Este é o primeiro ficheiro, versão 1.0

Segundo.txt

Este é o segundo ficheiro, versão 1.0

Terceiro.txt

Este é o terceiro ficheiro, versão 1.0



Neste momento, pode executar um outro comando do Git, o “git status”, que permite verificar o estado do repositório, e perceber quais são os ficheiros que não estão a ser controlados (novos ficheiros), os ficheiros que foram modificados e os que foram apagados. Este é normalmente o comando que irá utilizar com mais frequência, para consultar o estado do repositório conforme efectua alterações e antes de efectuar commits.

Neste exemplo, o comando “git status” deve mostrar que existem três ficheiros na pasta, mas que não estão a ser controlados pelo Git, por enquanto:

```
$ git status
```

```
MINGW32/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar (
$ git status
# On branch master
#
# Initial commit
#
Untracked files:
#
#   (use "git add <file>..." to include in what will be committ
#
#   Primeiro.txt
#   Segundo.txt
#   Terceiro.txt
#
nothing added to commit but untracked files present (use "git a
$
```

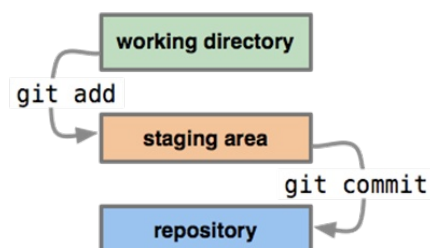
Staging Area e Commits

Conforme efectua alterações no repositório, deverá guardar as versões (efectuar *commits*) dos conteúdos que desejar, quando for apropriado. Idealmente cada commit representa uma unidade lógica, como por exemplo uma nova funcionalidade implementada numa aplicação, ou uma correcção de um bug, ou seja, um *commit* é um conjunto de alterações que foram feitas no repositório, e podem ser alterações em ficheiros existentes, criação de novos ficheiros, ou remoção de um ou mais ficheiros.

No entanto, ao trabalhar em uma tarefa específica (por exemplo, na implementação de uma funcionalidade X), é normal um programador efectuar outras alterações no código que não estão relacionadas com a tarefa em questão, mas que já sabe que será preciso na implementação de uma funcionalidade Y (outra tarefa futura), e a grande vantagem do Git é compreender que isso é normal acontecer, e permitir definir quais são as alterações que pertencem a uma funcionalidade X (commit X), e quais pertencem a uma funcionalidade Y (commit Y), e é por isso que o Git conta com um recurso chamado “staging area”.

Assim, para criar um *commit*, é necessário primeiro indicar quais são as alterações que devem fazer parte do

commit. Estas alterações são adicionadas nessa área chamada “*staging area*”, de forma que apenas as alterações que correspondem a uma determinada unidade lógica sejam armazenadas como um commit.



TEMA DE CAPA

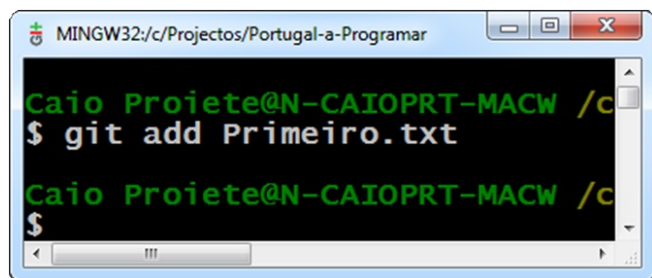
Git: Controlo de Versões para Pequenos e Grandes Projectos

O comando utilizado para adicionar ficheiros (ou parte de um ficheiro) à staging area é o comando “git add”, e depois de adicionar todas as alterações na staging area, deve executar o comando “git commit” para transformar todas as alterações adicionadas na staging area em um commit.

O comando “git add” pode receber diferentes parâmetros, e permite adicionar ficheiros individualmente, todos os ficheiros alterados ou novos que encontrar, conteúdos de ficheiros, entre outras opções.

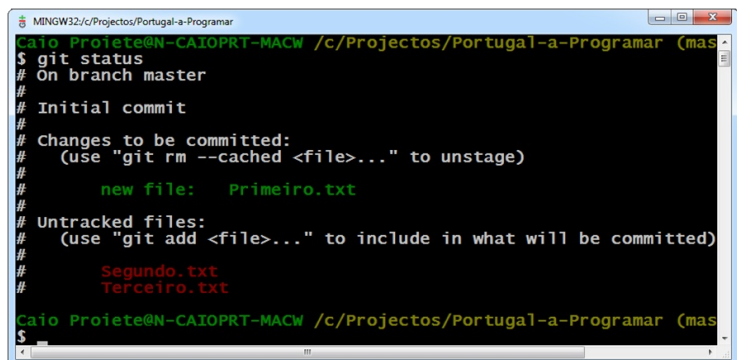
Para adicionar o ficheiro “Primeiro.txt” na staging area, pode informar o nome do ficheiro individualmente no comando “git add”:

```
$ git add Primeiro.txt
```



```
MINGW32:/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$ git add Primeiro.txt
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$
```

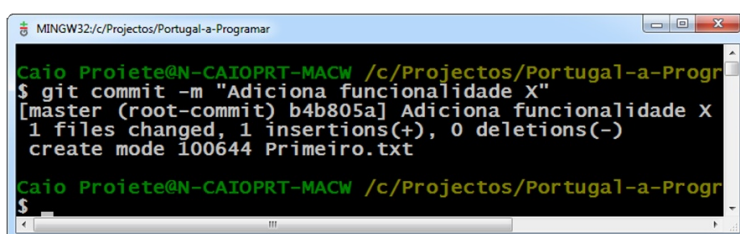
Neste momento, o ficheiro “Primeiro.txt” foi adicionado a staging area, o que significa que fará parte do próximo commit, enquanto os outros dois ficheiros continuam sem estarem a ser controlados pelo Git. Se executar o comando “git status” novamente pode confirmar que o ficheiro foi efectivamente adicionado a staging area:



```
MINGW32:/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar (master)
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   Primeiro.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       Segundo.txt
#       Terceiro.txt
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar (master)
$
```

Para efectuar o commit, deve utilizar o comando “git commit” e pode opcionalmente utilizar o parâmetro “-m” e informar a descrição do commit. Se não utilizar o parâmetro “-m” o Git irá executar o editor de texto padrão configurado para que possa informar a mensagem a descrever as alterações contidas no commit.

```
$ git commit -m "Adiciona funcionalidade X"
```



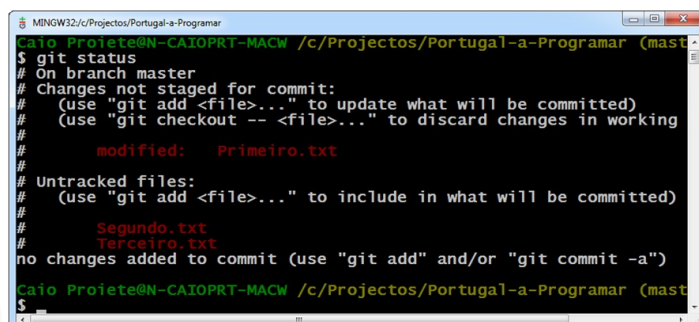
```
MINGW32:/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$ git commit -m "Adiciona funcionalidade X"
[master (root-commit) b4b805a] Adiciona funcionalidade X
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 Primeiro.txt
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar
$
```

Se após efectuar este primeiro commit consultar novamente o status do repositório, verá que apenas os ficheiros “Segundo.txt” e “Terceiro.txt” aparecem como ficheiros que não estão a ser controlados, e o ficheiro “Primeiro.txt” deixa de aparecer na lista.

Para este exemplo, o ficheiro “Primeiro.txt” deve ser alterado para demonstrar uma actualização de conteúdo no repositório:

```
Primeiro.txt
Este é o primeiro ficheiro, versão 1.1
(alterado)
```

Ao consultar o status do repositório, verá que o ficheiro “Primeiro.txt” está marcado como “modificado”, ou seja, o seu conteúdo está diferente do conteúdo que está no último commit efectuado, e os outros dois ficheiros continuam como estavam antes.



```
MINGW32:/c/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar (master)
$ git status
# On branch master
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   Primeiro.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       Segundo.txt
#       Terceiro.txt
no changes added to commit (use "git add" and/or "git commit -a")
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-a-Programar (master)
$
```

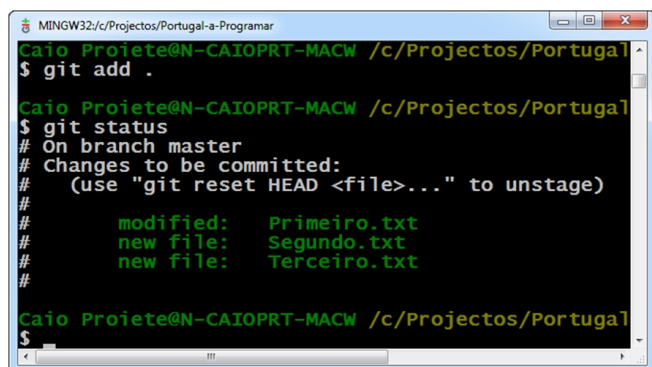

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Para um próximo commit, e supondo que a alteração no ficheiro “Primeiro.txt” juntamente com os ficheiros “Segundo.txt” e “Terceiro.txt” fazem parte de uma mesma funcionalidade Y, pode adicionar cada um dos ficheiros manualmente a staging area, ou pode utilizar um atalho e adicionar todos os ficheiros, novos e modificados, de uma só vez:

```
$ git add .
```

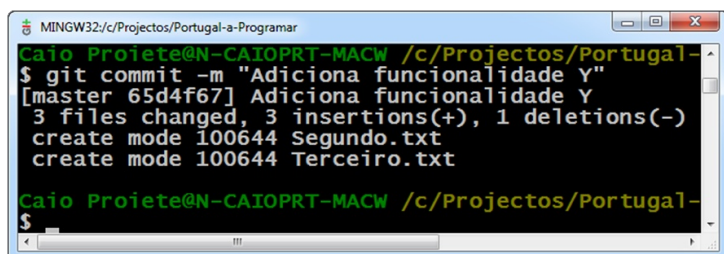
Ao consultar o estado do repositório, verá que todos os ficheiros estão seleccionados para fazerem parte do próximo commit:



```
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal
$ git add .
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   Primeiro.txt
#       new file:   Segundo.txt
#       new file:   Terceiro.txt
#
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal
$
```

Para concluir e efectuar o commit, basta utilizar o comando “git commit” como feito anteriormente:

```
$ git commit -m "Adiciona funcionalidade Y"
```



```
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-
$ git commit -m "Adiciona funcionalidade Y"
[master 65d4f67] Adiciona funcionalidade Y
3 files changed, 3 insertions(+), 1 deletions(-)
create mode 100644 Segundo.txt
create mode 100644 Terceiro.txt
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-
$
```

E é desta forma que controlamos as versões no repositório. Após efectuar um conjunto de alterações, deve seleccionar quais alterações devem fazer parte de commit através do comando “git add”, e então finalizar a operação efectuando o commit, através do comando “git commit”.

Nota: Nas imagens acima pode reparar que os ficheiros

modificados e/ou adicionados à staging area são mostrados na cor **verde** ao executar o comando “git status”, assim como ficheiros que não estão a ser controlados aparecem na cor **vermelho**. Para habilitar a utilização de cores, deve configurar o parâmetro global “color.ui” com o valor “true”:

```
$ git config --global color.ui true
```

Ignorar ficheiros e pastas com o .gitignore

Por padrão, o Git permite controlar as versões de qualquer tipo de ficheiro que existir dentro da pasta (ou sub-pastas) onde está o repositório, mas nem sempre queremos controlar as versões de todos os tipos de ficheiro.

Ficheiros temporários que são criados pelo sistema operativo ou pela ferramenta de desenvolvimento, não devem fazer parte do repositório, ou seja, nunca devem adicionados aos commits. No entanto, poderão existir na working folder enquanto trabalha, mas devem ser ignorados pelo Git quando adicionamos alterações a staging area e efectuamos commits.

Por exemplo, o Windows costuma criar ficheiros com o nome “Thumbs.db” em pastas que possuem imagens, onde armazena uma espécie de cache das miniaturas das imagens contidas na pasta. Esses ficheiros “Thumbs.db” jamais devem ser guardados no repositório, pois podem ser recriados pelo Windows conforme o conteúdo da pasta muda.

Ainda, num projecto .NET, o Visual Studio cria ficheiros do tipo “NomeProjecto.suo”, “NomeProjecto.user”, “NomeProjecto.sln.cache”, e outros tipos de ficheiros temporários que estão relacionados com o utilizador actual e podem ser recriados pelo Visual Studio a qualquer momento. Estes ficheiros também não devem ser armazenados no repositório Git, e portanto devem ser ignorados.

Além disso, é uma prática comum não guardar ficheiros executáveis, e outros ficheiros que podem ser gerados (compilados) a partir do código que está a ser controlado

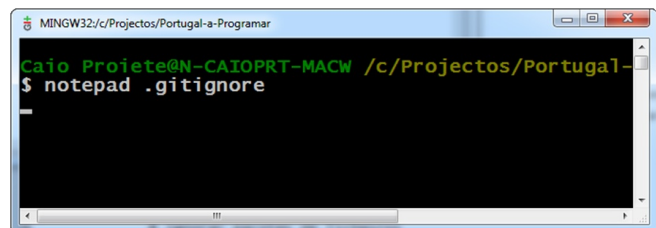
TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

pelo repositório Git, ou seja, deve-se guardar apenas o código-fonte e as dependências necessárias para que seja possível compilar o código-fonte. Por exemplo, num projecto .NET, normalmente temos as pastas “bin” e “obj” que possuem o resultado da compilação de cada projecto do Visual Studio. Estas pastas também não devem ser armazenadas no repositório Git, pois deve ser possível recriá-las a qualquer momento, a partir do código-fonte do projecto.

Para poder ignorar determinados tipos de ficheiros ou pastas, criamos um ficheiro chamado “.gitignore” na pasta onde criamos o repositório. O Windows por padrão não permite criar ficheiros que não tenham nome e apenas uma extensão, como é o caso do “.gitignore”, por isso uma alternativa simples é criar o ficheiro a partir do Git Bash, executando o Bloco de Notas para criar o ficheiro:

```
$ notepad .gitignore
```



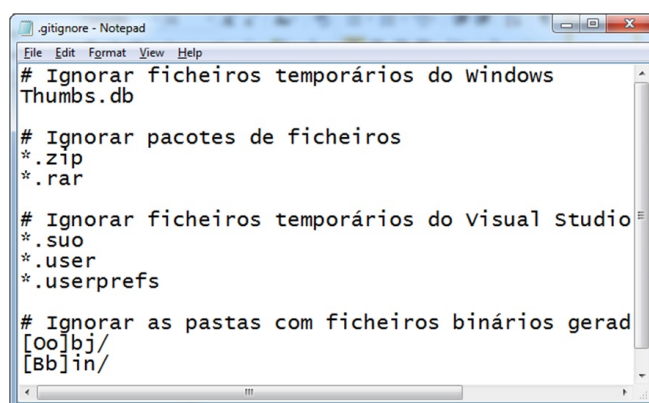
No conteúdo do ficheiro .gitignore, pode inserir comentários (linhas que começam com “#”), definir os tipos de ficheiros que pretende ignorar e as pastas que pretende ignorar por completo, independente do conteúdo. Neste exemplo:

```
# Ignorar ficheiros temporários do Windows
Thumbs.db

# Ignorar pacotes de ficheiros
*.zip
*.rar

# Ignorar ficheiros temporários do Visual Studio
*.suo
*.user
*.userprefs
```

```
# Ignorar as pastas com ficheiros binários
gerados via compilação
[Oo]bj/
[Bb]in/
```



A sintaxe é bastante intuitiva, e pode utilizar “*.extensão” dos ficheiros que pretende ignorar, e informar o nome das pastas com uma barra “/” no final para indicar que trata-se de uma pasta. O Git faz diferença entre letras maiúsculas e minúsculas, por isso pode utilizar expressões como “[Oo]bj/” que permite indicar que tanto as pastas “obj” quanto “Obj/” devem ser ignoradas.

Ao criar o ficheiro .gitignore, os ficheiros e pastas definidos no conteúdo deste ficheiro serão ignorados pelo Git, e deixarão de aparecer, por exemplo, quando visualizar o estado do repositório com o comando “git status”, no entanto é importante adicionar o ficheiro .gitignore ao repositório para garantir que continuará a ignorar os ficheiros desejados nos próximos commits, e também para que todos os membros da equipa estejam a ignorar os mesmos tipos de ficheiros e pastas.

```
$ git add .gitignore
$ git commit -m "Adiciona o ficheiro .gitignore
ao repositorio"
```

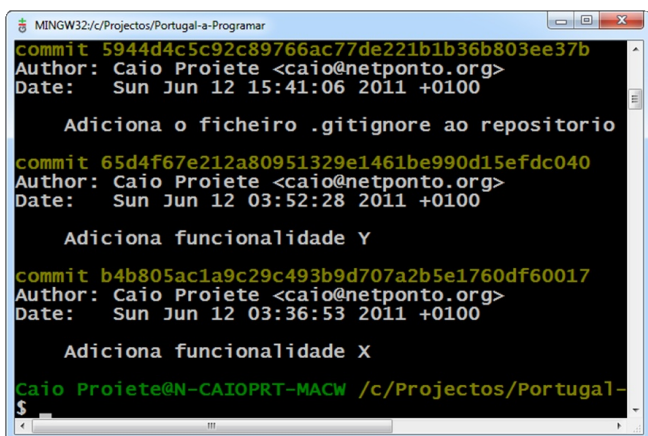
TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Consulta do Histórico de Commits

Conforme efectuamos commits, é muito comum necessitarmos consultar a história do projecto para, por exemplo, perceber quando foi introduzida uma determinada alteração. Para este efeito, o Git possui um comando chamado “git log”, que permite visualizar os commits que estão armazenados no repositório, incluindo informações como a data e hora, nome e e-mail do utilizador que efectuou cada *commit*, e também o identificador único de cada commit (ex: 5944d4c5c92c89766ac77de221b1b36b803ee37b), que podemos utilizar quando necessitamos efectuar operações específicas em determinados commits.

```
$ git log
```



```
Commit 5944d4c5c92c89766ac77de221b1b36b803ee37b
Author: Caio Proiete <caio@netponto.org>
Date: Sun Jun 12 15:41:06 2011 +0100
Adiciona o ficheiro .gitignore ao repositorio

Commit 65d4f67e212a80951329e1461be990d15efdc040
Author: Caio Proiete <caio@netponto.org>
Date: Sun Jun 12 03:52:28 2011 +0100
Adiciona funcionalidade Y

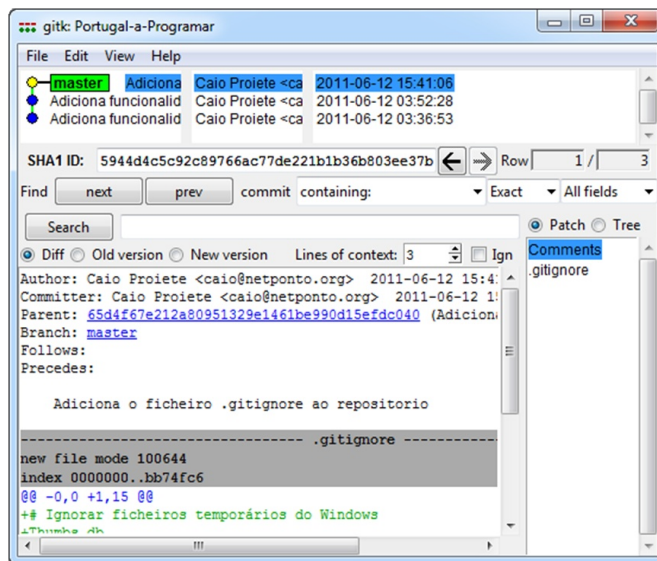
Commit b4b805a1a9c29c493b9d707a2b5e1760df60017
Author: Caio Proiete <caio@netponto.org>
Date: Sun Jun 12 03:36:53 2011 +0100
Adiciona funcionalidade X

Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Portugal-
```

O comando “git log” pode receber diferentes parâmetros que permitem visualizar mais, ou menos informação sobre os commits, algumas opções de formatação, entre outros recursos.

Uma outra forma de consultar o histórico de *commits* é utilizar um utilitário instalado juntamente com o msysGit chamado “gitk”, que permite consultar os commits de forma gráfica.

```
$ gitk
```



Desenvolvimento em Paralelo

Ao trabalharmos no desenvolvimento de software profissional, é muito comum termos um ambiente de desenvolvimento onde efectuamos testes de novas funcionalidades que estão a ser desenvolvidas, separado do ambiente de produção onde a aplicação está a ser executada pelos utilizadores finais. Em realidade, é também muito comum termos um ambiente intermédio de “controlo de qualidade” (também conhecido como ambiente “de qualificação”, “de qa” ou “de testes”), que geralmente possui as mesmas características do ambiente de produção, e possui uma versão da aplicação com funcionalidades que ainda precisam ser testadas antes de serem promovidas para o ambiente de produção.



Desta forma, o ambiente de produção possui sempre a versão mais antiga do projecto, mas também a mais estável e que passou pelos testes de controlo de qualidade, enquanto o ambiente de qualificação (se houver) possui uma versão mais nova do projecto, mas que ainda necessita ser testado antes de evoluir para o ambiente de produção, e por fim o ambiente de desenvolvimento possui

TEMA DE CAPA

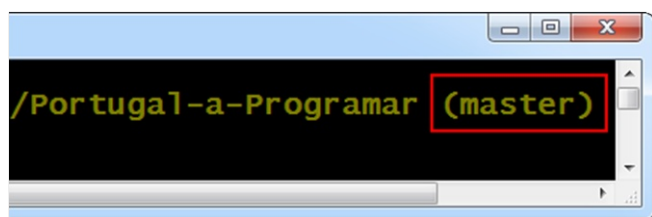
Git: Controlo de Versões para Pequenos e Grandes Projectos

uma versão ainda mais nova do projecto, com as funcionalidades que estão a ser desenvolvidas e que após testes dos developers poderão ser enviadas para o ambiente de qualificação, para serem efectuados mais testes.

Uma vez que temos ambientes separados, podemos utilizar a ferramenta de controlo de versões para manter diferentes as versões dos nossos projectos em paralelo de forma a conseguirmos enviar uma nova versão para qualquer um dos ambientes o mais rápido possível e idealmente a qualquer momento.

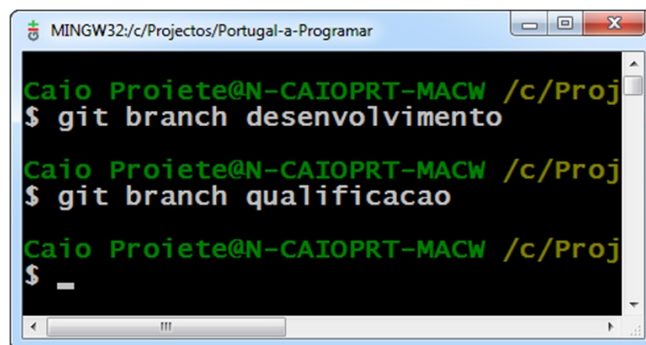
Para este efeito, o Git e a grande maioria de sistemas de controlo de versões oferece um recurso chamado “branch”, que no Git é representado por um conjunto de commits é identificado por um nome escolhido pelo developer que efectua criação do “branch”.

Ao criar um novo repositório Git, automaticamente é criada um primeiro branch chamado “master”, que irá agregar todos os *commits* que fizer neste *branch*. Pode identificar a qualquer momento em qual *branch* encontra-se posicionado através do nome entre parênteses após o caminho da pasta:



Para criar um novo *branch*, pode utilizar o comando “git branch”, e informar o nome do *branch* a ser criado:

```
$ git branch desenvolvimento
$ git branch qualificacao
```

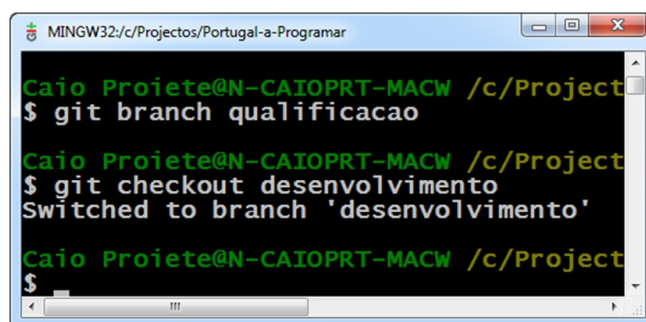


Estes comandos efectuam a criação de dois novos *branches* chamados “desenvolvimento” e “qualificacao” respectivamente, mas que estão a apontar para o mesmo commit do *branch* actual (*master*), que neste exemplo, é o commit que adiciona o ficheiro `.gitignore`.

O *branch* “*master*”, neste exemplo, está a ser usado como sendo o *branch* com a versão de produção, enquanto os outros *branches* representam as versões dos outros ambientes. Isto pode variar de acordo com a preferência pessoal da equipa. Em alguns casos, a *branch* “*master*” será utilizada como *branch* de desenvolvimento, por exemplo, e são criadas outras *branches* para os outros ambientes.

Para mudar para um novo *branch*, deve utilizar o comando “git checkout” e informar o nome da *branch* para onde deseja ir:

```
$ git checkout desenvolvimento
```



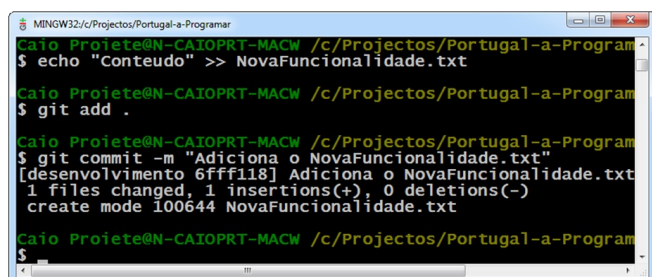
A partir de agora todos os commits efectuados serão armazenados na *branch* “desenvolvimento”, de forma isolada da *branch* “*master*” criada inicialmente, e também de forma isolada da *branch* “qualificacao”.

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Para este exemplo, pode criar um ficheiro chamado "NovaFuncionalidade.txt" e efectuar o commit:

```
$ echo "Conteudo" >> NovaFuncionalidade.txt
$ git add .
$ git commit -m "Adiciona o
NovaFuncionalidade.txt"
```

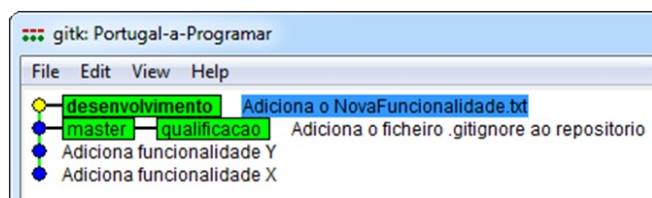


```
MINGW32/c:/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$ echo "Conteudo" >> NovaFuncionalidade.txt
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$ git add .
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$ git commit -m "Adiciona o NovaFuncionalidade.txt"
[desenvolvimento 6fff118] Adiciona o NovaFuncionalidade.txt
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 NovaFuncionalidade.txt
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$
```

E neste momento, o repositório possui quatro commits diferentes:

- Commit 4 | Adiciona o NovaFuncionalidade.txt
- Commit 3 | Adiciona o ficheiro .gitignore ao repositório
- Commit 2 | Adiciona funcionalidade Y
- Commit 1 | Adiciona funcionalidade X

No entanto, o commit 4 está presente apenas no branch "desenvolvimento", enquanto a branch "master" e "qualificacao" continuam a apontar para o commit 3, como pode visualizar através do utilitário gitk:



Desta forma, é possível continuar a efectuar commits neste branch sem comprometer as versões que estão em paralelo e correspondem aos outros ambientes, e apenas quando for apropriado, poderá juntar os commits efectuados em uma branch, com outra.

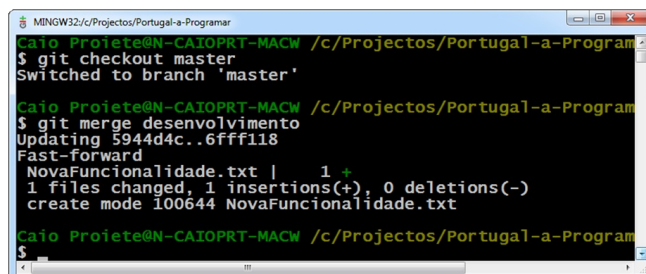
A utilização de *branches* no Git é tão simples e tão rápida, que muitos developers adoptam uma convenção conhecida como "branch-per-feature", onde criam novas branches

para cada nova funcionalidade que pretendem implementar num projecto, e depois decidem funcionalidade/branch deve ser adicionada nas branches principais de cada ambiente.

A operação de juntar os commits de uma branch com os commits de outra *branch* é chamada de "merge" e para este efeito o Git possui o comando "git merge" que permite juntar a *branch* informada como parâmetro, na branch onde está posicionado.

Por exemplo, para juntar as alterações da branch "desenvolvimento" com a branch "master", em primeiro lugar é preciso ir para a *branch* "master" (com o comando "git checkout") e então executar o comando "git merge" e informar que deve ser efectuado o merge da branch "desenvolvimento" com a branch actual:

```
$ git checkout master
$ git merge desenvolvimento
```



```
MINGW32/c:/Projectos/Portugal-a-Programar
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$ git checkout master
Switched to branch 'master'
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$ git merge desenvolvimento
Updating 5944d4c..6fff118
Fast-forward
 NovaFuncionalidade.txt | 1 +
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 NovaFuncionalidade.txt
Caio Proiete@N-CAIOPRT-MACW /c:/Projectos/Portugal-a-Programar
$
```

Como pode reparar na imagem acima, o Git efectuou um "Fast-forward" que é o tipo de merge mais simples que existe, onde apenas o apontador da branch actual move-se para apontar para o novo *commit*, que neste caso é mais novo.

Existem outros tipos de *merge*, e em alguns casos um merge pode causar conflitos, por exemplo, caso as mesmas linhas de um ficheiro tenha sido alteradas por commits diferentes, e pode necessitar de intervenção manual do developer, e que normalmente utiliza uma ferramenta para auxiliar a resolução de conflitos. Este é um assunto que merece um artigo próprio, que ficará para uma próxima edição desta revista.

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

Partilha de Alterações em Equipa

A partilha de *commits* entre os membros da equipa pode ser feita directamente entre os repositórios dos developers envolvidos, pode ser utilizado um repositório partilhado, ou outras formas de acordo com o fluxo de trabalho da equipa. Na próxima secção irá encontrar breve explicação sobre os workflows mais comuns para controlo de versões em sistemas distribuídos.

Os principais comandos para a utilização do Git em equipa são o “git clone”, “git pull” e o “git push”.

O comando “git clone” serve para criar uma cópia integral de um repositório Git. Este é o comando utilizado quando desejamos participar de um projecto, e para isso precisamos ter uma cópia do repositório em nosso computador. O “git clone” automaticamente guarda uma referência para o repositório original, de forma a facilitar obter actualizações desse repositório, bem como enviar as actualizações feitas localmente.

Já o comando “git pull” permite receber novos commits que tenham sido adicionados num repositório de origem. É tipicamente utilizado para receber as alterações enviadas por outros membros da equipa para um repositório partilhado, ou ainda para receber novos commits de um repositório específico de um membro da equipa.

E por fim, o comando “git push”, como o nome indica, faz exactamente o inverso do “git pull”, e serve para enviar as alterações efectuadas localmente no repositório, para um repositório de origem, tipicamente um repositório remoto partilhado com os membros da equipa.

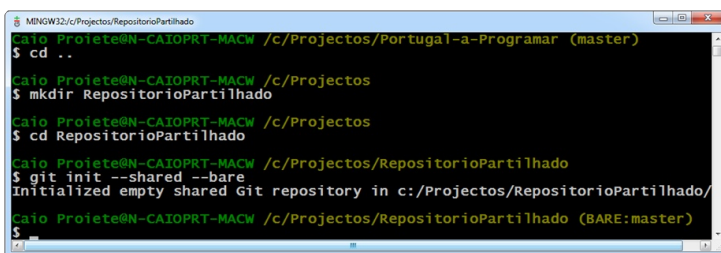
Criação de um Repositorio Partilhado

Um repositório partilhado pode estar no mesmo computador, ou em um computador remoto que pode estar na mesma rede, em uma rede separada, ou ainda em um servidor que pode aceder via Internet. A comunicação entre repositórios pode ser feita de diferentes formas, via rede (partilha de pastas), SSH, HTTP, HTTPS, entre outras formas. Para efeitos de exemplo, todos os repositórios

mostrados abaixo são criados no mesmo computador, em pastas diferentes.

Como explicado no início desta secção, a criação de repositórios é feita através do comando “git init”. No entanto, para a criação de repositórios partilhados que poderão receber actualizações (*push*) de outros utilizadores, é necessário indicar que trata-se de um repositório partilhado através dos parametros “--shared” e “--bare”.

```
$ cd ..  
$ mkdir RepositorioPartilhado  
$ cd RepositorioPartilhado  
$ git init --shared --bare
```



Após a execução das instruções acima, é criado um repositório chamado “RepositorioPartilhado” que irá servir como um repositório intermédio para dois membros da equipa, o “Tiago” e o “Carlos”, que terão cada um os seus próprios repositórios.

Criação de Clones de Repositórios

Para criar um clone de um repositório, como referido acima, utilizamos o comando “git clone”, informando o nome do repositório a ser clonado:

```
$ cd ..  
$ git clone RepositorioPartilhado Tiago  
$ git clone RepositorioPartilhado Carlos
```

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

```
MINGW32/c/Projectos
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/RepositorioPar
$ cd ..
Caio Proiete@N-CAIOPRT-MACW /c/Projectos
$ git clone RepositorioPartilhado Tiago
Cloning into Tiago...
done.
warning: You appear to have cloned an empty repository.
Caio Proiete@N-CAIOPRT-MACW /c/Projectos
$ git clone RepositorioPartilhado Carlos
Cloning into Carlos...
done.
warning: You appear to have cloned an empty repository.
Caio Proiete@N-CAIOPRT-MACW /c/Projectos
$
```

As instruções acima permitiram criar dois clones do repositório "RepositorioPartilhado", um para o "Tiago" e outro para o "Carlos", criados e pastas separada, e neste momento estão vazios (sem qualquer *commit* armazenado).

Para este exemplo, é criado um ficheiro no repositório do "Tiago" que será então armazenado em um *commit*. Em seguida este commit será enviado (push) para o repositório partilhado "RepositorioPartilhado", e a partir daí o "Carlos" pode obter as actualizações (pull) do repositório "RepositorioPartilhado" e consequentemente irá obter o commit efectuado inicialmente no repositório do "Tiago" e que foi partilhado no repositório "RepositorioPartilhado".

```
MINGW32/c/Projectos/Tiago
Caio Proiete@N-CAIOPRT-MACW /c/Projectos
$ cd Tiago/
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)
$ echo "Alteracao Tiago" >> NovaFuncionalidade.txt
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)
$ git add .
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)
$ git commit -m "Adiciona nova funcionalidade (Tiago)"
[master (root-commit) a0c5613] Adiciona nova funcionalidade
1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 NovaFuncionalidade.txt
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)
$
```

É importante reparar que o *commit* foi efectuado na branch "master" deste repositório.

Passo 2: Enviar as alterações para o repositório partilhado (push)

Como referido acima, para enviar as alterações efectuadas no repositório local para o repositório de origem, deve utilizar o comando "git push", e para isto deve informar o nome da referência do repositório de origem, e o nome da branch que deve ser considerada para o envio.

Ao efectuar um clone de um repositório, o Git automaticamente cria uma referência para o repositório de origem com o nome "origin". É possível alterar este nome se desejar, e também é possível criar outras referências para outros repositórios remotos.

Assim, para enviar as alterações para o repositório partilhado, basta informar "origin" como referência para o repositório partilhado, e "master" como nome da branch, uma vez que o commit que deve ser enviado está nesta *branch*.

```
$ git push origin master
```

```
MINGW32/c/Projectos/Tiago
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)
$ git push origin master
counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 260 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To c:/Projectos/RepositorioPartilhado
 * [new branch]      master -> master
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)
$
```



Passo 1: Efectuar as alterações no repositório do Tiago

```
$ cd Tiago/
$ echo "Alteracao Tiago" >>
NovaFuncionalidade.txt
$ git add .
$ git commit -m "Adiciona nova funcionalidade
(Tiago)"
```

TEMA DE CAPA

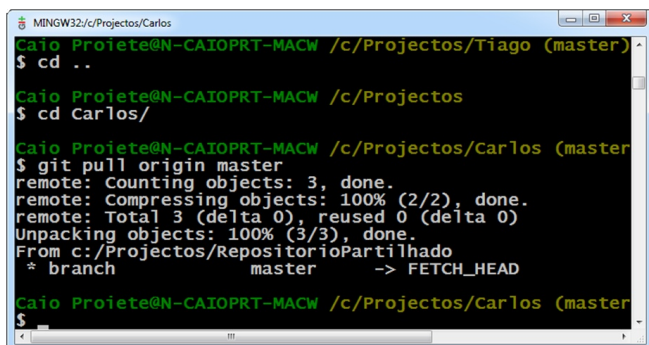
Git: Controlo de Versões para Pequenos e Grandes Projectos

Passo 3: Receber as alterações enviadas para o repositório partilhado no repositório do “Carlos”

Para receber as alterações existentes no repositório partilhado, deve utilizar o comando “git pull”, e para isto deve informar o nome da referência do repositório de origem, e o nome da *branch* que deve ser considerada para o recebimento.

Dessa forma, assim como o comando “git push” basta informar “origin” como referência para o repositório partilhado, e “master” como nome da *branch*.

```
$ cd ..  
$ cd Carlos/  
$ git pull origin master
```



```
MINGW32/c/Projectos/Carlos  
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Tiago (master)  
$ cd ..  
Caio Proiete@N-CAIOPRT-MACW /c/Projectos  
$ cd Carlos/  
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Carlos (master)  
$ git pull origin master  
remote: Counting objects: 3, done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 3 (delta 0), reused 0 (delta 0)  
Unpacking objects: 100% (3/3), done.  
From c:/Projectos/RepositorioPartilhado  
* branch          master       -> FETCH_HEAD  
Caio Proiete@N-CAIOPRT-MACW /c/Projectos/Carlos (master)  
$
```

E a partir de agora, os três repositórios estão sincronizados e possuem os mesmos commits.

Assim, o fluxo de trabalho comum no dia-a-dia do Git em equipa é algo como:

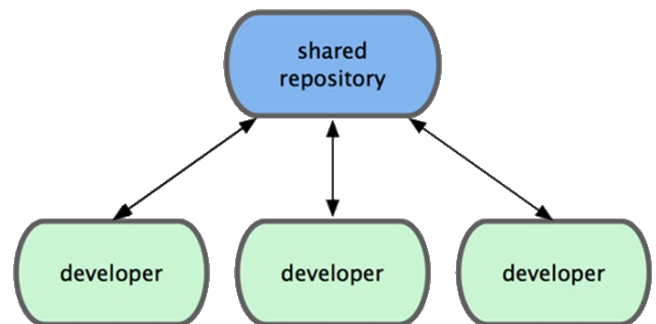
- inicio
- Efectuar alterações/novos *commits* (git commit)
 - Juntar actualizações do repositório partilhado com o repositório local (git pull)
 - Enviar as actualizações do repositório local para o repositório partilhado (git push)
- loop

Workflows Comuns para Controlo de Versões Distribuído

Subversion-style

Este é o *workflow* mais simples, e normalmente utilizado em equipas que estão a utilizar o Git pela primeira vez. Neste *workflow*, utiliza-se o Git como se fosse um sistema de controlo de versões centralizado, mas com as vantagens de um sistema distribuído, onde pode-se efectuar commits localmente, de forma desconectada, e enviar para o repositório partilhado apenas quando for apropriado.

Todas as alterações são partilhadas num repositório partilhado e não há comunicação directa entre os membros da equipa.



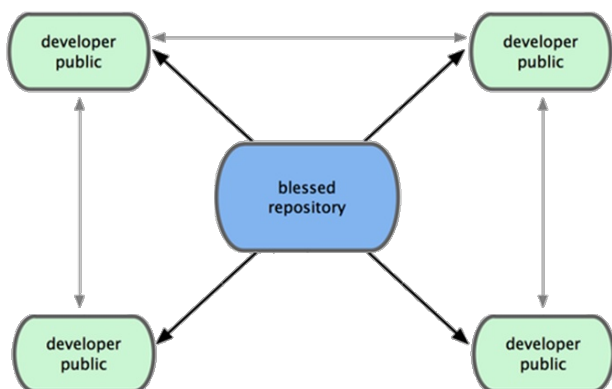
Decentralized but centralized

Este é o *workflow* mais comum para pequenos e médios projectos em equipas com alguma experiência com o Git. Os membros da equipa acordam entre si que todas as alterações que devem ser consideradas para as futuras versões do projecto serão armazenadas em um repositório partilhado principal, conhecido por “blessed repository” (repositório “abençoado”).

Os membros da equipa podem então partilhar commits entre eles directamente enquanto trabalham em determinadas tarefas, e quando for apropriado, podem enviar os *commits* para o *blessed repository*.

TEMA DE CAPA

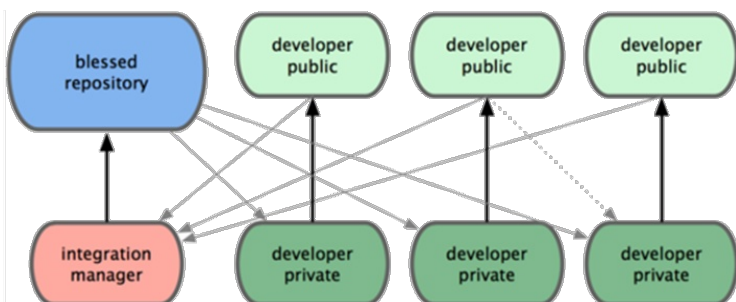
Git: Controlo de Versões para Pequenos e Grandes Projectos



Integration Manager

Este é um *workflow* mais sofisticado e indicado para projectos médios e grandes, onde cada membro da equipa possui dois repositórios, um público e outro privado (que podem estar no mesmo computador). Cada developer trabalha em seu repositório privado, e quando apropriado pode partilhar (*push*) as alterações que efectuou em seu repositório privado no repositório público.

Existe então uma pessoa da equipa que assume o papel de "Integration Manager" que é a pessoa responsável em obter (*pull*) as alterações do repositório público de cada developer, validar, e então enviar (*push*) para o *blessed repository*.



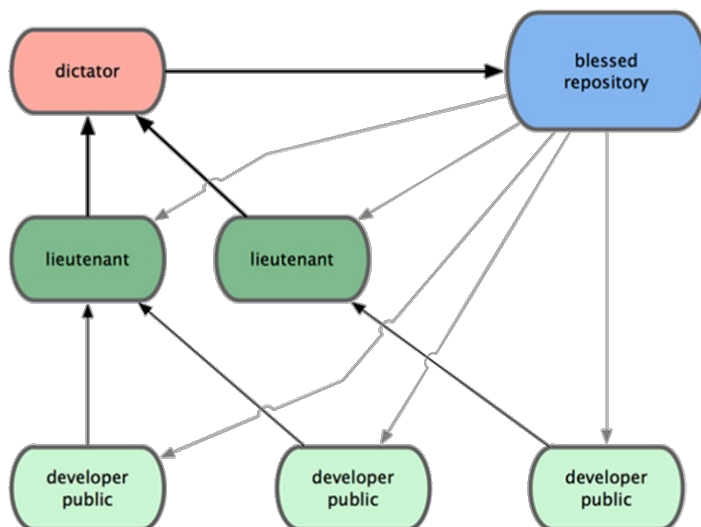
Dictator and Lieutenants

Este é um workflow ainda mais sofisticado que o anterior e indicado para projectos extremamente grandes e com muitas pessoas a participar no desenvolvimento.

Cada developer possui um repositório público onde pode partilhar as alterações que posteriormente serão validadas por pessoas na equipa que assumem o papel de "liutenant"

(tenente) e normalmente são responsáveis por módulos específicos do projecto.

Após a validação das alterações pelos liutenants, estas são enviadas para outra pessoa que assume o papel de "dictator" (ditador) e que efectua uma validação final, antes de enviar para o *blessed repository*.



Por curiosidade, este é o *workflow* utilizado actualmente para controlar as versões do kernel do Linux. Existem pessoas consideradas como sendo pessoas "de confiança" e que são responsáveis por diferentes módulos do kernel e validam as alterações enviadas pelas centenas de pessoas que contribuem para o projecto, e por fim são enviadas para o "ditador" que as valida e escolhe quais alterações farão parte do repositório principal e que eventualmente irão fazer parte de uma futura versão do sistema operativo.

Serviços de Alojamento de Repositórios Git na Internet

Existem dezenas de empresas que fornecem serviços de alojamento de repositórios Git na Internet, permitindo desenvolver projectos (open-source ou não) com equipas distribuídas sem precisar criar e manter uma infra-estrutura própria.

O serviço mais popular e provavelmente o mais utilizado em todo o mundo é o GitHub (<http://github.com>) que oferece a possibilidade de criar repositórios públicos

TEMA DE CAPA

Git: Controlo de Versões para Pequenos e Grandes Projectos

gratuitos com até 300 Mb, para quem pretende desenvolver software open-source, e oferece também a possibilidade de criar repositórios privados para empresas que queiram ter repositórios privados partilhados e não possui infra-estrutura própria com preços que variam entre os \$7 e \$22 dólares americanos por mês no momento em que escrevo este artigo. Em Abril de 2011, o GitHub ultrapassava os 2 milhões de repositórios.

Outros exemplos serviços que oferecem serviços semelhantes são:

- Gitorious (<http://gitorious.org>)
- Unfuddle (<http://unfuddle.com>)
- ProjectLocker (<http://www.projectlocker.com>)
- RepositoryHosting (<http://repositoryhosting.com>)
- Assembla (<http://www.assembla.com>)

Exemplos de Grandes Projectos que Utilizam o Git

O Git é amplamente utilizado em projectos open-source em todo o mundo, em pequenos, médios e grandes projectos, e foi originalmente desenvolvido para controlar as versões do kernel do Linux e desde a sua primeira versão continua a ser utilizado para tal.

Alguns projectos populares que utilizam Git para controlo de versões, além do kernel Linux, são: Ruby on Rails, Node.js, jQuery, Modernizr, Scriptaculous, Android, CakePHP, Sinatra, VLC, entre muitos outros, e o próprio Git. Exacto! O controlo de versões do código-fonte do Git é feito através do próprio Git.

Uma lista mais detalhada de projectos que utilizam o Git está disponível no Wiki do Git em <https://git.wiki.kernel.org/index.php/GitProjects>, e pode acompanhar os projectos open-source mais populares no GitHub em <https://github.com/popular/watched>.

Links para Referência

Git Scm - Site oficial do Git

<http://git-scm.com>

Posts sobre Git em meu blog

<http://caioproiete.net/pt/tag/git/>

Vídeo: Controlo de Versões Distribuído com Git

<http://vimeo.com/20652754>

Pro Git (e-book)

<http://progit.org>

Git Ready (tutorial / tips)

<http://www.gitready.com>

Git Magic (e-book)

<http://www-cs-students.stanford.edu/~blynn/gitmagic>

Git for Beginners

<http://stackoverflow.com/questions/315911/git-for-beginners-the-definitive-practical-guide>

Why Git is Better than X

<http://whygitisbetterthanx.com>

Git Is Your Friend not a Foe

<http://hades.name/blog/2010/01/17/git-your-friend-not-foe>

A successful Git branching model

<http://nvie.com/posts/a-successful-git-branching-model>

Use Git For What It Is Not Intended (UGFWIINI)

<http://thread.gmane.org/gmane.comp.version-control.git/110411>

AUTOR



Escrito por **Caio Proiete**

Exerce as funções de arquitecto de software e analista-programador numa multinacional sediada em Portugal, e ministra cursos técnicos de formação na Ciclo (<http://ciclo.pt>). É formador certificado pela Microsoft (MCT), Microsoft Certified Professional Developer (MCPD) nas áreas Windows, Web e Enterprise em .NET 4.0, é Microsoft Most Valuable Professional (MVP) em ASP .NET desde 2009, e líder da Comunidade NetPonto (<http://netponto.org>), onde organiza reuniões presenciais todos os meses, e apresenta sessões de assuntos relacionados com desenvolvimento de software na plataforma Microsoft .NET. É autor do blog <http://caioproiete.net> - Twitter: [@CaioProiete](https://twitter.com/CaioProiete)

A PROGRAMAR

Lua – Linguagem de Programação (Parte 9)

Introd. Cloud Computing e à Plataforma Windows Azure

Managed Extensibility Framework (MEF) e AJAX

Microsoft BizTalk Server aos olhos dos programadores

O Editor de texto VIM

Lua – Linguagem de Programação (Parte 9)

Este artigo apresenta uma solução para o uso e implementação do operador lógico xor em linguagem Lua. Apresenta também instruções de uso e criação de módulos que são as bibliotecas de funções externas que podem ser criadas pelos próprios programadores.

ALGO A MAIS EM LUA

A título de ilustração sobre recursos variados que podem ser utilizados na linguagem Lua segue alguns poucos exemplos, como: modo de limpeza do ecrã e medição do tempo de CPU.

Uma forma de efectuar limpeza do ecrã (tela, monitor ou monitor de vídeo no Brasil) é executar o comando cls na janela de prompt de comando do Microsoft Windows (modo MS-DOS) ou o comando clear na janela de comando do Linux/UNIX. Outra maneira é por meio de uso dos recursos de terminal ANSI, mas este será assunto para outro momento.

Para fazer a execução do comando de limpeza de ecrã cls ou clear deve-se fazer uso da função execute da biblioteca os com a sintaxe os.execute("comando"), onde comando será substituído pelo comando de limpeza de ecrã do sistema operacional em uso.

Para fazer a métrica de tempo de execução de uso de certo recurso em linguagem Lua usa-se a função clock da biblioteca os a partir da sintaxe os.clock() que retorna o valor aproximado em segundos do tempo gasto para a execução do recurso medido.

Para efectuar um teste da função os.clock() considere um programa que apresente o resultado de uma tabuada de um número qualquer variando de 1 até 1000, como segue.

```
-- inicio do programa TABUADA

print("Programa Tabuada")
print("\n")
io.write("Entre um numero tabuada: ")
N = io.read("*number")
print("\n")

local TEMPO = os.clock()

for I = 1, 1000, 1 do
    R = N * I
    io.write(string.format("%4d", N))
    io.write(" X ")
    io.write(string.format("%4d", I))
    io.write(" = ")
    io.write(string.format("%5d", R))
    io.write("\n")
end

local R = os.clock() - TEMPO

io.write("\nTempo = " .. R)
io.write(" segs.\n")

-- fim do programa TABUADA
-- inicio do programa OPER_XOR
```

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome tabuada.lua e execute-o com a linha de comando lua 5.1 tabuada.lua.

OPERADOR XOR

Na linguagem de programação Lua não há a existência do operador lógico xor como não há tal operador também na linguagem C, a não ser quando se trabalha com operações de mais baixo nível com a manipulação de bits por meio do operador “^”.

A não existência do operador lógico xor não desmerece em nada a linguagem Lua, mas parece criar em alguns programadores certo desconforto por não conhecerem ou não saberem como resolver a questão.

Apesar de ser uma solução muito simples e de certa maneira fácil de ser encontrada em bons livros, sítios ou blogs que tratam sobre o tema da programação de computadores cabe neste espaço mostrar a solução para a linguagem Lua, que nada mais é do que uma solução meramente matemática.

O operador lógico XOR retorna o resultado verdadeiro quando apenas, e tão-somente, uma das condições da expressão lógica é verdadeira. No caso em que as condições avaliadas sejam todas falsas ou verdadeiras o resultado da expressão lógica será falso. Assim sendo, considere a tabela verdade a seguir para o operador lógico xor:

Condição 1	Condição 2	Resultado lógico
Verdadeiro	Verdadeiro	Falso
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso

A solução para uso da expressão lógica: (C1) xor (C2) em uma linguagem de programação, onde C1 e C2 são condições a serem avaliadas é usar a expressão lógica (C1 and (not C2)) or ((not C1) and C2).

No entanto, o uso desta expressão lógica na linguagem Lua não surte o efeito esperado. Isto posto, passa-se a ter outro problema a ser resolvido. Como então fazer o uso de tal necessidade? Uma solução é recorrer a um método no estilo “força bruta”.

No sentido de exemplificar o uso da ação de aplicação do conceito do operador lógico xor (ou exclusivo) em Lua considere um programa que efectue a entrada dos nomes e sexos de duas pessoas que pretendem formar um par para participar de uma dança de quadrilha. Os administradores da festa determinaram que somente serão aceitos pares de sexos heterogéneos. Não serão aceitos casais formados por pessoas do mesmo sexo. Para atender a condição estabelecida o programa deve, após a entrada do sexo dos participantes, verificar se formam par, e neste caso apresentar uma mensagem informando esta possibilidade. Caso não seja a condição verdadeira o programa deve indicar a impossibilidade da composição do par de dança. Observe que serão aceitos pares caso o sexo do 1º participante seja masculino e do 2º participante for feminino ou vice-versa. Assim sendo, considere como exemplo o seguinte código de programa:

```
io.write("Nome 1o. dançarino: ")
N1 = io.read()
repeat
    io.write("Sexo 1o. dançarino: ")
    S1 = string.upper(io.read())
until (S1 == "M") or (S1 == "F")

io.write("Nome 2o. dançarino: ")
N2 = io.read()
repeat
    io.write("Sexo 2o. dançarino: ")
    S2 = string.upper(io.read())
until (S2 == "F") or (S2 == "M")

if (S1 == "M") and (S2 == "F") or
(S1 == "M") and (S2 == "F") or
(S1 == "F") and (S2 == "M") or
(S1 == "F") and (S2 == "M") then
    print(N1 .. " dança com " .. N2)
else
    print(N1 .. " nao dança com " .. N2)
end
-- fim do programa OPER_XOR
(S1 == "M") and (S2 == "F") or
```


A PROGRAMAR

Lua – Linguagem de Programação (Parte 9)

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome `oper_xor.lua` e execute-o com a linha de comando `lua 5.1 oper_xor.lua`.

Observe no programa o uso da instrução de laço `repeat...until` na entrada dos sexos dos participantes no sentido de evitar que ocorram entradas de sexo que não sejam M para masculino ou F para feminino. Outro detalhe a ser observado é o uso da função `string.upper()` que formata uma entrada de texto para maiúsculo.

O trecho escrito entre a instrução `if...then` faz o uso de uma ação de força bruta para a simulação da execução do operador lógico `xor` com o trecho de código seguinte.

```
(S1 == "M") and (S2 == "F") or
(S1 == "F") and (S2 == "M") or
(S1 == "F") and (S2 == "M")
-- inicio do programa MODULO01
```

A solução indicada pode não ser elegante, mas é funcional e atende a necessidade de solução do problema, pois quem programa um computador é um programador e não a linguagem em si.

MÓDULOS

Na estrutura operacional de Lua chama-se módulo a biblioteca de funções e variáveis externa contidas em uma tabela de cunho global utilizada por meio da função `require()`.

Há também a possibilidade de se trabalhar com módulos a partir da função `module()`, mas este não será o foco tratado neste artigo.

A linguagem Lua possui alguns módulos em sua biblioteca padrão (biblioteca interna), sendo:

- **coroutine**: possui as funções de uso do recurso de corrotinas;

- **debug**: funções para processo de depuração;
- **io**: possui as funções para as operações de entrada e saída;
- **math**: possui as funções para o uso de operações matemáticas;
- **os**: funções que facilitam operações com o sistema operacional;
- **package**: possui funções para o tratamento de módulos;
- **string**: possui as funções que manipulam cadeias de caracteres;
- **table**: possui as funções para a manipulação de tabelas.

Os módulos padrão da linguagem Lua são automaticamente carregados quando do uso do interpretador. Não havendo necessidade de uso da função `require()`.

O uso da função `require()` é obrigatório quando da definição e uso de uma biblioteca externa e particular criada pelo programador para atender as suas próprias necessidades. Para fazer uso de módulos de forma simples considere como exemplo o seguinte código de programa:

Grave o código do programa anterior com o nome `modulo01.lua`.

```
function saudacao(NOME)
    print("Olá, " .. NOME)
end

function raiz(BASE,INDICE)
    local X = BASE ^ (1 / INDICE)
    return X
end

-- fim do programa MODULO01
-- inicio do programa MODULO02
```

Em seguida escreva o próximo código de programa em um editor de texto, gravando-o com o nome `modulo02.lua` no

A PROGRAMAR

Lua – Linguagem de Programação (Parte 9)

mesmo local do programa modulo01.lua e execute-o com a linha de comando lua 5.1 modulo02.lua.

Ao ser o programa modulo02.lua executado será aberto o acesso a biblioteca modulo01 por meio da execução da linha de código require("modulo01").

```
require("modulo01")

print("Seja bem vindo, visitante")
io.write("Informe se nome: ")
N = io.read()
saudacao(N)

io.write("Me de uma base .....: ")
B = io.read("*number")
io.write("Me de um índice .....: ")
I = io.read("*number")

R = raiz(B, I)
io.write("Resultado = " .. R, "\n")

-- fim do programa MODULO02
```

A partir deste momento ficam disponíveis para o programa modulo02 as funções da biblioteca externa modulo01, as quais são executadas quando do uso das linhas de instrução saudacao(N) e raiz(B,I).

É possível encontrar na Internet bibliotecas para a linguagem Lua disponibilizadas por outros programadores. Caso interesse-se por este tema poderá consultar o sítio <http://www.tecgraf.puc-rio.br/~lhf/ftp/luas/> o qual encontrava-se no ar até Abril de 2011 quando este artigo estava sendo finalizado.

As bibliotecas externas para uso em na linguagem Lua poderão possuir a extensões de identificação .lua ou .so (ou .dll no caso do sistema operacional da Microsoft).

CONCLUSÃO

Neste artigo foi visto os procedimentos de uso e criação de bibliotecas externas e a simulação de uso do operador lógico xor.

No que tange ao uso do recurso de módulos o assunto é mais extenso do que o apresentado. O leitor poderá aprofundar-se nesta temática com a leitura do livro Beginning Lua Programming publicado pela editora Wiley Publishing, Inc. dos autores Kurt Jung e Aaron Brown.

No próximo a intenção e abordar a temática relacionada ao uso de co-rotinas e a integração da linguagem Lua com outras linguagens de programação.

AUTOR



Escrito por **Augusto Manzano**

Natural da Cidade de São Paulo, 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.

Introdução ao Cloud Computing e à Plataforma Windows Azure

Nos últimos tempos muito se tem falado sobre o Cloud Computing/Cloud, mas a verdade é que muita gente ainda se questiona sobre: O que é? Para que serve? E a mais importante de todas as questões; O que é que isto me ajuda? Tendo estas questões em mente, e também após o desafio lançado pelos responsáveis da revista Portugal-a-Programar, decidi escrever este artigo para clarificar estas e outras questões.

O Cloud Computing embora se tenha vindo a falar muito ultimamente, não é uma revolução, mas sim uma evolução da computação, pois evoluiu naturalmente da adopção em massa da Virtualização, e também no seguimento de outros tipos de computação distribuída, como por exemplo o Grid Computing.

Mas para melhor compreendermos o conceito de Cloud Computing o melhor é olhar-mos para uma pequena analogia, e para isso o mais simples é olhar-mos para algo que todos conheçamos e também sejamos consumidores, como por exemplo a energia eléctrica ou mesmo a água. Vamos então olhar para o exemplo da água (uma vez que a versão energia eléctrica está deveras utilizada). A realidade é que durante muitos anos cada um de nós éramos responsáveis por produzir a água que consumia-mos, fosse através de ir buscar água aos rios/ fontes, ou mesmo criarmos um nosso próprio poço para que pudéssemos ter mais facilmente a nossa água disponível.,

Mas a questão importante é que nenhuma das opções era simples, e requeria acima de tudo bastante trabalho da nossa parte, pois quer fosse por transportarmos a água ou mesmo pela manutenção do nosso poço, muito trabalho e dinheiro era gasto no processo. Tudo isto foi bastante normal até um determinado momento em que surgiu uma rede de água, que tinha a vantagem de ser simples, sem qualquer esforço necessário da nossa parte para que tudo funcionasse, e acima de tudo com um valor associado à utilização que fazíamos da mesma, o que tornava os investimentos anteriores completamente desadequados.



Com o Cloud Computing aconteceu exactamente a mesma coisa. Até este momento todos nós temos andado a procurar formas de obtermos mais capacidade de computação, não sendo fácil nem mesmo interessante em termos de investimento. Também em termos de disponibilidade tem sido muito complicado, pois qual de nós ainda não passou pela experiência de identificar a necessidade de aumento do número de servidores, e “desesperar” pela disponibilização do mesmo, e isto independentemente de sermos nós ou um serviço de Hosting a gerir o Data Center. Aqui é onde esta nova forma de computação chama mais à atenção, uma vez que podemos ter quase no “imediato” disponível o poder de computação que necessitamos, e acima de tudo pagando apenas pelo que utilizamos, e não pelo simples facto de existirem os servidores, como estávamos habituados até aqui.

Como curiosidade, por vezes surgem questões

A PROGRAMAR

Introdução ao Cloud Computing e à Plataforma Windows Azure

relativamente ao porquê da palavra Cloud. Este termo deve-se ao facto de a Internet ter sido representada ao longo dos anos com uma nuvem, e por isso mesmo o termo Cloud/Nuvem, ser utilizado, pois estamos a falar de um tipo de computação acessível via Internet.

Características

Uma questão importante a ter em conta é o facto de ser necessário compreendermos exactamente o que é e o que não é Cloud Computing, pois ao longo dos anos já assistimos por diversas vezes a situações em que utilizam nomes novos para reciclar ofertas antigas. Por isso mesmo existem algumas características principais para que possamos identificar as ofertas mais rapidamente, sendo elas:

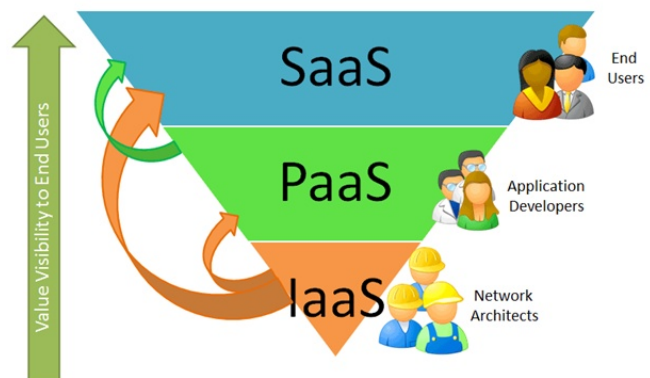
- **Elasticidade / Agilidade** – É possível ou não aumentar e diminuir a capacidade de computação rapidamente (menos de 1 dia)?
- **API** – Existem ou não uma API capaz de nos permitir interagir programaticamente com o sistema?
- **Custo** – Os custos são associados à utilização que fazemos?
- **Multi-Tenancy** – O sistema suporta mais do que um cliente sem quaisquer alterações?
- **Fiabilidade** – Existe ou não redundância e tolerância à falha?
- **Escalabilidade** – É fácil ou não escalar soluções e recursos?
- **Segurança** – Existem ou não acordos de nível de serviço?

“ Se respondeu sim a todas as questões anteriores, então está certamente perante uma verdadeira oferta de Cloud Computing.”

Se respondeu sim a todas as questões anteriores, então está certamente perante uma verdadeira oferta de Cloud Computing.

Modelos de Utilização

Em conjunto com o Cloud Computing surgiram outras siglas que têm sido muito badaladas como o SaaS (Software como um Serviço), PaaS (Plataforma como um Serviço) e IaaS (Infra-estrutura como um Serviço). Mas o que são na realidade? Olhando de uma perspectiva de alto nível, elas definem a responsabilidade que o cliente sobre o serviço:



- **SaaS** – O cliente apenas se preocupa com a utilização da aplicação e nada mais, pois os dados, aplicação e respectiva infra-estrutura passam a ser responsabilidade do fornecedor do serviço. (ex. Office 365, Google Apps, Salesforce.com, ...)
- **PaaS** – O cliente preocupa-se apenas com a construção da aplicação e respectiva gestão dos dados da mesma, pois toda a restante infra-estrutura e plataforma onde as mesmas irão correr passam a ser responsabilidade do fornecedor do serviço. (Ex. Windows Azure, SQL Azure, Amazon AWS, Google App Engine, Force.com, ...)
- **IaaS** – A responsabilidade prende-se com a gestão de todos os elementos “virtuais” da infra-estrutura, como o Sistema Operativo, Serviços de Suporte, etc., mas não se preocupando com qualquer questão de gestão física dessa mesma infra-estrutura, pois a mesma será responsabilidade do fornecedor desse mesmo serviço. (Ex. Amazon EC2, vCloud Express, Hyper-V Cloud, ...)

A PROGRAMAR

Introdução ao Cloud Computing e à Plataforma Windows Azure

Modelos de Disponibilização

Sempre importante é também compreendermos as formas como podemos encontrar o Cloud Computing disponível, pois ao contrário do que se possa pensar o Cloud Computing não tem a ver apenas com computação que se encontra disponível publicamente (embora exista um grande grupo que o defenda), pois na realidade existem 4 tipos de disponibilização que poderemos encontrar, sendo eles os seguintes:

- **Pública** - Cloud Computing de uma forma tradicional (como é maioritariamente vista), onde os recursos são provisionados dinamicamente e com enorme detalhe, mas sempre de uma forma autogerida via Internet através de Aplicações ou Serviços Web, um parceiro que nos cobra a um nível de detalhe na base do utility computing.
- **Privada** - Capacidade de ter todas as capacidades de uma Cloud interna a uma organização.
- **Comunitária** - Poderá ser estabelecida entre organizações que tenham requisitos semelhantes e procurem partilhar infra-estrutura. Exemplo deste tipo de modelo é a nuvem comunitária da Google "Gov Cloud".
- **Híbrido** - Este termo tem sido utilizado como significando quer duas nuvens integradas (pública, privada, interna ou externa), quer a combinação de instâncias virtualizadas em conjunto com hardware real.

Plataforma Windows Azure



Windows Azure

Agora que já conhecemos melhor o que é efectivamente o Cloud Computing, podemos iniciar uma visão mais detalhada sobre uma das ofertas de Cloud sobre a qual muito se tem falado, e ela é a Plataforma Windows Azure da Microsoft. No que respeita ao modelo de

disponibilização esta é uma oferta de Cloud Pública, e fornece-nos um modelo de utilização PaaS (Plataforma como um serviço).

Tendo terminado o processo inicial de categorização da oferta, passemos então a compreender melhor aquilo que é então a Plataforma Windows Azure.

Na realidade esta plataforma assenta muito na reutilização das competências, pois encontra-se assente nas tecnologias que a Microsoft tem vindo a disponibilizar ao longo dos anos para o desenvolvimento de soluções, permitindo que o processo de desenvolvimento para a Cloud seja mais simples, uma vez que utiliza ferramentas e frameworks que programadores, gestores, e todos os outros interlocutores no processo de desenvolvimento de aplicações já se encontram familiarizados, mas não só, pois também é possível utilizar esta oferta com frameworks e ferramentas não Microsoft, como são os casos do Java, PHP, Ruby, Python, entre outros.

Dado isto, uma das questões que se levanta é: "então afinal o que pode correr em Windows Azure?".

A resposta não poderá ser mais simples, pois salvo raras excepções, tudo aquilo que seja possível instalar num sistema operativo Windows Server 2008 SP2 ou Windows Server 2008 R2, poderá ser disponibilizado na plataforma Windows Azure. Este é um aspecto muito importante pois não é uma plataforma limitada à disponibilização de soluções desenvolvidas apenas numa tecnologia específica, mas sim num sistema global com capacidade de disponibilizar uma grande maioria das soluções que têm vindo a ser desenvolvidas, pois consegue abranger um grande leque de tecnologias.

Então agora que já sabemos o que podemos colocar, vamos olhar para a sua composição, pois a mesma é composta 4 áreas principais:

- **Windows Azure** - Muitas vezes denominado de Sistema Operativo da Cloud, fornece serviços de computação, armazenamento, e também automatização na gestão dos mesmos. É este o responsável por correr todas as nossas

A PROGRAMAR

Introdução ao Cloud Computing e à Plataforma Windows Azure

soluções disponibilizadas na plataforma, bem como tratar dos dados associados à mesma.

- **Windows Azure AppFabric** – Tem como objectivo fornecer serviços aplicativos para consumo nas diversas aplicações, sejam eles ao nível da Identidade e Controlo de Acessos, até a serviços de conectividade e caching. Disponibiliza acima de tudo um conjunto de serviços que permitem tornar as nossas soluções mais ricas, de uma forma simplificada.

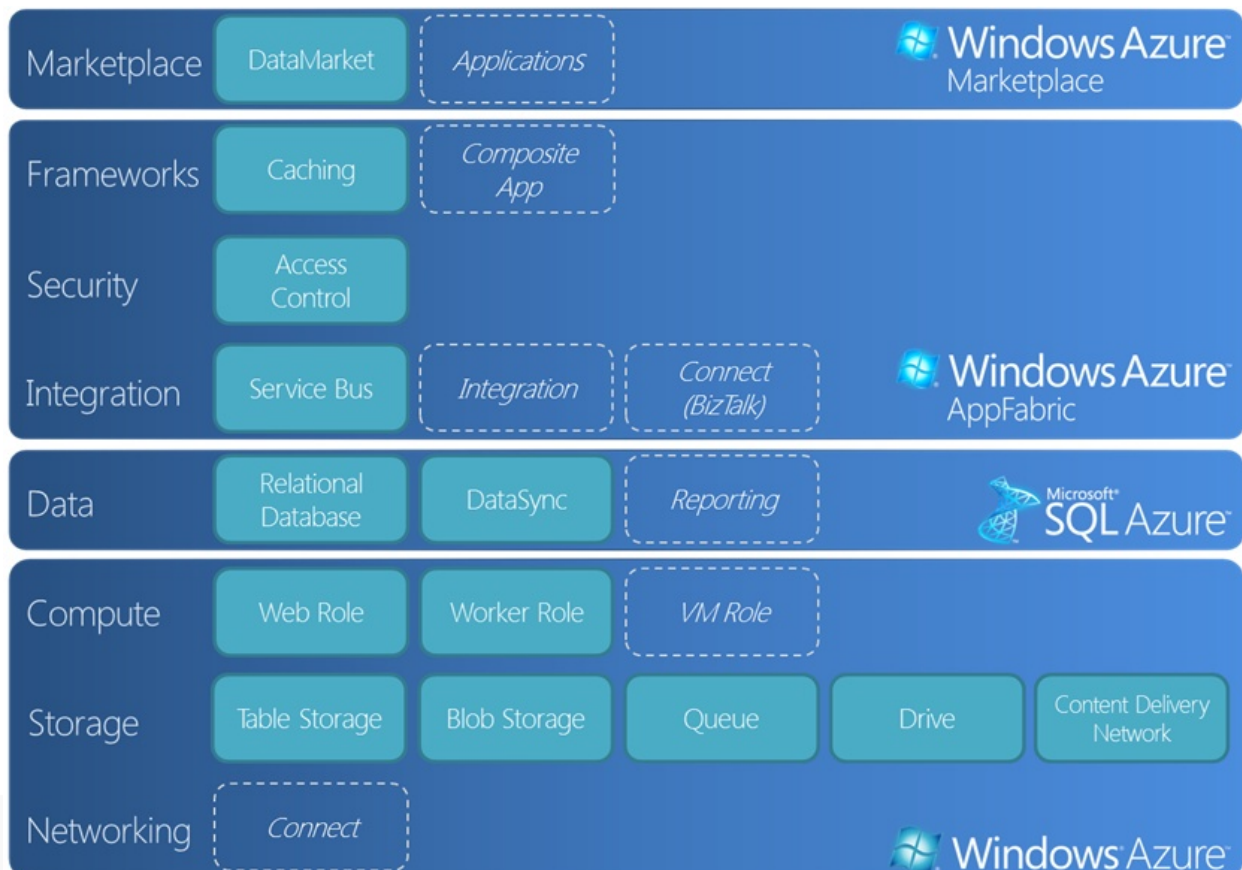
- **SQL Azure** – Serviços de dados relacionais como por exemplo uma base de dados relacional (SQL Azure), um sistema de relatórios (SQL Azure Reporting Services) e também um sistema de sincronização de dados (SQL Azure DataSync), permitindo que possamos criar soluções que tirem partido quer de uma base de dados relacional altamente distribuída e disponível, entre outros serviços muito importantes quando trabalhamos com dados.

- **MarketPlace** – Plataforma para a monetização de aplicações e dados. Este é um dos pontos muito interessantes da plataforma, pois não se limita apenas a fornecer um conjunto de serviços para “criar” e “correr” aplicações, mas também uma forma de monetizar os

mesmo, uma vez que permite ao fornecedor de serviços, descrever a sua oferta e o valor a ser cobrado pela mesma.

Olhando então numa perspectiva funcional e de alto nível, esta plataforma fornece-nos os seguintes serviços:

- **Ao nível com Computacional**, o Windows Azure Compute, que é nada mais do que um dos principais blocos desta oferta, pois para além de definir o que deverá “correr” em termos de computação, permite descrever também a forma como a mesma deverá ser efectuada, seja ao nível da capacidade de processamento, até ao nível da memória alocada a esse processo de computação. É também aqui que acontece o processo que permite fornecer a elasticidade à solução, pois através do conceito de instância (ambiente de computação funcional de acordo com as regras definidas na configuração do Serviço) permite definir para cada um dos serviços disponibilizados na plataforma, quantas instâncias deverão estar funcionais ao mesmo tempo, detendo ainda a capacidade de analisar a “saúde” das mesmas por forma a evitar problemas e, ao mesmo tempo, resolver rápida e automaticamente todos os



A PROGRAMAR

Introdução ao Cloud Computing e à Plataforma Windows Azure

problemas que possam surgir. É aqui que muita da “magia” acontece, pois é através deste serviço que podemos ter o sistema elástico necessitamos em tantas situações.

• **No que respeita ao nível de Armazenamento**, propõe duas opções:

o **Windows Azure Storage** – Sistema de Armazenamento disponibilizado como um serviço. Permite-nos focar apenas nos dados armazenados e não na forma como se encontram armazenados ou até dos processos necessários para os tornar mais disponíveis. É composto por 4 abstrações diferentes: Tables (Base de Dados Hierárquica, não relacional), Queues (Filas de Trabalho que nos irão permitir a criação de soluções assíncronas, e ao mesmo tempo tolerantes a falhas), Blobs (Ficheiros) e Drives (Simulação de um Disco que poderá ser utilizado pelos serviços de computação).

o **SQL Azure** – Base de Dados Relacional disponibilizada como um serviço, sendo fácil de provisionar e disponibilizar, mas acima de tudo preparada para ser altamente disponível e tolerante a falhas. Mantém também ao mesmo tempo a mesma forma de utilização que já é habitual para consumir e gerir dados de qualquer base de dados SQL Server.

Uma nota importante ao nível do armazenamento é que a plataforma Windows Azure efectua em qualquer dos casos a 3 réplicas dos dados que são armazenados, garantindo desta forma uma maior segurança e tolerância a falha dos mesmos. Isto faz com que o grau de segurança que obtemos ao utilizar este tipo de armazenamento é bastante elevado, pois encontramos-nos protegidos em caso de falha do sistema.

• **Em termos de Conectividade**, as opções são:

o **Rede Virtual** - Mecanismo simples e de fácil manutenção que nos permite criar conectividade/rede virtual entre recursos que se encontrem na Cloud com outros que se encontrem On-Premise (internamente nas organizações), sem que para isso seja necessário as longas e complexas configurações de rede, habitualmente necessárias sempre que seja necessário ligar recursos de

fora de uma determinada organização a recursos internos à mesma.

o **Windows Azure AppFabric ServiceBus** – Permite fornecer uma forma de criar e consumir serviços de uma forma simples e poderosa, ficando para o criador do serviço apenas o que diz respeito às regras de negócio que o mesmo deverá disponibilizar, e também a forma como o serviço deverá ser disponibilizado, ficando todo o processo de disponibilização e manutenção do mesmo a cargo da plataforma.

o **Windows Azure AppFabric Integration** – Proporciona a simplificação do processo de criação de serviços para integrar diferentes soluções e até transformação de dados.

o **Windows Azure AppFabric Caching** – Garante a possibilidade de obter de uma forma simples e rápida um ambiente de Caching altamente distribuído e disponível, auxiliando dessa forma a melhoria da performance das soluções desenvolvidas.

o **CDN (Content Delivery Network)** – Serviço que nos irá permitir melhorar o acesso quer a elementos presentes no serviço de armazenamento do Windows Azure ou até mesmo ao nível da computação. É composto por um conjunto de cerca de 22 nós dispersos geograficamente que têm como responsabilidade disponibilizar os conteúdos requeridos pelo cliente rapidamente e com tendo por base a localização do mesmo.

• **Quanto à Identidade** – Windows Azure AppFabric Access Control que nos permite definir a forma como a identidade das nossas soluções irá ser efectuada, inclusivamente disponibilizando uma forma simples de implementar o Single-Sign On, que à tanto tempo se vem falando. É sem dúvida uma enorme mais-valia para as soluções uma vez que cria uma camada de abstracção sobre a forma como o utilizador é identificado, pois essa transformação, entre a origem do utilizador até ao que a nossa solução necessita de saber sobre esse mesmo utilizador, é-nos fornecida e tratada por este serviço.

Também muito importante, e de acordo com o que vimos no que respeita ao Cloud Computing, é compreender o Acordo de Nível de Serviço (SLA) associado a esta oferta da Microsoft, pois é através dele que garantimos o nível de confiança na plataforma. Os SLAs existentes são os de

A PROGRAMAR

Introdução ao Cloud Computing e à Plataforma Windows Azure

Computação que nos garantem 99,95% de disponibilidade ao nível de soluções Web, e 99,9% para todos os restantes serviços de computação, sendo para isso necessário que existam pelo menos duas instâncias activas desse mesmo serviço. Quanto ao SQL Azure o SLA garante 99,9% de disponibilidade da mesma forma como acontece com o Windows Azure AppFabric e Windows Azure CDN. Estes SLAs têm também um aspecto muito importante que se prende com o facto de fornecerem uma compensação monetária ao cliente em caso de incumprimento.

Conclusão

Tendo em consideração tudo o que falamos, poderemos considerar o Cloud Computing como a próxima geração de computação, e mais importante é que não é algo que apenas poderemos começar a tirar partido num futuro próximo, mas sim JÁ, pois é algo que já está disponível e pronto para utilizarmos de forma a melhorarmos os nossos investimentos.

“ não é algo que apenas poderemos começar a tirar partido num futuro próximo, mas sim já ”

É também importante ter em atenção que não estamos a

falar de algo que é importante apenas para alguns, mas sim para todos, quer seja através da utilização de uma simples aplicação como o CRM na Cloud, ou até ao desenvolvimento de novas soluções de negócio e até formas nichos de mercado, o Cloud Computing irá ajudar-nos a atingirmos os nossos objectivos, de uma forma mais ágil, económica e adequada às necessidades. E lembrem-se que como diz o ditado popular “nem tudo o que luz é ouro”, por isso sempre que lhes apresentem uma solução denominada de Cloud Computing, sejam capazes de vocês próprios a avaliar, e compreender se é ou não na verdade.

Também importante é compreender que o Cloud Computing não é a solução para todos os problemas, mas sim mais uma opção para tirarmos o máximo partido dos investimentos que fazemos.

Podemos também olhar para a oferta Windows Azure da Microsoft e considerar a mesma como uma aposta séria e de elevada qualidade, pois não só nos permite colocar os recursos que deverão ser processados, como também nos fornece um enorme conjunto de serviços para melhorar as nossas soluções, mantendo ou mesmo aumentando disponibilidade das soluções.

Espero que com este artigo vos tenha ajudado a compreender melhor o que é afinal o Cloud Computing e a Plataforma Windows Azure e ao mesmo tempo desmistificar um pouco esta questão. Muito mais poderia ser dito, mas parece-me que aqui ficou o essencial.

AUTOR



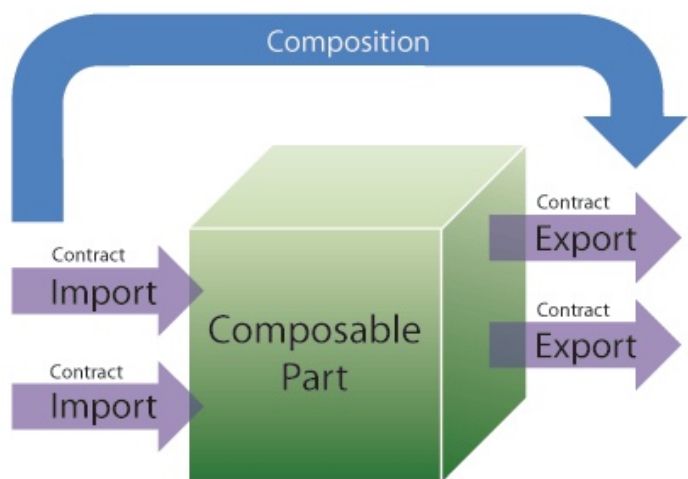
Escrito por **Nuno Godinho**.

Consultor Independente com 10 anos de Experiência e principal responsabilidade de ajudar os clientes a identificar, planear, gerir e desenvolver soluções e produtos de software. Especialista em Tecnologias Microsoft. Orador em alguns dos maiores eventos de desenvolvimento da Microsoft Portugal como MSDN, TechDays, DevDays, além de eventos internacionais como TechEd Europa, TechEd Online Worldwide, MVP Nation e CloudViews.Org. Microsoft MVP há 4 anos, inicialmente em ASP.NET e a partir do início deste ano em Windows Azure com blogs em <http://www.pontonnetpt.com/blogs/nunogodinho> (Português e Inglês) e <http://www.msmvps.org/blogs/nunogodinho> (Inglês), INETA Country Leader por Portugal, e parte da equipa de gestão de por diversas comunidades Portuguesas como PontoNetPT, XAMLPT e Fundador da AzurePT (Windows Azure em Português).

A PROGRAMAR

Managed Extensibility Framework (MEF)

Neste artigo vou expor a framework oficial de extensibilidade de aplicações para a plataforma .NET, que existe desde a versão 3.5 mas que foi oficialmente lançada com a plataforma .NET a partir da versão 4. Esta framework vem preencher uma lacuna na plataforma para a construção de software extensível, sem as típicas dores de cabeça que esta problema tem trazido até agora. Embora já existessem outras abordagens ao problema como Unity, IoC Containers, Castle Project, a Microsoft percebeu que nenhuma delas era leve e fácil o suficiente para o efeito, e como tal, decidiram lançar e incorporar na plataforma a sua própria framework. A MEF é o culminar de aprendizagem com todos estes sistemas, ao perceber onde erravam e ao preencher os espaços que estavam por preencher para que todos possamos desenvolver sistemas extensível com o mínimo esforço possível.



Composable Part - É algo que tem tanto consome serviços de outras Parts como fornece também serviços. As Parts podem tanto vir de dentro da mesma aplicação como de um componente externo. Export - É um serviço que uma Part fornece. Quem importar esta Part vai obter a/as funcionalidades inerentes. Uma Part pode exportar mais do que um serviço se assim for necessário. Exemplo: Uma Part poderia por exemplo exportar um sistema de logging transversal a uma organização, e qualquer aplicação feita

no âmbito dessa organização não teria de se preocupar em desenvolver um sistema de logging, apenas importaria o já existente através de MEF.Import - É um serviço que uma Part consome. Uma Part pode importar um ou vários serviços tanto quanto necessário. Composition - É o acto de satisfazer as importações e exportações definidas na aplicação, connect the dots.Contracts - Definem e identificam o que está a ser exportado e o que é para ser importado.A MEF disponibiliza out-of-the-box um modelo de desenvolvimento por atributos em que o modo como se define o que é importado, o que é exportado e os metadados destas importações e exportações, é através destes.

Este é o método de utilização que vamos falar neste artigo, mas o core da MEF é completamente agnóstico, poderia-se usar qualquer outro modelo. Para entrarmos realmente na utilização de MEF e num cenário real, vamos supor que temos uma aplicação e que esta necessita de um sistema de cache. Este sistema tem de ser externo porque temos de conseguir trocar de sistema de cache de um modo transparente, ou seja, a cache é um ponto de extensibilidade da nossa aplicação. Em primeiro lugar (não sendo obrigatório para a utilização de MEF, mas como boa prática em modelos desacoplados) vamos definir um interface que é o contrato da nossa extensão:

```
public interface IPersist
{
    void SaveToCache(string key, string
payload, List<string> dependentKeys);

    string GetFromCache(string key);
}
```

A partir deste interface, podemos criar um projecto separado da nossa aplicação e apenas referenciar uma assembly (tipicamente temos um projecto com todos os

contratos, que tanto é referenciado pela aplicação como por quem cria componentes importados pela aplicação, é o contrato conhecido por ambas as partes que define os termos das extensões).Ao fazê-lo, implementamos essa assembly da forma que esse sistema de cache funcionar:

```
public class MemoryCache : IPersist
{
    public void SaveToCache(string key,
string payload, List<string> dependentKeys)
    {
        //lógica
    }
    public string GetFromCache(string key)
    {
        //lógica
    }
}

public class DiskCache : IPersist
{
    public void SaveToCache(string key,
string payload, List<string> dependentKeys)
    {
        //lógica
    }
    public string GetFromCache(string key)
    {
        //lógica
    }
}
```

Deste modo temos duas classes que podem servir como o nosso sistema de cache, a MemoryCache e a DiskCache, que terão as suas lógicas próprias que para o âmbito do artigo não são relevantes, mas mostram-nos que podemos, ao implementar o contrato (interface) definido, criar várias implementações diferentes para o mesmo problema, e que com o MEF, como vou mostrar de seguida, podemos

utilizar estes sistemas de um modo simples, e o mais importante, totalmente desacoplado:

```
public partial class App : Application
{
    private IPersist cacheSystem { get;
set; }

    public App()
    {
        //falta aqui algo, como carregamos o
sistema de cache desejado ?

        cacheSystem.SaveToCache("key",
"payload", new List<string>());

        cacheSystem.GetFromCache("key");
    }
}
```

O que temos de alterar para que o MEF se encarregue de carregar a assembly que contém o sistema de cache que queremos utilizar? (qualquer um que implemente a interface IPersist).Vamos assumir que a assembly que tem a nossa lógica de cache se encontra na directoria da nossa aplicação (poderia estar noutra qualquer directoria, ou até poderíamos definir a classe dentro da nossa própria aplicação, não existe qualquer obrigatoriedade de ser uma assembly externa).Em primeiro lugar temos de especificar que as classes que contêm a lógica de cache se vão "exportar", e isto é apenas feito decorando as classes com o atributo:[Export(typeof(IPersist))]

O que este atributo nos diz é que a classe está a disponibilizar-se para ser importada, e o tipo com que ela está a expor é o IPersist, e isto torna-se importante especialmente quando estamos a exportar e importar várias funcionalidades em simultâneo, pois facilitam a identificação.Na nossa aplicação, vamos ter de instanciar um catálogo, e os tipos de catálogos que existem são:- AssemblyCatalog: procura numa assembly- DirectoryCatalog: procura numa directoria (relativa ou

A PROGRAMAR

Managed Extensibility Framework (MEF)

absoluta)- AggregateCatalog: agrega vários catálogos- TypeCatalog: procura num tipo- DeploymentCatalog (exclusivo Silverlight): carrega XAPs dinamicamente- Custom: podemos definir o nosso próprio catálogo Neste caso utilizamos um DirectoryCatalog, visto que estamos a obter as assemblies que se estão a disponibilizar para importação na directoria da nossa aplicação. Após instanciar o catálogo temos de instanciar um CompositionContainer, fornecendo o nosso catalogo como parâmetro. O CompositionContainer é responsável por aceder ao catálogo fornecido e satisfazer os imports e exports:

```
public partial class App : Application
{
    [Import]
    private IPersist cacheSystem { get;
set; }

    public App()
    {
        var catalog = new
DirectoryCatalog(Environment.CurrentDirecto
ry);
        var container = new
CompositionContainer(catalog);
        container.ComposeParts(this);
    }
}
```

```
cacheSystem.SaveToCache("key",
"payload", new List<string>());

cacheSystem.GetFromCache("key");
}
}
```

Estas poucas linhas de código contêm um poder extraordinário se pensarmos no que seria necessário. Para implementar um sistema destes manualmente. Podemos inclusive alterar o sistema de cache sem qualquer recompilação, apenas trocando a assembly.

Não é necessária qualquer referência às assemblies que implementam o sistema de cache, tudo é desacoplado. O único ponto em comum é o contrato, que é o elo de ligação, um interface: a aplicação não tem qualquer noção sobre o funcionamento do sistema de cache, nem o sistema tem qualquer necessidade de saber o que se passa na aplicação. O nível de abstracção é quase total. Este artigo tem como objectivo demonstrar o poder da Managed Extensibility Framework, conceitos, casos de uso e a facilidade com que podemos tirar partido dela. Stay tuned!

AUTOR



Escrito por **Ricardo Rodrigues**

É técnico Nível III em Informática/Gestão pela Fundação Escola Profissional de Setúbal, tendo ingressado após na FCT da Universidade Nova de Lisboa.

Posteriormente frequentou vários cursos da Microsoft em diversas áreas como Windows Forms, ASP.NET, Securing .NET Applications, WCF, WWF, Web Services e COM+ tendo obtido as certificações MCP .NET 2.0, MCAD .NET 1.1, MCSD .NET 1.1, MCPD Windows, Web e Distributed Applications e MCPD - Enterprise Applications Developer. Contribui activamente em comunidades como StackOverflow e também possui um blog/twitter como temática relacionada: [Blog](#) / [@ricmrodrigues](#)

Microsoft BizTalk Server aos olhos dos programadores

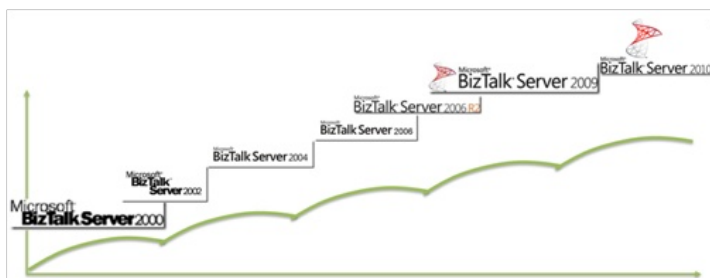
Já muito se tem falado sobre a plataforma BizTalk Server, o que é, e as vantagens que oferece as organizações. Para os mais distraídos, BizTalk Server é a plataforma de excelência da Microsoft para a integração de sistemas e processos empresariais.

Mas quais os benefícios que esta plataforma oferece aos programadores?

Mercado de trabalho

Onde o produto é utilizado? Em que contextos? Quais as oportunidades de trabalho que oferece? Remunerações? Estas são algumas perguntas que um programador quer saber antes de se dedicar a uma tecnologia/produto.

Microsoft BizTalk Server 2010 é a sétima versão do produto, tornando-o num dos produtos mais maduros e estáveis da Microsoft.



Ao longo dos anos o produto tornou-se uma referência de sucesso entre os sistemas de integração, sendo actualmente o produto mais utilizado no mundo na sua área, como asseguram os seguintes factos:

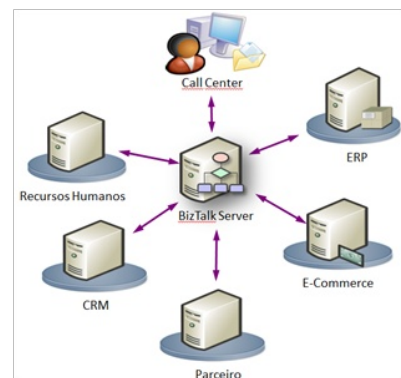
- Mais de 10 000 clientes espalhados por todo o mundo
- 81% do "Top 100" da revista Fortune
- 12 dos 15 maiores retalhistas mundiais utilizam BizTalk
- 5 das 10 maiores cadeias de hotéis mundiais utilizam BizTalk
- 9 das 10 maiores seguradoras mundiais utilizam BizTalk
- 23 dos 27 governos membros da UE utilizam BizTalk para

prestação de serviços do governo

- 9 das 10 maiores companhias de telecomunicações norte americanas utilizam BizTalk
- 6 das 8 maiores companhias farmacêuticas norte americanas utilizam BizTalk
- 9 das 10 maiores companhias espaciais e de defesa nos EUA utilizam BizTalk

Com uma robusta infra-estrutura de mensagens, funcionalidade de resiliência (Dehydration e Rehydration), mais de 25 adaptadores multiplataforma, motor de regras, possibilidade de obter informações de desempenho sobre processos críticos de negócios, debug, persistência, tratamento e recuperação de erros, suporte a transacções.

Torna o BizTalk Server numa ferramenta e infra-estrutura única, ideal para ser usado principalmente para integração de aplicações corporativas (EAI), integração de sistemas entre parceiros de negócio (B2B) e para gestão de processos de negócio (BPM).



No que diz respeito às oportunidades de trabalho, podemos definir 3 perfis:

BizTalk Architect: conhecedor de todo o sistema de integração, backup e planos de recuperação, segurança, logging, o fluxo de mensagens, interface de comunicação. Utilizando os seus conhecimentos também como um programador, o arquitecto deverá conhecer as capacidades e limitações das ferramentas à sua disposição (BizTalk,

A PROGRAMAR

Microsoft BizTalk Server aos olhos dos programadores

BAM, BRE) e desenhar os projectos fazendo as melhores escolhas para cada situação.

BizTalk Developer: implementa e estende as funcionalidades base, tirando partido das diferentes ferramentas. Aqui existem muitas áreas completamente ortogonais e um programador poderá não dominar ao mesmo nível todas elas: Orchestration, Adapters, Pipelines, Mappings, Functoids, Routing, Rules, Tracking, OLAP, entre outras.

BizTalk Administrator: um administrador de sistemas terá outras preocupações, como sejam a saúde dos servidores e a sua actividade (HAT), desbloqueando mensagens e processos, garantindo o devido fluxo das mensagens, bem como o fluxo de dados de telemetria necessários ao bom diagnóstico dos processos de negócio.

Apesar do nosso mercado interno ser muito diferente do internacional aqui estão alguns exemplos reais de oportunidades de trabalho (fonte LinkedIn):

- “BizTalk Developers needed in”: Miami, NYC, Jacksonville, Austin, Sacramento, Louisville, Hawaii, Panama City, Montevideo, London, Toronto, Preston, Porto.
- “BizTalk Administrator needed in Ft. Lauderdale FL 75K - 100K depending on experience.”
- “BizTalk Developer/Architect needed for International Law Firm in NY. 110k-130k.”
- “6 Month BizTalk 2009 Consultant required for Public Sector project in London. Rates negotiable but c£500-£550 per day.”
- “2 BizTalk Developer Needed in NYC - Healthcare, mid-level \$95-110K and senior level \$105-120K.”

Funcionalidades para os programadores

Não há nada que o BizTalk faça que com código escrito de raiz não se consiga fazer, a questão é quanto tempo demora a implementar uma solução sem um uso de middleware (sistema de integração). Facilmente se conseguirá efectuar uma integração entre dois sistemas com o uso de tecnologias de comunicação e transporte de dados tais como o Microsoft WCF, mas quando se analisa

com mais atenção alguns requisitos, é fácil identificar certos desafios que se podem tornar bastante complexos: implementação de persistência, correlação de mensagens assíncronas, recuperação em caso de falhas nas comunicações, mapeamentos complexos, atomicidade, transacções de longa duração, monitorização e visibilidade dos processos, e tudo isto se reflecte, para um programador, em custos enormes na implementação.

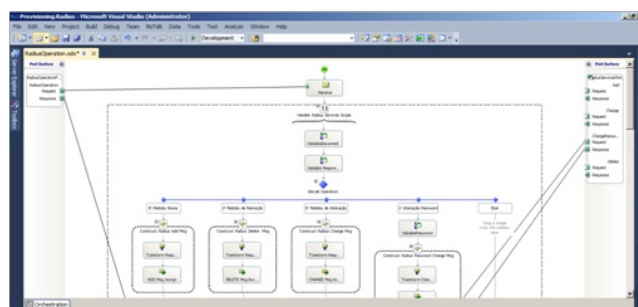
É aqui que o BizTalk entra e o que ele faz de melhor, porque todas estas funcionalidades estão ao dispor dos programadores “out of the box” com o produto, libertando assim, os mesmos, da necessidade de terem de reimplementar vezes sem conta estas funcionalidades e permitindo assim um foco no aspecto mais crítico que é a implementação da lógica de negócio associado aos fluxos de integração.

São diversas as funcionalidades ou módulos que poderíamos enumerar. Um ajudam a simplificar a interoperabilidade, outras a reduzir custos na implementação:

BizTalk Orchestration Designer

Antes de aprender a programar é ensinado a todos os programadores como representar todos os passos necessários para a execução de um processo na forma de um fluxograma, isto porque é mais legível e ilustra de forma simplificada o progresso da execução.

O BizTalk inclui um Orchestration Designer, integrado no Visual Studio, que possibilita aos programadores representarem os processo de negócio de forma visual, assim como uma representação das portas, configurações e das ligações entre as diversas actividades (shapes), tornando-se assim mais fácil de gerir e ler do que numa linguagem textual generalista (exemplo C#).

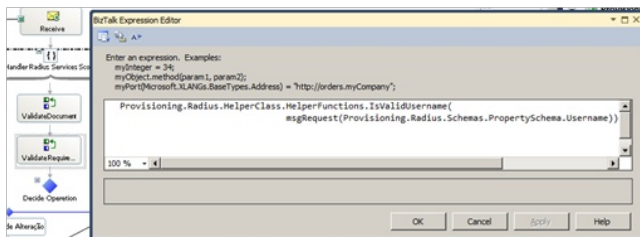


A PROGRAMAR

Microsoft BizTalk Server aos olhos dos programadores

BizTalk Expression Editor

O BizTalk Expression Editor permite aos programadores introduzir código .NET, com suporte a intellisense, directamente nas orquestrações ou invocar bibliotecas externas, o que em certos cenários se torna bastante útil, como por exemplo: manipular valores das mensagens a partir da Message Assignment shape, manipular variáveis na Expression shape, construção de expressões booleanas dentro de Loop e Decide shapes, definir tempo de pausa na Delay shape, ou até mesmo configurar portas dinâmicas de saída.

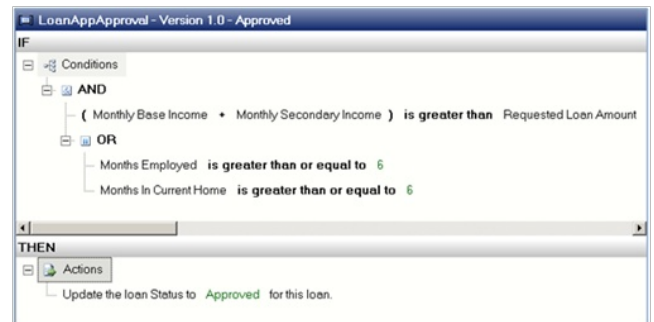


Motor de regras (BRE)

O BizTalk Server inclui o Business Rules Framework onde podem ser isolados critérios de decisão de negócio. Sendo que os módulos principais incluem o Business Rule Composer para construção de políticas (regras), o Rule Engine Deployment Wizard para as instalar no Run-Time Business Rule Engine.

A tarefa da criação e alteração das regras poderá não ser uma tarefa do programador. Os analistas, consultores ou responsáveis de negócio poderão criá-las e a qualquer momento actualizar as mesmas. O programador poderá então reutilizar as regras de negócio nas suas orquestrações para suportar uma variedade de cenários como por exemplo para determinar o caminho de execução de um processo de negócio, ou um valor a aplicar numa transacção.

Mais que um parâmetro aplicativo, este motor permite que o valor ou condição seja completamente dinâmico e apenas determinado em contexto de runtime.

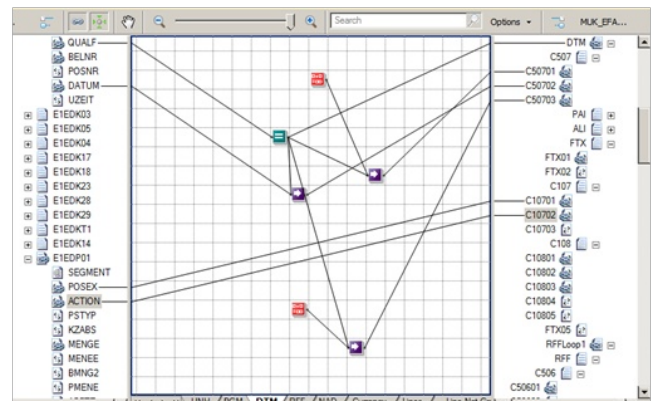


Usar regras, que mudam constantemente, em vez de código poderá permitir aos programadores não terem de refazer sistematicamente as suas aplicações.

Para mais informação: [The Business Rules Framework](#)

BizTalk Mapper Designer

Inclui o BizTalk Mapper Designer, integrado no Visual Studio, que possibilita efectuar transformações de mensagens complexas de forma visual e extremamente simples.



Na realidade, este editor está a gerar um mapa XSLT, e em determinadas situações poderá ser mais adequado usar snippets XSLT directamente no mapa. Estas opções dependem muito da experiência do programador. Em termos de estruturação/modularidade, estes mapas podem ainda ser enriquecidos com funções (Functoids), chamadas externas (SQL lookups, outros mapas) ou código XSLT, .NET/C#, COM, VBScript.

Transacções, Excepções e Persistência de dados

É comum nas aplicações Service-Oriented Architecture (SOA) ou Business Process, onde os processos podem abranger vários endpoints (sistemas), as operações não

A PROGRAMAR

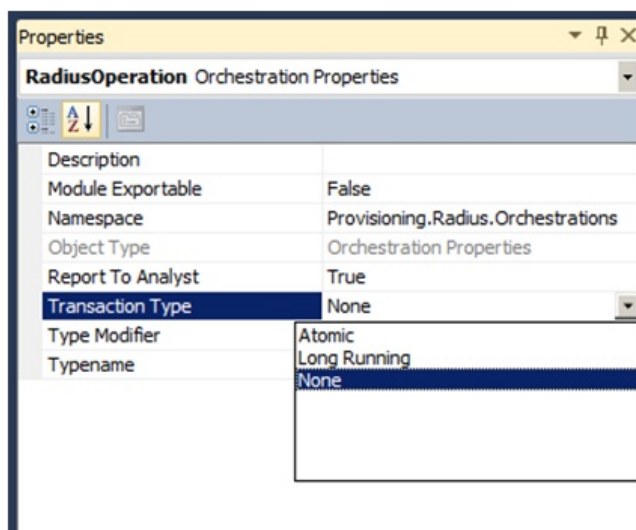
Microsoft BizTalk Server aos olhos dos programadores

poderem ser realizados num curto período de tempo, o que origina a que as transacções ACID não sejam as mais adequadas para este tipo de cenários.

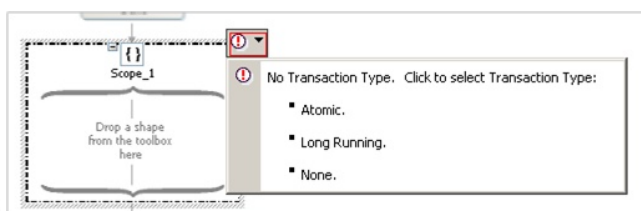
Ao contrário da programação tradicional, o BizTalk Server suporta dois tipos distintos de transacções:

- **Atómicas:** permite que uma transacção volte automaticamente para o seu estado anterior em caso de uma operação não seja concluída com sucesso. Podemos desenhar orquestrações com suporte ACID (Atomic, Consistent, Isolated e Durable) configurando a Scope (ou mesmo ao nível da orquestração) como atómica.
- **Longa duração:** Estes processos podem ficar activos por dias, semanas ou por períodos de tempo mais longos, podem conter várias transacções (nested transactions), e possibilita tratar excepções para recuperação de falha. Suporta consistência e durabilidade.

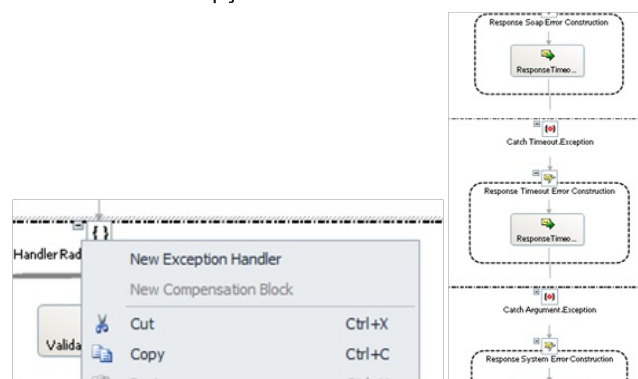
Entre várias opções, pode-se configurar o tipo de transacção ao nível da orquestração:



Assim como definir a transacção ao nível do “scope” por forma a encapsular uma unidade de trabalho dentro de um contexto de transacção:



O programador também tem a sua disposição a possibilidade de definir vários blocos (Handlers) para tratamento de excepções:



Quando lidamos com processos de negócio, nomeadamente os de longa duração, necessitamos sempre de falar de persistência dos dados por forma a salvaguardarmos de falhas e podermos efectuar o reprocessamento a partir de um determinado ponto do processo. A Orquestration engine persiste, automaticamente, o estado das instâncias em execução baseado no desenho das orquestrações efectuada pelo programador, existindo alguns eventos ou etapas em que é despoletado a operação de persistência, chamados pontos de persistência:

- Ao nível do BizTalk Engine: quando as instancias das orquestrações são suspensas, quando o sistema é desligado de forma controlada, quando a engine determina hidratar/desidratar os processos ou quando uma instância da orquestração é concluída.
- Ao nível da Orquestração: quando o final de uma scope transaccional é alcançado, na execução de outras orquestrações através da Start Orchestration shape, na Send Shape ou em breakpoints de debugging.

A persistência do estado das instâncias inclui: o progresso actual da instância, o estado de qualquer componente .NET que contém informação do estado e está a ser utilizado na orquestração assim como os valores das mensagens e variáveis.

Para mais informação:

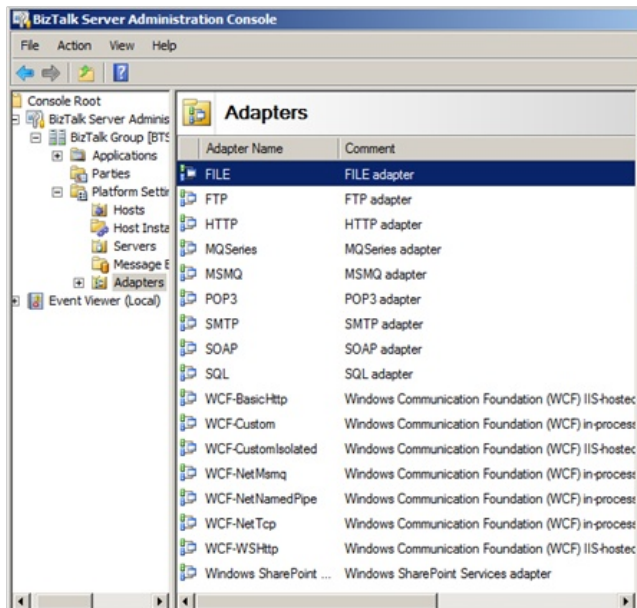
[BizTalk Orchestration – Understanding Persistence points](#)

A PROGRAMAR

Microsoft BizTalk Server aos olhos dos programadores

Adaptadores

Inclui mais de [25 adaptadores](#) que simplificam a integração directa para as top “Line of Business (LOB) Applications” (tais como o Siebel, SAP, JD Edwards, Oracle ou Dynamics CRM), Base de dados (Microsoft SQL Server, Oracle, DB2) e outras tecnologias (Tibco, Java EE).



WCF LOB Adapter SDK

Fornece um modelo de programação enriquecido para desenvolvimento de adaptadores baseados em Windows Communication Foundation. ([WCF LOB Adapter SDK](#)).

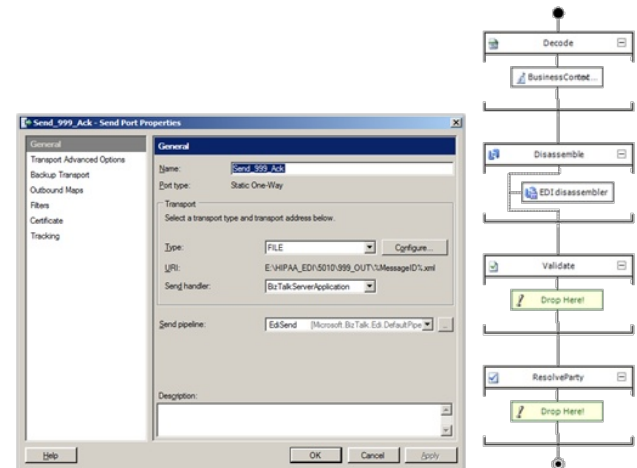
Suporte poderoso e completo para EDI/AS2

Electronic Data Interchange (EDI) é uma das formas mais comuns de comunicação electrónica entre organizações (Facturas, encomendas, notas de débito). Estes padrões (EDIFACT, ASC X12) são projectados para leitura electrónica, e portanto inadequados para leitura humana, reflectindo à troca estruturada de dados de negócios entre sistemas utilizando um formato de dados padronizado que garante a fiabilidade dos dados através de diferentes checksums.

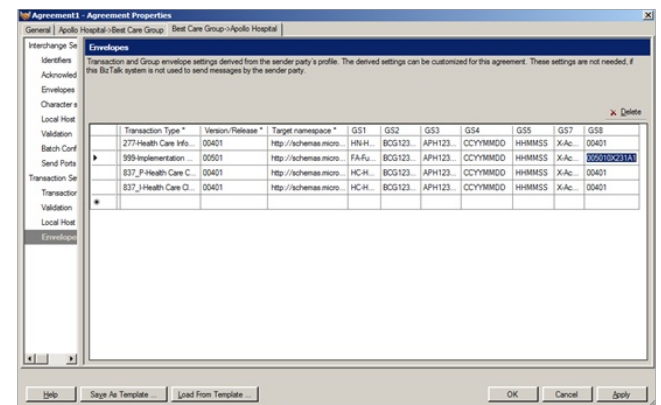
- As aplicação BizTalk EDI contém pipelines, orquestrações e schemas que são úteis e indispensáveis para a capacidade de processar transacções de EDI.
- A pipeline de entrada EDI é capaz de dividir os lotes EDI

(EDI Batch), analisar os documentos, converter os arquivos EDI em arquivos XML e efectuar validações EDI e/ou XSD.

- A pipeline de envio EDI permite converter arquivos XML em documentos X12 ou EDIFACT, codificar as mensagens e efectuar validações XSD e/ou EDI.



- A Trading Partner Management Interface permite que uma organização defina as propriedades de processamento para os parceiros comerciais fazendo intercâmbio EDI.



Para mais informações:

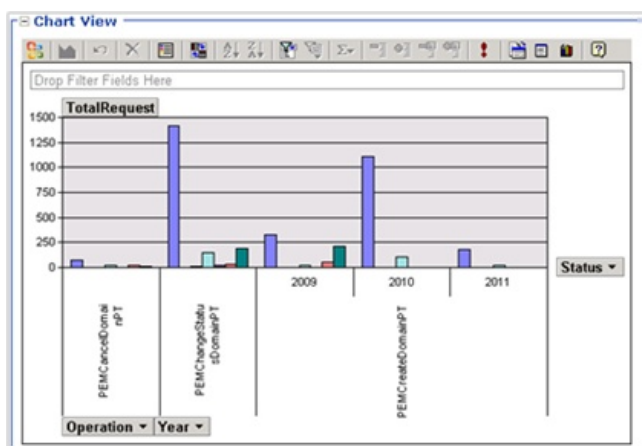
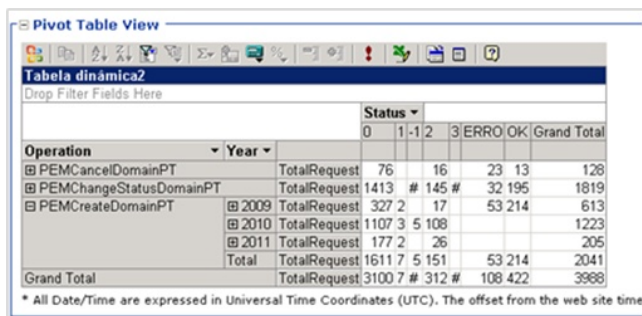
- [EDI Support in BizTalk Server](#)
- [EDI Support in BizTalk Server 2000, ..., 2006, 2009, and 2010](#)

Monitorização de processos de negócio (BAM)

O Business Activity Monitoring ou BAM é um módulo que captura dados de negócio e milestones do processo permitindo que os analistas de negócios monitorizem e analisem os dados em tempo real.

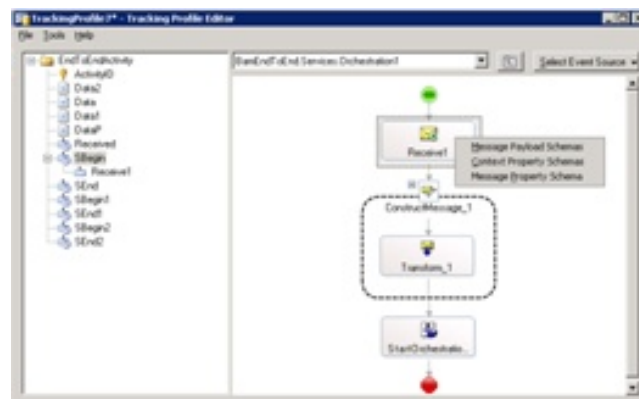
A PROGRAMAR

Microsoft BizTalk Server aos olhos dos programadores



Estas ferramentas permitem libertar os programadores da preocupação, na altura do desenvolvimento, de qual a informação que pretendem recolher e da implementação da monitorização, uma vez que os analistas podem desenvolver o seu modelo com uma ferramenta extremamente familiar (Microsoft Excel) e o administrador poderá posteriormente ligar o modelo com os processos utilizando a ferramenta Tracking Profile Editor.

Operacao	CodigoErro	Destino_1	Destino_2	Destino_3	Total Geral
Operacao_1	CodigoErro_1	3			3
Operacao_1	CodigoErro_2		2		2
Operacao_1	CodigoErro_3	1			1
Operacao_1	Total	4	2		6
Operacao_2	CodigoErro_1	4	1		5
Operacao_2	CodigoErro_2	5	3	2	10
Operacao_2	CodigoErro_3	1	1	1	3
Operacao_2	Total	10	5	3	18
Operacao_3	CodigoErro_1	1	1		2
Operacao_3	CodigoErro_2		1	1	2
Operacao_3	CodigoErro_3	3			3
Operacao_3	Total	4	2	1	7
Total Geral		19	11	6	36



O grande benefício do BAM é que proporciona a visibilidade sobre os processos de negócios. Este pequeno grande efeito, poderá ser o ideal para envolver os responsáveis de negócio na operação activa dos processos criando um feedback loop importantíssimo para a melhoria contínua dos projectos de automação (BPA).

Uma vez que a implementação do BAM é muitas vezes apenas a definição do modelo e toda a implementação é automatizada (geradores), o esforço para incluir o BAM é muito simplificado para os programadores.

Mais informações: [Business Activity Monitoring](#)

BizTalk WCF Service Publishing Wizard e BizTalk WCF Service Consuming Wizard

Mesmo quando for preciso expor um processo como um Web Service, existem wizards que nos ajudam a criar tanto os projectos WCF, como na instalação destes no servidor Web (IIS). O mesmo acontece no caso de quisermos incluir nos nossos processos, chamadas a outros Web Services.

Estas ferramentas permitem aos programadores abstraírem-se de programação repetitiva, agilizando o processo de integração.

BizTalk Flat File Schema Wizard

Uma dos padrões mais antigos para a troca de mensagens é a utilização de arquivos texto (Flat Files) como CSV ou TXT, muitos deles customizados à medida para os sistemas. Porém com a adopção do XML como formato de eleição na troca de mensagens, muitas vezes é necessário transformar arquivos texto em XML e vice-versa.

A PROGRAMAR

O Editor de texto VIM

Introdução

O editor de texto Vim, é uma aplicação open source disponível para os vários sistemas operativos Unix-based, existindo também alguns “ports” para Windows, como o caso do cygwin, bem como diversas versões gráficas. Para quem não conhece, o Cygwin é uma espécie de “ambiente unix” e linha de comandos para o sistema operativo da Microsoft, no entanto não é possível através deste “ambiente”, correr aplicações criadas nativamente para Linux, existindo uma necessidade de as recompilar.

O seu conjunto alargado de funcionalidades, como o syntax-highlight, fazem com que este seja um editor de texto útil para programadores, daí a razão deste artigo.

Com este artigo, pretendo fazer um “tour” pelo Vim, mais propriamente pela versão 7.3, não vou abordar todas as funcionalidades do mesmo, pois se o fizesse, talvez desse origem a um livro. O objectivo deste artigo é que no fim do mesmo fiquem a saber os conceitos básicos para desfrutarem do editor.

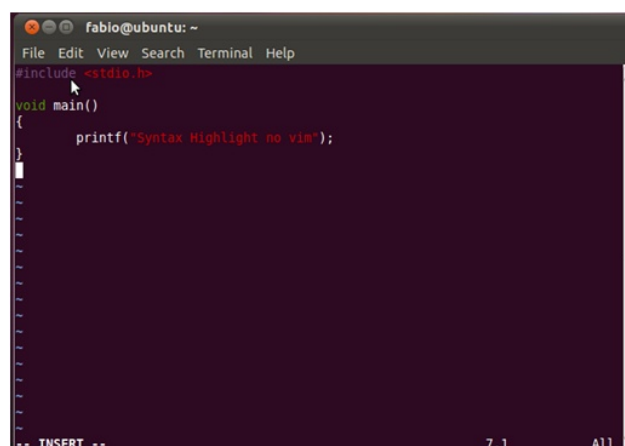
Instalar o Vim

Em muitas distribuições de Linux e no Mac OS X, o Vim já vem instalado, no entanto certas distribuições como o Linux Ubuntu não o trazem instalado, e a instalação deverá ser feita abrindo uma janela do terminal e escrevendo o seguinte comando:

```
$sudo apt-get install vim
```

Poderá também utilizar o Software Center desta distribuição.

Se utiliza Windows e não desejar instalar o Linux nem que seja recorrendo a uma máquina virtual, poderá recorrer ao cygwin ou a uma versão gráfica deste editor como a que se encontra no site <http://www.vim.org>.



Utilização do Syntax-Highlight

Uma das principais funcionalidades para programadores do vim é o syntax highlight, que permite distinguir o código mais facilmente, e consequentemente ser um excelente auxílio ao programador, como podemos ver na imagem seguinte:

Esta funcionalidade, poderá ser usada abrindo um ficheiro de código, com a extensão apropriada, por exemplo, um ficheiro com o nome programa.c, irá ser aberto com o syntax highlight para c.

Como abrir

Se utilizar um “port” para Windows, poderá abri-lo como qualquer outro programa (menu iniciar ou acedendo ao executável do mesmo), se utiliza cygwin, Linux ou Mac OS X, deverá executá-lo através do terminal, existindo duas formas de o abrir:

Se quisermos abrir um ficheiro já existente escrevemos o seguinte comando no terminal:

```
$vim <directoria e nome do ficheiro>
```

Se quisermos especificar um nome para um novo ficheiro

A PROGRAMAR

O Editor de texto VIM

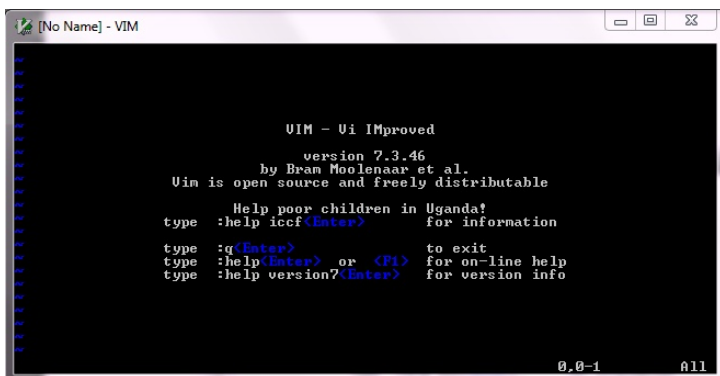
escrevemos o seguinte comando no terminal:

```
$vim <nome do ficheiro>, caso esteja na mesma
directoria onde o quer criar/abrir.
```

Para abrir o editor de texto vim com um ficheiro sem nome, podemos utilizar o seguinte comando no terminal:

```
$vim
```

Iremos então ver o ecrã inicial do vim:



Se nunca trabalhou com este editor de texto, provavelmente você está neste momento a interrogar-se “como é que eu crio um novo documento nisto?”, “como é que eu começo a escrever?”. O motivo de não conseguir escrever nada, é porque você não se encontra no modo que possibilita escrever, ou seja o modo insert, mas afinal o que são os modos do vim? Vamos ver na próxima secção.

Os modos do vim

O vim possui 3 modos de edição, o modo Insert, o modo Command e o Last Line Mode (modo última linha), vamos ver quais as funções de cada um deles:

Insert – Permite inserir texto, sendo por isso o modo básico do editor de texto, e possibilita escrever.

Command – Permite inserir comandos, que permitem executar as várias funcionalidades do editor, como guardar e formatar o texto, fazer tipos de deslocamento específicos, etc. Para introduzir estes comandos, usamos as teclas do teclado, por exemplo, para entrar no modo insert, usamos a tecla i, que é a tecla que deverá premir para começar a

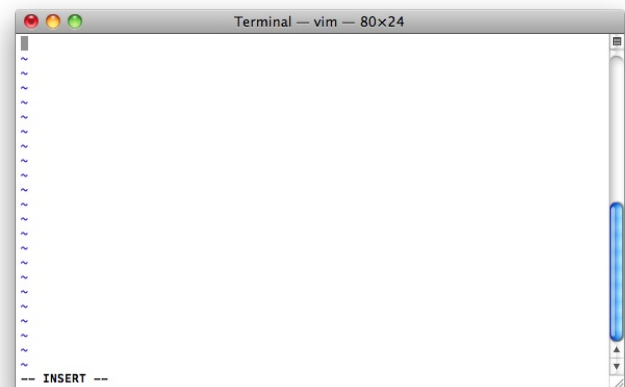
escrever quando abre o vim.

Para passar do comand mode para o insert mode, premimos, tal como já foi referido a tecla i, para fazer o contrario (insert mode para command mode), usa-se a tecla esc.

Last Line Mode – É um submodo do modo Command, para aceder ao mesmo, prime-se a tecla : no modo Command.

O Modo Insert

Quando nos encontramos no modo insert, irá aparecer essa indicação no rodapé do terminal, tal como podemos ver na imagem seguinte:



Este modo, é bastante fácil de usar, as teclas das setas movem o cursor, e poderá escrever livremente usando as teclas. Para guardar o documento, é necessário recorrer ao modo command, para isso, e tal como anteriormente referido, prime-se a tecla esc.

O Modo Command

Comandos para Mover o Cursor

O cursor pode fazer movimentos simples e múltiplos neste modo, uma das características do Vim é o facto de poder usar teclas do teclado principal para além das do cursor. Este sistema tem muitos fãs, uma vez que possibilita mover o cursor sem ser necessário tirar os dedos do centro do

Tecla	Movimento
h	Move 1 espaço para a esquerda
j	Move 1 espaço para baixo
k	Move 1 espaço para cima
l	Move 1 espaço par a direita

A PROGRAMAR

O Editor de texto VIM

teclado. Na tabela abaixo, podemos ver uma listagem das teclas utilizadas para os movimentos singulares:

Tecla	Movimento
\$	Coloca o cursor no fim da linha actual
0 (zero)	Coloca o cursor no início da linha actual
b	Coloca o cursor para o início da palavra anterior
w	Avança o cursor para o início da próxima palavra

É também possível efectuar movimentos múltiplos através destas teclas de movimentos simples precedendo a tecla com o número de vezes que pretendemos carregar, por exemplo, se quisermos avançar quatro caracteres para a direita, em vez de premirmos 4 vezes a tecla "l", podemos escrever "4l". Existem também mais algumas formas úteis de movimentar o cursor:

Tecla	Função
ZZ	Sai e grava o documento
:e!	Elimina todas as alterações feitas ao ficheiro e volta ao estado original do mesmo
:q!	Elimina todas as alterações feitas ao ficheiro e fecha o editor
:q	Fecha o vim, no entanto se existirem alterações não guardadas, o vim recusa-se a encerrar
:w <ficheiro>	Grava o ficheiro actual com o nome especificado em ficheiro, se quiser guardar o ficheiro "teste", deverá escrever ":w teste", caso não especifique nenhum nome, o vim grava no ficheiro que está a editar.

Gravar o documento

Para gravar o documento, é necessário também recorrer ao modo de comando, para isso utilizam-se os seguintes comandos:

Tecla	Função
A	Inserir texto no documento, a diferença entre o "a" e o "i", é que o "a" insere o texto depois do cursor e o "i" antes.
s	Elimina o carácter onde o cursor se encontra, e permite substituí-lo.
S	Elimina uma linha inteira
R	Faz com que o vi entre no modo replace, fazendo com que os caracteres que premirmos substituam os existentes; noutras palavras, à medida que vamos escrevendo novo texto, vai substituindo o existente "passando por cima".
dw	Apaga a palavra em que o cursor se encontra
dd ou D	Apaga a linha em que o cursor se encontra
X	Apaga o carácter sobre o qual o cursor se encontra
~	Altera a capitulação de uma letra (passa de maiúsculas para minúsculas e vice-versa)
C	Ver abaixo

Se tentar gravar um ficheiro com o mesmo nome de um já existente, poderá usar o comando :w! para fazer overwrite, ou seja substituir o ficheiro mais antigo pelo novo

Edições Simples

Quando inserimos texto no ficheiro, este raramente está perfeito, muitas vezes encontramos erros. É por este motivo que o Vim, contém um conjunto de ferramentas que nos permitem fazer algumas edições simples ao documento.

Na tabela seguinte, podemos ver alguns dos comandos mais utilizados para editar o documento:

Tecla	Função
A	Inserir texto no documento, a diferença entre o "a" e o "i", é que o "a" insere o texto depois do cursor e o "i" antes.
s	Elimina o carácter onde o cursor se encontra, e permite substituí-lo.
S	Elimina uma linha inteira
R	Faz com que o vi entre no modo replace, fazendo com que os caracteres que premirmos substituam os existentes; noutras palavras, à medida que vamos escrevendo novo texto, vai substituindo o existente "passando por cima".
dw	Apaga a palavra em que o cursor se encontra
dd ou D	Apaga a linha em que o cursor se encontra
X	Apaga o carácter sobre o qual o cursor se encontra
~	Altera a capitulação de uma letra (passa de maiúsculas para minúsculas e vice-versa)
C	Ver abaixo

A tecla "c" tem como função substituir texto e tem a vantagem de podermos dizer quanto texto queremos alterar, combinando-a com teclas de movimento. Desta forma, eis algumas das possibilidades de utilização desta tecla para que melhor fiquem a perceber a sua utilização:

Tecla	Função
cw	Elimina a palavra onde o cursor se encontra (note-se que a tecla w serve para avançar uma palavra)
c2w	Elimina as duas palavras que estão a seguir ao cursor (note-se que a tecla 2w permite avançar 2 palavras)
c\$	Elimina toda a linha onde o cursor se encontra (note-se que a tecla \$ permite avançar até ao fim da linha actual)

Com estes três exemplos, em princípio deverá compreender como funciona a tecla "c". Vejam as teclas de movimento anteriormente referidas e abuse desta funcionalidade.

Mover Texto

Uma vez que o Vim, guarda temporariamente num buffer o texto eliminado, é possível restaurá-lo premindo a tecla “p”. Por isso, o procedimento para mover texto neste editor é apagar o que se pretende mover, e restaurar no local desejado com a tecla “p”. No entanto tenha em atenção que após qualquer outra edição, o texto é limpo, por isso restaure sempre antes de qualquer outra edição.

Copiar Texto

Para copiar texto no vim, usa-se a tecla “y” seguida de uma tecla de movimento, ou da tecla “y” para copiar a linha em que o cursor se encontra. Para colar, usa-se a tecla “p”. Para obter mais informações, acerca de alguns exemplos de teclas de movimento que poderá usar, consulte as tabelas referentes aos movimentos.

Repetir ou anular o comando anterior

Se desejar fazer várias vezes o mesmo comando, coloque o cursor no local onde o deseja correr e prima a tecla ponto (.). Irá aplicar novamente o comando anterior.

Caso tenha cometido um erro de edição, recorra à tecla “u”, para anular a última acção.

Deslocamentos no ecrã

Existem também algumas sequências de teclas que podemos digitar no modo command e que permitem

deslocar (fazer scroll) ao ecrã:

Tecla	Função
Ctrl + F	Avança um ecrã para a frente
Ctrl + B	Avança um ecrã para trás
Ctrl + D	Avança meio ecrã para a frente
Ctrl + U	Avança meio ecrã para cima

Conclusão

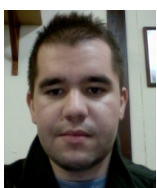
Termina assim, este tour pelo vim. Agora que já conhece os conceitos básicos deste editor e se estiverem interessado em perceber mais poderá encontrar diversos sites que explicam as funcionalidades avançadas deste editor.

Para mim, este é um excelente editor, o facto de não dar uso ao rato, faz com que tenha como público-alvo os mais puristas do teclado, no entanto, e apesar deste argumento, isso não significa que as pessoas que “gostam” do rato, não o possam usar, pois este editor é bastante simples de utilizar, e a grande quantidade de comandos disponíveis, tornam mais rápidas certas operações.

Se precisar de ajuda, o vim possui uma série de documentos, bastando para aceder aos mesmos, entrar no Command Mode, e digitar o comando :help.

Um bom sítio para começar é a cheat-sheet disponível em <http://www.tuxfiles.org/linuxhelp/vimcheat.html>, onde poderá consultar outros comandos úteis.

AUTOR



Escrito por **Fábio Domingos**

Estudante de Gestão de Sistemas de Informação na Escola Superior de Ciências Empresariais do Instituto Politécnico de Setúbal, tem como principal hobbie a informática, nomeadamente as áreas de programação, bases de dados e IT Support. É moderador global do forum Portugal-a-Programar, membro do Staff desta revista e por vezes contribui com soluções para a comunidade Experts-Exchange.

Blog pessoal - [@fdomingos](#)



ppp
portugal-a-programar



facebook.com/portugal.programar



twitter.com/pt_programar

COLUNAS

CORE DUMP - Fazer mal = Rápido?

VISUAL (NOT) BASIC - Introdução ao OpenXML SDK

Fazer mal = Rápido?

Em conversa recente com um programador sobre uma determinada opção que não me parecia a melhor, este respondeu-me que “teve de ser assim porque não havia tempo”.

Esta justificação, ou fatalidade, é comum no meio da programação, sendo impossível a um profissional desta área não se ter cruzado com ela. Todos nós, mesmo os que não são profissionais desta área, já olharam para o seu trabalho e o criticaram. Isso é excelente dado que indica que sabemos fazer melhor. Então porque não o fazemos? Só porque havia pouco tempo?

Ouvi tantas vezes esta explicação, ou desculpa, que tal me levou a parar e pensar um pouco sobre a mesma.

Sem dúvida que os prazos têm poderes mágicos sobre os projectos. **Quando um dead-line se aproxima começa a magia: os testes passam para segundo plano, a documentação resume-se a pouco mais do que código fonte e o código fonte começa a ser escrito de forma menos ortodoxa...** E é precisamente neste último ponto que reside a origem da justificação de código mal feito: a tal falta de tempo...

Seguindo um raciocínio lógico, corroborado ao longo de anos a fio pela expressão “falta de tempo => mau código”, qualquer um poderá acreditar que tal é verdade. Para provar se esta expressão é verdadeira ou falsa comecei a perguntar a todos quantos quantos usam este dogma, a seguinte questão: **“Dizes que fizeste mal porque era mais rápido. Isso quer dizer que se fizesses bem demorarias mais tempo?”**

Silêncio pensativo...

Resposta inconclusiva...

A verdade é que a larga maioria das pessoas aceita esta expressão como um dogma. “Para fazer mais rápido tenho de fazer mal” parece uma fatalidade, qual fado de quem ganha a vida a programar. No entanto, quando confrontadas com o facto de que demorariam mais tempo se fizessem bem, nenhuma, até ao dia de hoje, me respondeu que sim, que necessitaria de mais tempo para fazer as coisas de forma correcta.

A minha teoria, obviamente não provada cientificamente, é de que **com a pressão, a maioria das pessoas entra em stress e baixa os seus próprios padrões de qualidade, aceitando como médio um trabalho que rejeitariam em condições normais.** Em casos mais críticos, esta baixa de padrões é acompanhada por trashing mental, onde as pessoas deixam de conseguir raciocinar de forma clara, afectando assim o resultado final do seu trabalho.

Sejam quais forem as razões na verdade “fazer mal” não implica que se faça “mais rápido”. Obviamente haverá excepções, mas mesmo sobre a pressão de um dead line devemos ser capazes de manter a cabeça fria e ser fieis aos nossos próprios padrões de forma a não comprometer o resultado final do nosso trabalho. Numa área onde a pressão de dead lines agressivos é demasiado recorrente, não ceder e saber manter os seus padrões é um factor valiosos que diferencia os bons profissionais.

No entanto, e infelizmente, continuo a ouvir esta expressão demasiadas vezes...

AUTOR



Escrito por **Fernando Martins**

Faz parte da geração que se iniciou nos ZX Spectrum 48K. Tem um Mestrado em Informática e mais de uma década de experiência profissional nas áreas de Tecnologias e Sistemas de Informação. Criou a sua própria consultora sendo a sua especialidade a migração de dados.

VISUAL (NOT) BASIC

Introdução ao OpenXML SDK

Resumidamente, o OpenXML é um padrão aberto (ISO) de arquivos para documentos, folhas de cálculo e apresentações, que é baseado em XML e que pode ser implementado por diversas aplicações em diversas plataformas.

A sua principal função é poder alterar e visualizar o conteúdo dos documentos através de diversas aplicações, desenvolvidas em diferentes plataformas, garantindo assim a fidelidade da informação sem qualquer perda de dados.

Como é a sua estrutura

Um arquivo no formato OpenXML é um arquivo ZIP que contém uma estrutura de pastas, arquivos XML e mais qualquer outro objecto que esteja presente no documento. Dai o formato ocupar substancialmente menos do que as versões mais antigas (*.doc, *.xml, *.ppt)

Se renomearmos, por exemplo, um ficheiro *.xlsx para *.zip podemos ver a sua estrutura e os ficheiros XML que compõem este documento. Usando o Open XML SDK 2.0 Productivity Tool permite-nos também ver a estrutura hierarquicamente e o seu conteúdo. Neste exemplo podemos ver que o Book1.xlsx tem três Worksheets (Sheet1, Sheet2 e Sheet3) e que os textos das worksheets são guardados no ficheiro SharedString.xml.

Uma grande vantagem desta estrutura é a possibilidade de aceder a uma determinada parte do arquivo sem que seja necessária a sua leitura total, o que leva a ganhos de tempo no que toca a arquivos de grande dimensão.

“Uma grande vantagem desta estrutura é a possibilidade de aceder a uma determinada parte do arquivo sem que seja necessária a sua leitura total”

Este SDK também oferece suporte a programação através de LINQ para XML, o que torna a programação com XML muito mais fácil do que o tradicional modelo de programação.

O que pode ser feito com OpenXML

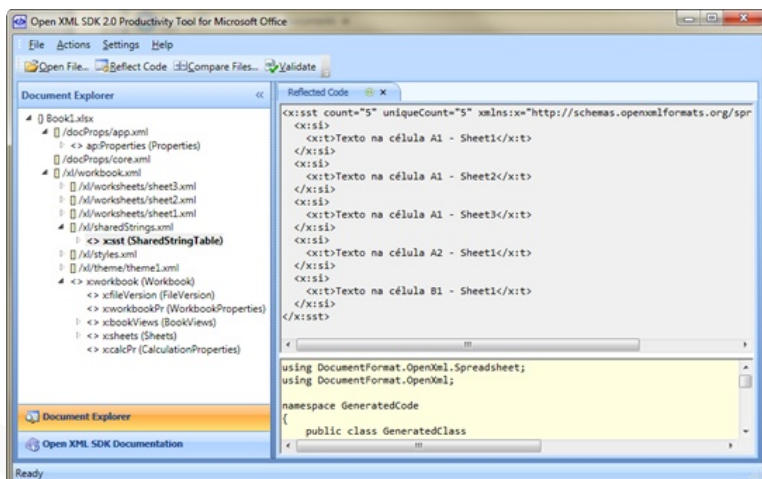
O OpenXML permite, como já foi referido anteriormente, a manipulação e criação de documentos, folhas de cálculo e apresentações, para tal existem as seguintes subcategorias:

WordProcessingML - Manipulação de documentos de texto (Word)

SpreadSheetML - Manipulação de folhas de cálculo (Excel)

PresentationML - Manipulação de apresentações (PowerPoint)

Para manipular facilmente este tipo de ficheiros, existe o Open XML SDK que é gratuito e de fácil/rápida instalação.

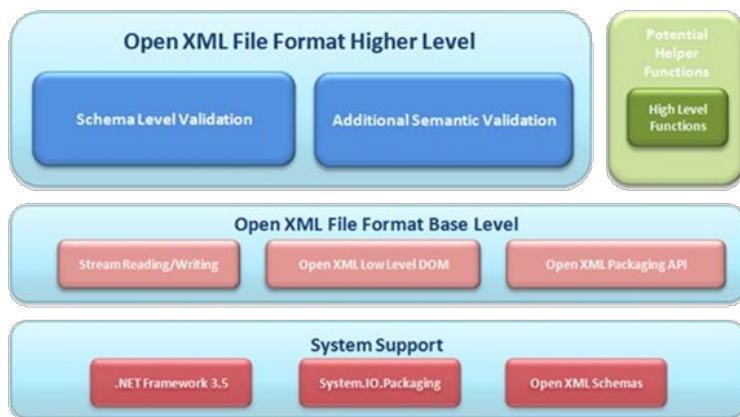


VISUAL (NOT) BASIC

Introdução ao OpenXML SDK

Arquitectura

O Open XML SDK é implementado camadas, a partir de uma camada de base que vai subindo a uma de maior nível de funcionalidade. O diagrama abaixo mostra o esquema do Open XML SDK.



System Support

.NET Framework 3.5: Simplifica a manipulação dos ficheiros XML mais simples através de Linq to XML

System.IO.Packaging: Adiciona, edita e remove os ficheiros XML contidos no pacote

Open XML Schemas: Estes esquemas são a base do Open XML SDK que actualmente é baseado no padrão ECMA-376



Open XML File Format Base Level

Open XML Packaging API: Este componente é construído sobre o componente System.IO.Packaging. Em vez de retornar partes do documento, retorna classes de tipos e objectos.

Open XML Low-Level DOM: A grande vantagem deste componente é que se pode ver facilmente as propriedades contidas numa determinada classe através do IntelliSense

Stream Reading/Writing: O seu funcionamento é semelhante ao funcionamento das classes XmlReader e XmlWriter, mas são mais fáceis de usar



Open XML File Format

Schema Level Validation: Esta camada facilita a depuração e validação de documentos Open XML

Additional Semantic Validation: Fornece informações adicionais com base em restrições e sintaxe



Potencial Helper Functions

High Level Functions: É um auxiliar com exemplos de código para executar operações comuns. Estas funções têm como função modificar o XML



Vantagens

A principal vantagem de usar o Open XML SDK é que é totalmente suportado no servidor, ao contrário dos aplicativos do Microsoft Office. Não necessita de ter o Microsoft Office instalado, para criar ficheiros neste formato, e além disso, há uma enorme vantagem de desempenho no desenvolvimento com o Open XML quando se trata de um grande número de documentos. Podem ser criados milhares de documentos em segundos!

VISUAL (NOT) BASIC

Introdução ao OpenXML SDK

O Open XML SDK é um formato especializado na manipulação e criação de pacotes de Open XML.

O SDK é totalmente compatível com a estrutura e esquema do Open XML Formats.

“A principal vantagem de usar o Open XML SDK é que é totalmente suportado no servidor ... não necessita de ter o Microsoft Office instalado”

Limitações

Não é um substituto para o Microsoft Office e além disso, é necessário compreender a estrutura dos formatos para usar o Open XML SDK.

Não é possível converter formatos Open XML para outros formatos, como HTML ou XPS. Não garante a validade do documento quando se opta por manipular directamente o XML. Não possui um layout ou funcionalidades de recalculo

Utilização

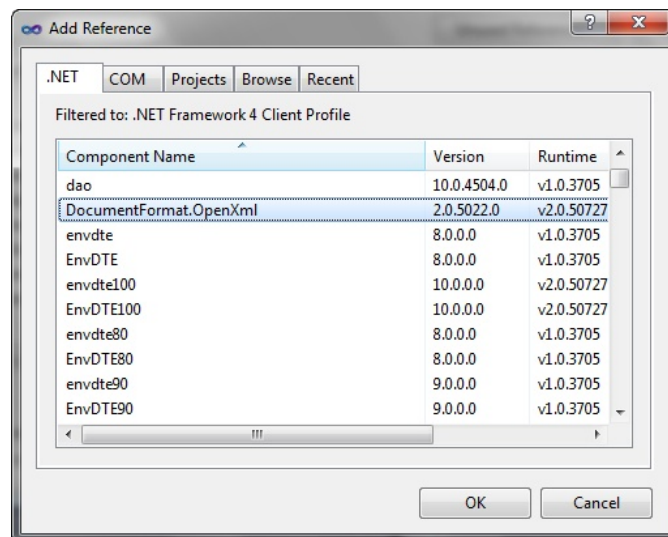
O SDK pode ser obtido através desta hiperligação: <http://bit.ly/jkaykK>

Necessário: “OpenXMLSDKv2.msi” – É aqui que está a DLL que vamos precisar

Opcional: “OpenXMLSDKTool.msi” – É uma espécie de explorador do arquivo (Open XML SDK 2.0 Productivity Tool), onde se pode visualizar o seu esquema (árvore) de construção, documentação, etc.. Também dispõe de um gerador de código, mas apenas para C#.

Depois de instalado, terá de ser referenciado. Para isso

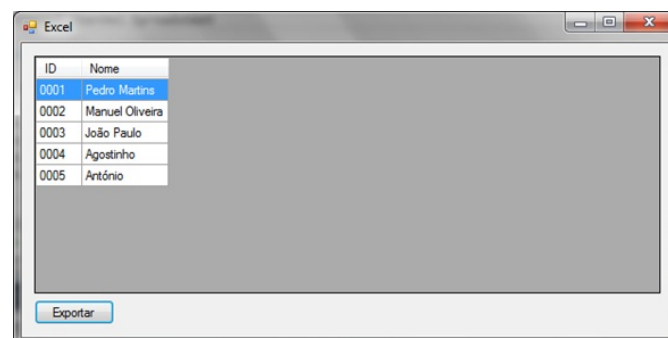
escolham a referência “DocumentFormat.OpenXML”, no separador .NET



Exemplos de utilização

Exemplo 1

Este exemplo mostra como efectuar uma simples exportação do conteúdo de uma DataGridView para uma folha de Excel, usando um Template



Criamos um template:

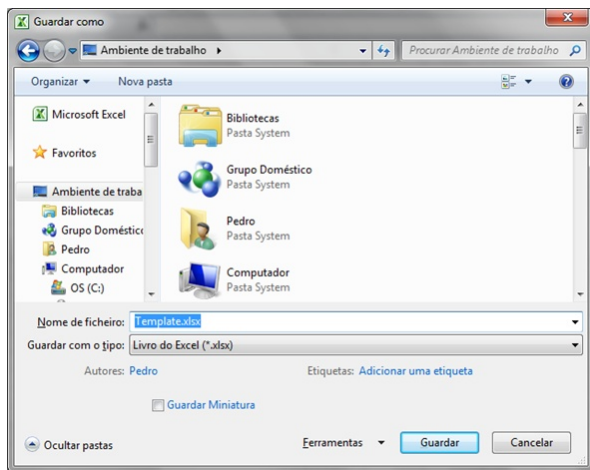
	A	B	C	D
1	ID	Nome		
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Agora guardamos o nosso template com o nome

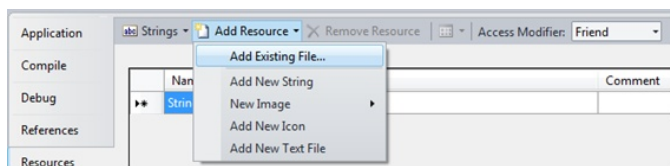
VISUAL (NOT) BASIC

Introdução ao OpenXML SDK

“template.xlsx”



Depois de termos o template criado, vamos adicionar aos resources da aplicação



Depois do template pronto vamos passar ao código.

Primeiro devem-se importar os namespaces necessários

```
Imports DocumentFormat.OpenXml
Imports DocumentFormat.OpenXml.Packaging
Imports DocumentFormat.OpenXml.Spreadsheet
```

Código do botão exportar:

```
Private Sub BtnExportar_Click(...) Handles
BtnExportar.Click

    'Escreve o template numa pasta temporária
    Dim ficheiro As String =

My.Computer.FileSystem.SpecialDirectories.Temp
p & "\\ListaClientes.xlsx"

My.Computer.FileSystem.WriteAllBytes(,
```

```
My.Resources.Template, False)
```

```
'Abre o template criado
Using Doc As SpreadsheetDocument =
SpreadsheetDocument.Open(ficheiro, True)
```

```
'Abre o workbookPart
Dim Parte As WorkbookPart =
Doc.WorkbookPart
Dim WSheet As WorksheetPart =
Parte.WorksheetParts.First()
```

```
'Acede aos dados nela contidos
Dim Dados As SheetData =
WSheet.Worksheet.GetFirstChild(Of
SheetData)()
```

```
'Algumas variaveis que vamos precisar
' no ciclo que se segue
Dim R As Row = Nothing
Dim C As Cell = Nothing
```

```
'Percorre todas as linhas da Grid
For Each Linha As DataGridViewRow In
Dgv.Rows
```

```
'Cria uma nova linha
R = New Row
R.RowIndex =
CType(Linha.Index + 2, UInt32Value)
```

```
'Percorre as colunas da Grid
For Each Coluna As DataGridViewColumn
In Dgv.Columns
```

```
C = New Cell
C.CellReference =
Coluna.Name & Linha.Index
C.DataType = CellValues.String()
```

```
'Adiciona o valor à coluna
C.CellValue =
New CellValue(Linha.Cells
(Coluna.Index).Value.ToString())
```

VISUAL (NOT) BASIC

Introdução ao OpenXML SDK

```
'Adiciona a coluna à linha
R.AppendChild(C)

Next

'Adiciona a linha à folha
Dados.AppendChild(R)

Next

'Guarda o documento actual e fecha-o
Parte.Workbook.Save()
Doc.Close()
End Using

'Depois de tudo feito, abre-se a folha
Process.Start(ficheiro)
End Sub
```

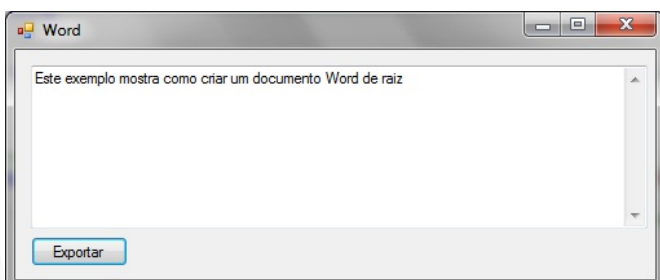
E este é o resultado:

	A	B
1	ID	Nome
2	0001	Pedro Martins
3	0002	Manuel Oliveira
4	0003	João Paulo
5	0004	Agostinho
6	0005	António
7		
8		

Como podemos ver, esta operação foi efectuada em pouco tempo, com simplicidade e rapidez.

Exemplo 2

Este exemplo mostra como criar uma folha de Word de raiz e colocar lá algum texto



Importação dos namespaces necessários

```
Imports DocumentFormat.OpenXml
Imports DocumentFormat.OpenXml.Packaging
Imports DocumentFormat.OpenXml.Wordprocessing
```

Código do botão exportar

```
'Cria um novo documento
Dim ficheiro As String =
    "C:\ExportacaoWord.docx"

Using Doc As WordprocessingDocument =
    WordprocessingDocument.Create(ficheiro,
        WordprocessingDocumentType.Document)

    'Cria uma nova parte do documento
    Dim Parte As MainDocumentPart =
        Doc.AddMainDocumentPart

    'Cria uma nova estrutura
    Parte.Document = New Document

    'Cria o corpo do documento
    Dim Corpo As Body =
        Parte.Document.AppendChild(New Body())

    'Adiciona um paragrafo
    Dim Paragrafo As Paragraph =
        Corpo.AppendChild(New Paragraph())

    'Escreve um texto
    Dim Texto As Run =
        Paragrafo.AppendChild(New Run())
    Texto.AppendChild(New Text(TxtTexto.Text))

    'Fecha o documento
    Doc.Close()
End Using

'Abre o documento
Process.Start(ficheiro)
```

VISUAL (NOT) BASIC

Introdução ao OpenXML SDK

E este é o resultado:

Este exemplo mostra como criar um documento Word de raiz

Exemplo 3

Este exemplo mostra como ler parágrafos de uma folha de Word. Importação dos namespaces necessários

```
Imports DocumentFormat.OpenXml
Imports DocumentFormat.OpenXml.Packaging
Imports DocumentFormat.OpenXml.Wordprocessing
```

Lê o texto e escreve na consola

```
'Cria um novo documento
Dim ficheiro As String =
    "C:\ExportacaoWord.docx"

'Abre o documento anteriormente criado
Using Doc As WordprocessingDocument =
    WordprocessingDocument.Open(ficheiro, False)

'Percorre a colecção de parágrafos
For Each Paragrafo In
    Doc.MainDocumentPart.Document.Body
        Console.WriteLine(Paragrafo.InnerText)
Next
```

```
'Fecha o documento
Doc.Close()

End Using
```

Conclusão

Neste artigo foi feita uma pequena abordagem ao OpenXML SDK 2.0 Productivity Tool, onde foi possível ver a sua arquitectura, o que é possível fazer e quais são as suas vantagens e desvantagens. Resta dizer que é uma excelente alternativa à programação directa nos objectos do Microsoft Office.

Algumas Referências

Welcome to the Open XML SDK 2.0 for Microsoft Office
<http://bit.ly/84hC3>

Office 2010 Sample: Open XML SDK 2.0 Code Snippets for Visual Studio 2010
<http://bit.ly/azN4Pw>

OpenXML Developer
<http://openxmldeveloper.org>

Open XML Explained Ebook
<http://openxmldeveloper.org/archive/2007/08/13/1970.aspx>

AUTOR



Escrito por Pedro Martins

É técnico Nível III em Informática/Gestão pela Escola Profissional Cisave

Exerce funções de programador numa multinacional sediada em Portugal onde trabalha com várias tecnologias da Microsoft .NET. É moderador do quadro Visual Basic.NET e ASP.NET na comunidade Portugal@Programar e também faz parte do staff da WikiTeam

COMUNIDADES

NetPonto - Certificações Microsoft

Certificações Microsoft

Neste artigo pretendo abordar o que é a certificação Microsoft e clarificar o leitor sobre o percurso a percorrer para obter a certificação e quais os benefícios.

Decerto que os mais interessados neste tema, já se devem ter deparado com as seguintes dúvidas:

- O que é a certificação?
- Quais os benefícios?
- Como começar?
- Que exame tenho que realizar?
- Onde devo realizar os exames?
- Quanto custa?
- Como são os exames?
- Como estudar?
- Quanto tempo demora obter?
- Que títulos posso obter?



A Certificação Microsoft não é mais do que formação profissional, que não exige que um aluno tenha curso superior para se tornar um profissional qualificado para desenvolver e dar suporte às tecnologias Microsoft. É o método mais rápido e reconhecido para obter conhecimento sobre os produtos e tecnologias Microsoft que estão em uso ou que acabaram de ser lançados no mercado. A certificação é atribuída de acordo com a aprovação nos exames oficiais da Microsoft, desta forma um aluno que obtenha aprovação em um exame, torna-se Microsoft Certified Professional (MCP) e é emitido por parte da Microsoft, um documento comprovativo da sua capacidade e qualificação, garantindo desta forma a diferenciação no mercado.

A Microsoft oferece uma série de certificações que qualificam o profissional de acordo com a sua área de tecnologia.

“Os profissionais com Certificação Microsoft destacam-se no panorama laboral das TI”



Os **principais benefícios** que se destacam com a obtenção da certificação são:

- Reconhecimento Profissional
- Produtividade
- Certificação Internacional
- Informação Privilegiada
- Eventos
- Descontos

O vasto leque de produtos e tecnologias Microsoft origina que estejam disponíveis muitos exames, o melhor será filtrar pela área para a qual se pretende obter certificação. As áreas disponíveis são:

- Server technologies
- Developer tools and applications
- Microsoft Dynamics
- Windows
- Microsoft Office
- Security
- Other solutions

Escolhida a área em que o leitor se enquadra ou pretende obter certificação, coloca-se a grande questão:

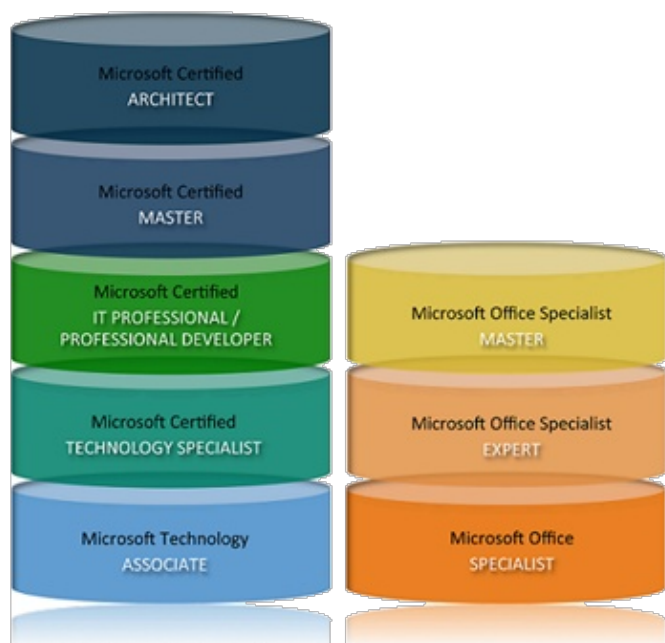
- #1 Como começar?
- #2 Escolher a certificação que pretende fazer;
- #3 Escolher o primeiro exame;

COMUNIDADE NETPONTO

Certificações Microsoft

Para escolhermos a certificação a realizar, o leitor deve pensar no(s) objectivo(s) atingir. Por exemplo, caso esteja interessado em melhorar os conhecimentos do dia-a-dia, deve-se focar na certificação que esteja relacionada com a(s) tecnologia(s) ou produto(s) que usa mais. Suponhamos que no dia-a-dia desenvolve aplicações usando Silverlight, então é aconselhado a realizar o exame 70-506 - TS: Silverlight 4, Development e em caso de sucesso obtém o título de Microsoft Certified Technology Specialist (MCTS): Silverlight 4, Development.

A certificação está hierarquizada da seguinte forma:

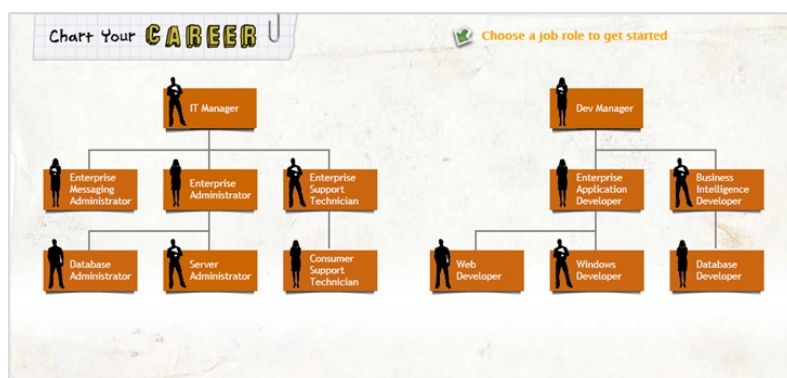


Os níveis mais comuns entre os profissionais são: Microsoft Technology Associate, Microsoft Certified Technology Specialist, Microsoft Certified IT Professional / Professional Developer, Microsoft Office Specialist. Enquanto que Microsoft Certified Master e Microsoft Certified Architect são níveis muito exigentes, com um percurso longo e para o qual é preciso requisito muito específicos e uma experiência e conhecimentos avançados, este níveis normalmente são obtidos no Estados Unidos e o custo deles é muito elevado. São poucos os profissionais que obtém estes títulos. Em Portugal temos alguns profissionais com estes títulos, no entanto não consigo quantificar.

Nota:

1. O grau dos vários níveis aumenta de baixo para cima. Por exemplo, um Microsoft Certified IT Professional terá como requisito, pelo menos, a obtenção de um título de Microsoft Certified Technology Specialist, que por sua vez implica a aprovação em pelo menos um exame.

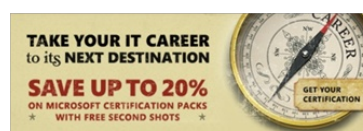
2. O seguinte "Professional Career Chart", permite ajudar a orientar no percurso profissional, consulte em <http://bit.ly/mQHUBn>



Mais á frente irão ser apresentadas as certificações e os respectivos títulos.

Os exames podem ser realizado em qualquer centro de exames Microsoft, em Portugal temos vários, como a Ciclo, a Rumos, a Galileu, a Actual Training, etc. Todos estes centros também são centros Prometric, podendo por isso usar o site da Prometric para efectuar a marcação do exame a que propõe, sendo o custo actual de cada exame 140€. Saliento que ao efectuar o registo através de um centro o valor poderá ter valor de IVA acrescido.

• Até dia 30 de Junho de 2011, poderá usufruir da possibilidade de ter um desconto de 15-25% no valor do exame e ainda a "second shots", isto é, a 2ª oportunidade para realizar o exame sem pagar mais por isso. Para mais informações consulte o seguinte link: <http://bit.ly/jSuNre>



• Para o leitor que seja estudante, chamo atenção que é

COMUNIDADE NETPONTO

Certificações Microsoft

possível até ao dia 30 de Junho de 2011, realizar uma certificação sem qualquer custo. Para mais informações consulte o seguinte link: <http://bit.ly/15N43t>



Nota: Um exame quando é marcado tem um prazo para poder ser alterado ou cancelado, no entanto é preciso apresentar uma justificação válida.

A estrutura dos exames varia de tecnologia para tecnologia ou de produto para produto, tanto em termos de tempo disponível para realizar a prova ou o tipo e número de questões. Para saber mais sobre o formato dos exames visita as seguintes referências:

- Exam Formats: <http://bit.ly/fsA9f5>
- Microsoft Certification Exam Demo - <http://bit.ly/fWoFyW>

Para além da experiência ser um factor fundamental para a realização do exame, nem sempre isso é um é um dado adquirido. Com experiência ou não é sempre bom realizar um estudo estruturado e para ajudar nesse estudo pode-se ter em conta:

- **Os objectivos de cada exame**, pois estes apresentam todo o conteúdo abordado no exame.
- **Cursos presenciais** nos centros Microsoft
- **Cursos Oficiais da Microsoft On-line:** <http://bit.ly/mRTUvj> - através do portal Microsoft e-Learning, pode fazer diversos cursos on-line com preços bastante reduzidos, comparado com os cursos presenciais.
- **Cursos do Programa RampUp:** <http://bit.ly/lk49d8> - O RampUp é uma iniciativa da Microsoft onde pode fazer alguns cursos on-line totalmente gratuitos. É ideal para quem é iniciante em um produto ou tecnologia!
- **Training Kits** da Microsoft Press - Podem ser comprados na Microsoft Press UK, Amazon, etc.)
- **Exames Simulados** - Existem várias empresas que fornecem exames simulados, para que possa praticar antes de fazer o exame real, como por exemplo a MeasureUp ou Self-Test Software
- **Prepare o seu Próprio Plano de Estudos** - No site da Microsoft Learning pode encontrar os "Learning Guides" para cada exame de certificação, contendo todos os tópicos que são abordados no exame em questão.

- Webcasts do MSDN e Comunidades
- Sessões dos Eventos da Microsoft - PDC, Mix, TechEd

À primeira vista poderá achar que existem muitos recursos, no entanto não é necessário que tenha que ter todos em conta. Associado ao ritmo de estudo, tipo de estudo, disponibilidade e maturidade, assim será definido o tempo de preparação para um exame. É preciso ter atenção quando se marca o exame, deve-se ter em conta um período de tempo suficiente para ficar bem preparado e ao mesmo tempo não ficar com intervalo muito grande para não dispersar por outras coisas.

Irei de seguida apresentar os títulos que se podem obter e os respectivos exames, de acordo com a tecnologia ou produto em causa.

Microsoft Visual Studio and Microsoft .NET Framework technologies

Percurso para obtenção do título - Microsoft Certified Technonology Specialist (MCTS)

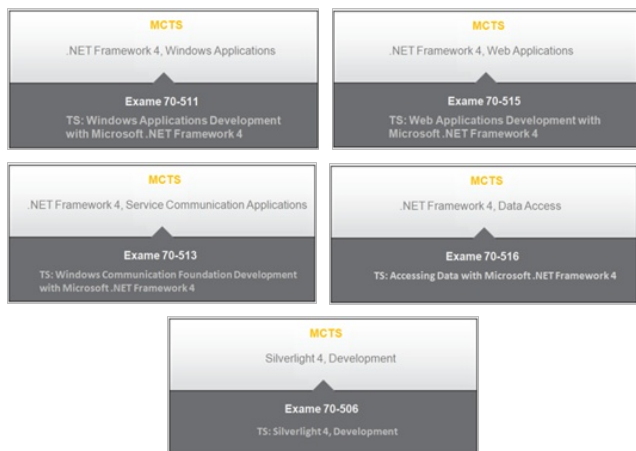
- .NET Framework 3.5 - Visual Studio 2008



- .NET Framework 4.0 – Visual Studio 2010

COMUNIDADE NETPONTO

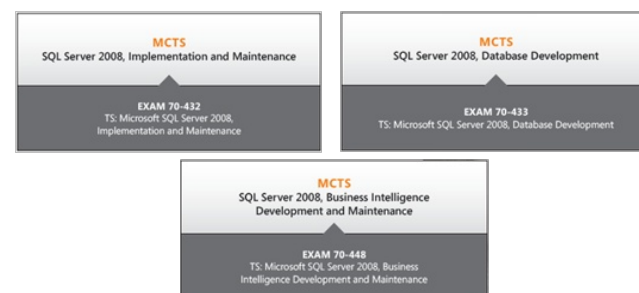
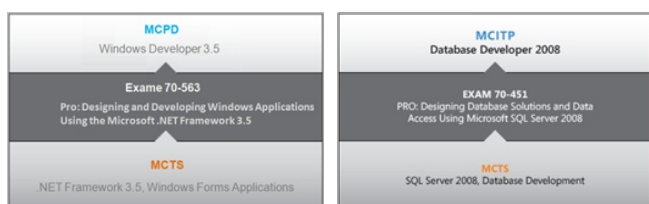
Certificações Microsoft



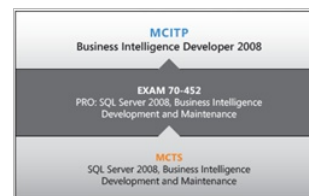
O guia de preparação para Microsoft VS and Microsoft .NET Framework está disponível em <http://bit.ly/9sqIA5>

Percurso para obtenção do título - Microsoft Certified Professional Developer (MCPD)

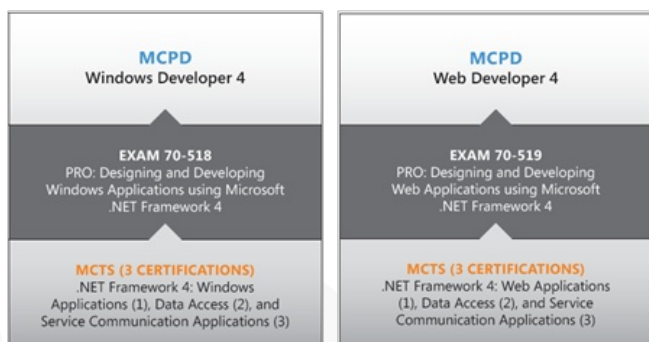
- .NET Framework 3.5 – Visual Studio 2008



Percurso para obtenção do título - Microsoft Certified IT Professional (MCITP)



- .NET Framework 4.0 – Visual Studio 2010



Nota: Para obter cada um dos títulos de MCITP é requerido um MCTS.

O guia de preparação para Microsoft SQL Server Technologies - SQL Server 2008 está disponível em <http://bit.ly/9mEOTK>

COMUNIDADE NETPONTO

Certificações Microsoft

Microsoft Office System Technologies

Percurso para obtenção do título - Microsoft Certified Technonology Specialist (MCTS)

• SharePoint 2007



• SharePoint 2010



• Project Server 2010

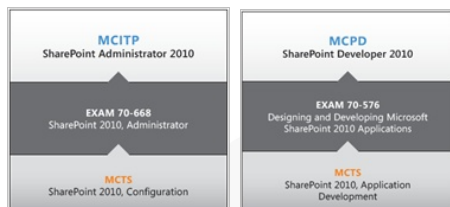


• Project Server 2007



Percurso para obtenção do título - Microsoft Certified IT Professional (MCITP)

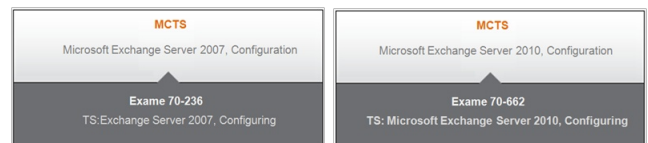
• SharePoint 2010



O guia de preparação para Microsoft Office System Technologies está disponível em <http://bit.ly/jeiqiz>

Microsoft Exchange Server Technology

Percurso para obtenção do título - Microsoft Certified Technonology Specialist (MCTS)



Percurso para obtenção do título - Microsoft Certified IT Professional (MCITP)



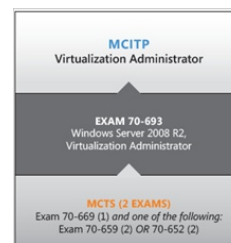
O guia de preparação para Microsoft Exchange Server Technology está disponível em <http://bit.ly/m18WvA>

Microsoft Virtualization Technology

Percurso para obtenção do título - Microsoft Certified Technonology Specialist (MCTS)



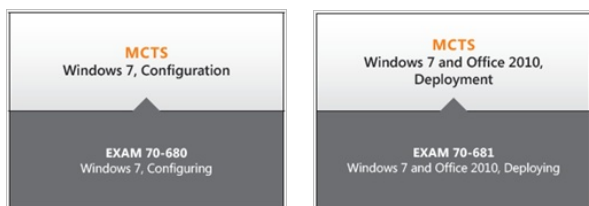
Percurso para obtenção do título - Microsoft Certified IT Professional (MCITP)



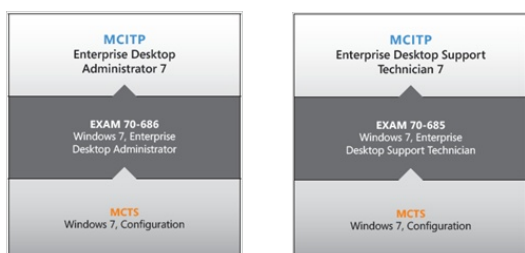
O guia de preparação para Microsoft Virtualization Technology está disponível em <http://bit.ly/m18WvA>

Windows 7

Percurso para obtenção do título - Microsoft Certified Technology Specialist (MCTS)



Percurso para obtenção do título - Microsoft Certified IT Professional (MCITP)



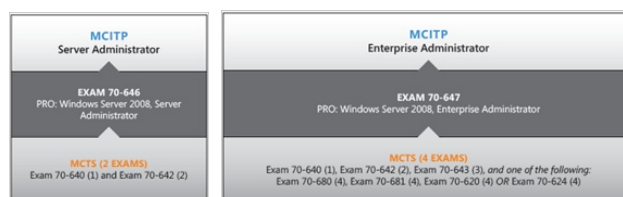
O guia de preparação para Windows 7 está disponível em <http://bit.ly/jUjHXw>

Windows Server

Percurso para obtenção do título - Microsoft Certified Technology Specialist (MCTS)



Percurso para obtenção do título - Microsoft Certified IT Professional (MCITP)



O guia de preparação para Windows Server está disponível em <http://bit.ly/hr6tZk>

Em conclusão, a certificação Microsoft é uma mais-valia no percurso profissional de um profissional de TI, dado a possibilidade ao profissional gerir todo o processo de certificação consoante a sua disponibilidade e objectivos.

Referências:

- Certificações Microsoft: <http://bit.ly/f7IN3o>
- Apresentação - Nova Geração de Certificação Microsoft – TechDays 2010: <http://bit.ly/dJQTPk>
- Microsoft Certification Exams: <http://bit.ly/ewmmbi>
- Microsoft Certification Tutorial: <http://bit.ly/hrDvYc>
- Learn about becoming an MCTS: <http://bit.ly/dlXbXA>
- Learn about becoming an MCITP: <http://bit.ly/f4AEWS>
- Learn about becoming an MCPD: <http://bit.ly/fuW70V>
- Learn about becoming an MOS: <http://bit.ly/hOfWpe>
- Explore Careers: <http://bit.ly/iVqHxN>
- Career Portal Home: <http://bit.ly/iDNDmv>
- IT Manager Portal: <http://bit.ly/mA0S4h>
- Student Career Portal Home: <http://bit.ly/iyl4fU>

AUTOR



Escrito por **Sara Silva**

É licenciada em Matemática – Especialidade em Computação, pela Universidade de Coimbra, actualmente é Software Developer no Porto.

O entusiasmo pela área resultou na obtenção dos títulos de Microsoft Certified Professional Developer – Windows 3.5, Microsoft Certified Technology Specialist – WPF 3.5, WPF 4 e Windows Forms. Faz parte de várias comunidades, tendo uma participação activa na Comunidade NetPonto e no P@P.

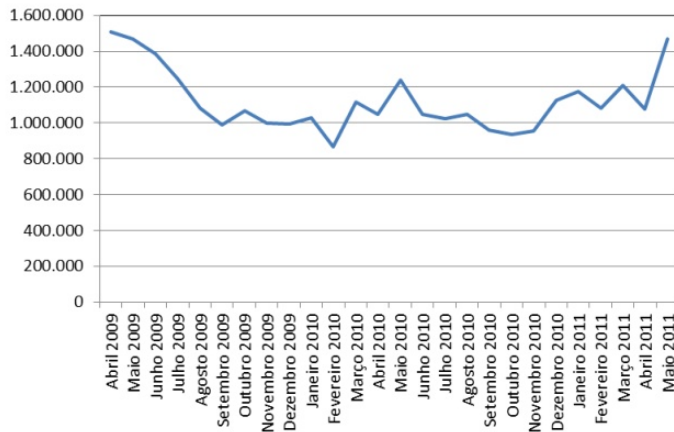
Feliz Aniversário P@P



A comunidade Portugal-a-Programar fez 6 anos de existência no dia 28 de Maio de 2011. São vários anos de trabalho e dedicação, de diversos colaboradores voluntários, para que tenha sido possível criar e fazer crescer uma comunidade de programadores na língua Portuguesa.

Aproveitamos para divulgar alguns números do P@P:

FÓRUM

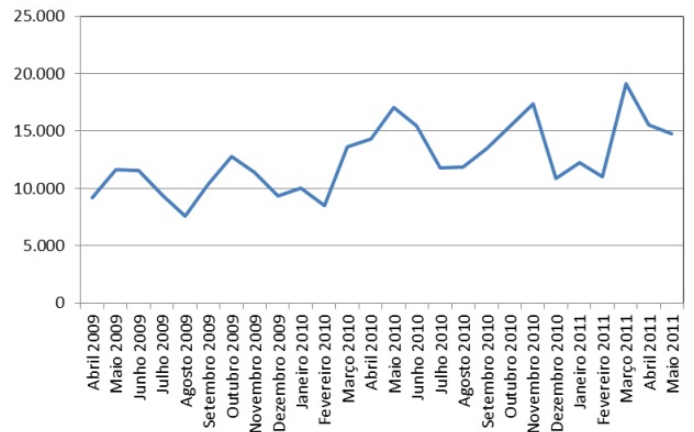


Nos últimos meses o crescimento do fórum da comunidade tem sido muito significativo, com uma média de **1.122.034 visualizações por mês**, recuperando uma quebra no primeiro semestre de 2009. São sem dúvida bons indicadores para o futuro!

REVISTA PROGRAMAR

A revista PROGRAMAR é sem dúvida um dos projectos mais importantes da comunidade, estando neste momento num dos melhores períodos de sempre. Embora as primeiras edições tenham mais downloads, por estarem disponíveis à mais tempo, as últimas edições (a partir da edição nº 25) têm tido os melhores resultados nos primeiros dois meses de lançamento e com **uma média de 14.375 downloads por edição**.

WIKI



A wiki da comunidade tem tido também um crescimento importante (média de **12.500 visualizações/mês**). No primeiros meses de 2011 foram efectuadas várias alterações e remodelações que têm sido correspondidas com mais visitas. Esperamos que possa crescer ainda mais.

Mas há mais ...

20.000 membros (23 novos membros por dia)

43.000 tópicos (28 novos tópicos por dia)

390.000 mensagens (200 novas mensagens por dia)

28 edições da revista PROGRAMAR

208 artigos na revista PROGRAMAR

110 redatores já contribuíram na revista PROGRAMAR

676 artigos/snippets na Wiki

Estes são apenas alguns dos muitos números que poderiam ser apresentados!

Numa iniciativa de divulgação do 6º aniversário, a Microsoft apoiou-nos, oferecendo t-shirts a todos os que participaram e divulgaram a iniciativa nos seus blogs e redes sociais.

Agradecemos desde já à Microsoft, pelo apoio, e a todos os que contribuem diariamente para o crescimento desta comunidade!

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?



portugal-a-programar
•org

