PROGRAMAR

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

EDIÇÃO #32 - DEZEMBRO 2011

ISSN 1647-0710

iOS, COCOA TOUCH & MVC

A PROGRAMAR

GERAÇÃO DE NÚMEROS ALEATÓRIOS PARTE 2

AUTOMATIZAÇÃO DE TAREFAS USANDO ATRIBUTOS

ARQUITECTURA WINDOWS AZURE

ENIGMAS C# ARRAYS

COLUNAS

FREE AS IN BEER CORE DUMP

UM POUCO MAIS DE WINDOWS PHONE 7 VISUAL (NOT) BASIC

COMUNIDADES

COMO SUPORTAR MÚLTIPLAS IDENTIDADES NO SEU WEBSITE USANDO WINDOWS AZURE APPFABRIC ACS AZUREPT

BIZTALK SERVER PRINCÍPIOS BÁSICOS DOS MAPAS NETPONTO

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

Website
http://www.revista-programar.info

ISSN
1 647-071 0

RIP

No inicio de Outubro faleceu Steve Jobs, mais precisamente um dia antes do lançamento da edição anterior da Revista PROGRAMAR. Sete dias depois faleceu Dennis Ritchie. Doze dias depois faleceu John McCarthy. Independentemente de se gostar ou não, a verdade é que aquilo que fizeram influenciou a vida de muitas pessoas. Dennis Ritchie foi um dos pais do **C**, linguagem que influenciou a vida de todos os programadores, quer directa quer indirectamente, quer usem ou não **C** para programar. John McCarthy foi o pai do Lisp, que apesar de não estar tão disseminado na programação, têm uma grande relevância, principalmente na área da Inteligência Artificial. Steve Jobs levou a Apple rumo ao sucesso, depois da empresa sem si ter afundado, quase a ponto de se extinguir. E é verdade que cada um fez muito mais, estes são, no entanto, e na minha opinião as marcas mais importantes de cada um deles.

Assim, não tenho qualquer receio de afirmar que todos eles merecem a nossa consideração pelo que fizeram e alcançaram, quer se concorde ou não com algumas ideias e filosofias. Mas a verdade é que o mais destacado foi sem dúvida Steve Jobs. Mereceu notícias de abertura em canais de televisão, notícias de primeira página em jornais e revistas, entre outras, enquanto Dennis Ritchie e John McCarthy ficaram quanto muito com algumas linhas em jornais e revistas, e um pouco mais em revistas e comunidades ligadas à programação e tecnologia. Muitos insurgiram-se pelo facto de não haver o mesmo destaque entre todos. Mas a verdade é que ambos viviam muito mais à sombra da sociedade em geral que Steve Jobs. Contudo não foram esquecidos. O **Fedora 16**, lançado cerca de um mês depois da morte de Dennis Ritchie foi-lhe dedicado. Durante a sua vida John McCarthy recebeu também inúmeros prémios, como por exemplo o prémio **Turing**. Também é verdade que muitos se apressaram quase a colocar Steve Jobs num pedestal, removendo-lhe os defeitos. Mas ele tinha os seus, tal como Dennis e John. Mas seria melhor pessoa, melhor "informático" que os outros dois? É provável que não. Apenas era mais visível, e o mediatismo fez o resto. Mas a verdade é que terminaram a sua existência terrena tal como a conhecíamos. Talvez sem puderem dizer um adeus a quem gostavam ou queriam, tal como Dennis Ritchie que faleceu sozinho em sua casa.

Da minha parte resta-me deixar um **obrigado** aos três pela sua contribuição para as tecnologias da informação, e um **até sempre** a todos os leitores e equipa da Revista PROGRAMAR, pois esta será a minha última edição como coordenador. Foram mais de 3 anos de participação neste projecto, passando por redactor, revisor, editor e agora coordenador. Bastante tempo, considerando que o projecto tem aproximadamente 6 anos. Mas saio com a sensação de dever cumprido, sabendo que fiz todos o que estava ao meu alcance para levar a revista mais longe, com ajuda de toda a equipa da Revista PROGRAMAR, a quem deixo também um muito obrigado. Durante o tempo que estive como editor e coordenador mudámos o visual para algo mais apelativo, criámos parcerias com comunidades, passamos a distinguir os artigos mais votados, entre várias outras mudanças internas. Tudo isto graças à excelente equipa que torna possível a existência da Revista PROGRAMAR. Saio também com a certeza de ser bem substituído pelo António Santos, aquém deixo desde já um obrigado, e desejo que com a sua a ajuda a Revista PROGRAMAR chegue ao "infinito e mais além".

Até sempre,
António Silva
<antonio.silva@revista-programar.info>

ÍNDICE

TEMA DE CAPA

A PROGRAMAR

COLUNAS

COMUNIDADES

EVENTOS

03 Dez 2011	- 4° Evento da Comunidade Azure PT
16 a 18 Dez 2011	- Windows Phone 7 App Code Camp (Palmela)
17 Dez 2011	- 25ª Reunião Presencial da Comunidade NetPonto em Lisboa
27 Fev a 1 Mar 2012	- XIX SINFO - Semana Informática IST
17 Março 2012	- SQL Pass Saturday Portugal #115

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

O que a Microsoft, Oracle, IBM e SAP Não Dizem aos Clientes

"Os quatro grandes vendedores de software - Microsoft, Oracle, IBM e SAP – têm motivos escondidos que os clientes precisam de compreender, de outra forma podem ser levados a comprar produtos e serviços que não servem as suas necessidades. Esse é o takeaway de uma palestra recente da Gartner na Austrália, relatado pela IT News.

Num simpósio esta semana, o analista da Gartner, Dennis Gaughan, explicou o que os quatro grandes vendedores estão realmente a tentar fazer, baseando-se na experiência da Gartner com os seus clientes.

A Microsoft quer principalmente proteger Windows e o Office. A Microsoft é uma empresa de plataformas e o seu maior objective é proteger os seus monopólios altamente lucrativos de Windows e Office, enquanto ao mesmo tempo estabelece outras plataformas que farão com que seja mais difícil para os clientes separarem-se delas mais tarde. A nova funcionalidade é dada a "conta-gotas" aos utilizadores dessas plataformas medulares, mas os novos produtos existem para proteger o núcleo. Ele aconselhou a que fosse tida extrema precaução antes de mudar para o Office 365, e disse que não se entre numa mentalidade "toda-Microsoft.

Os produtos da Oracle não funcionam muito bem juntos. A força de vendas da Oracle é extremamente agressiva em empurrar um conjunto de produtos, mas tem muito menos pontos de integração do que a SAP. De facto, a integração é normalmente deixada inteiramente ao cuidado do cliente. A Oracle está igualmente muito relutante em falar sobre mapas de produto por temer que produtos futuros canibalizem os existentes. A empresa obtém mais de 90% dos seus lucros através de taxas de manutenção e fará o que for necessário para manter essas taxas em entrada. Gaughan também expressou alguma surpresa por tantos clientes continuarem a trabalhar com a Oracle apesar de relatarem que a Oracle é "o vendedor com quem é mais difícil lidar."

A IBM quer apoderar-se da tua estratégia de TI. A IBM toma-se a si própria como um líder rígido, mas o seu verdadeiro negócio é vender serviços de consultoria. Parra prosperar, os gestores de conta da IBM tentam obter controlo sobre a estratégia de TI de uma empresa de forma a poderem continuar a empurrar novos produtos. Gaughan recomenda que se tome uma aproximação colaborativa ou de parceria.

A SAP confunde os clientes com o preço. Muitos dos clientes da SAP pedem a Gartner ajuda para descobrir os preços e licenciamento da SAP, uma vez que a SAP tem termos invulgares para a facturação da entrada e saída de dados nos sistemas. Gaughan também disse que uma grande transição tecnológica que estava a guiar as receitas da SAP durante os últimos anos - movimentando clientes existentes do velho sistema R/3 par o novo Business Suite - está quase terminada, o que significa que a SAP terá que ser mais agressiva com taxas de manutenção. Ele recomendou o bloqueio dos preços de manutenção agora.
De uma forma geral, Gaughan disse que a maior parte da inovação que está a ser feita por estas empresas está nos seus braços de pesquisa. O seu verdadeiro objectivo é proteger o "satus quo" o maior tempo que for possível.

Fonte: Business Insider, 19 de Novembro de 2011
Tradução: Sara Santos

Programação com Lego Vence Quinta Edição de Codebits

Uma aplicação que usa peças Lego com o objectivo de permitir que crianças consigam fazer programação informática ganhou o primeiro prémio da edição deste ano do Sapo Codebits.

O Codebits, que começou na quinta-feira e terminou este sábado, reuniu cerca de 800 participantes, que se dividiram em equipas para o habitual concurso de desenvolvimento de projectos – as equipas têm 48 horas para criar e apresentar uma ideia. Para além dos prémios (computadores, telemóveis e outros aparelhos), as melhores ideias recebem apoio do Sapo para serem desenvolvidas. O júri atribuiu o primeiro prémio a uma aplicação móvel que fotografa uma base onde é possível colocar peças Lego e que converte a disposição e cor dessas peças num pequeno programa informático.

O conceito foi trazido ao Codebits pelo programador alemão a viver em Portugal Peter Bouda. O informático Pedro Leite, um dos elementos da equipa de três pessoas que desenvolveu a aplicação, explica que a ideia permite às crianças "usarem algo tangível" para fazerem programação informática (uma tarefa que tradicionalmente envolve escrever código usando linguagens com sintaxe própria).

Na demonstração feita no Codebits, uma sequência de peças Lego foi usada para animar, no ecrã, uma pequena figura também da conhecida marca de brinquedos. As primeiras três peças (duas vermelhas e uma verde) eram usadas para dar a instrução de que a figura devia mover-se para a direita e as três peças imediatamente à frente indicavam quantas vezes esse movimento deveria ser feito. A linha seguinte dava uma nova instrução à figura (por exemplo, saltar). E a linha final pode servir para determinar se a animação pára ou se a sequência é novamente seguida.

O significado atribuído às sequências de peças depende daquilo para que a aplicação que as fotografa tiver sido concebida – uma aplicação pode interpretar três peças vermelhas como uma instrução para emitir um som e outra como uma instrução para mostrar um número.

Com este conceito, as peças Lego podem ser usadas para criar imagens animadas, fazer música, construir uma calculadora ou, teoricamente, para qualquer tipo de efeito que se consiga produzir com uma linguagem de programação.

Este processo implica que as crianças – ou alguém por elas – memorize o que significa cada sequência de peças. “As crianças são uma esponja”, afirmou Pedro Leite, mostrando-se confiante de que isto não será um obstáculo à utilização do conceito. Porém, admitiu que, uma vez desenvolvida a tecnologia, ainda é necessária “a killer app” – a expressão tipicamente usada para designar um uso de uma tecnologia que promove significativamente a sua adopção e, eventualmente, a transforma num sucesso comercial.

No último dia do Codebits, que decorreu no Pavilhão Atlântico, em Lisboa, foram mostrados cerca de 80 projectos, com cada equipa a ter 90 segundos para fazer a apresentação, num último esforço para convencer o júri (que acompanhou o trabalho das equipas ao longo dos dois dias) e ainda com o objectivo de conseguir os sete prémios atribuídos pelo público, que podia votar nos projectos no final de cada apresentação.

“O Codebits faz parte da necessidade de o Sapo ter sangue novo”, afirmou, no encerramento do evento, Abílio Martins, administrador do Sapo (que pertence ao grupo PT), referindo-se ao facto de as ideias nascidas no Codebits poderem ser integradas no Sapo ou desenvolvidas em parceria com a empresa. “Vinte destes projectos serão contactados para trabalharem connosco”, adiantou.

As 25 Piores Palavras-passe do Ano

A empresa SplashData, especialista em aplicações para smartphones incluindo gestão de palavras-passe, divulgou a lista das 25 piores passwords de 2011. Ou seja, aquelas que mais facilmente são descobertas por hackers. Esta lista da SplashData baseia-se no estudo de milhões de palavras-passe “roubadas” durante este ano e que foram posteriormente divulgadas online por hackers.

Muitos dos utilizadores usam sequências numéricas e alfabéticas carregando em teclas contíguas do teclado - como “123456” ou “qwerty” - ou então a própria palavra “password”. De acordo com o director-executivo da SplashData, Morgan Slain - citado pelo site especializado em tecnologia Mashable - “mesmo que as pessoas sejam encorajadas a escolher palavras-passe seguras e fortes, muitas continuam a escolhê-las fracas, fáceis de adivinhar, colocando-se em risco de fraude e de roubo de identidade”.

A hesitação dos utilizadores na escolha de uma password difícil poderá prender-se com o facto de, actualmente, cada pessoa ter de decorar várias palavras-passe para os diferentes serviços que consulta online.

Um estudo de 2007, levado a cabo pela Microsoft, concluiu que, em média, cada pessoa utiliza 25 palavras-passe diferentes e que, diariamente, usa oito delas. Desde 2007 que este número terá certamente aumentado.

O roubo de palavras-passe é um problema que afecta muitas pessoas em todo o mundo. Em 2010 a Comissão Federal de Comércio dos EUA recebeu 1,3 milhões de queixas por fraude ou roubo de identidade.

Como pode, então, tornar a sua password mais segura? Usando uma variedade não sequencial de letras, números e símbolos e mudando a palavra-passe a cada seis meses. Outras dicas importantes: não use sempre a mesma password e evite usar palavras verdadeiras.

Finalmente, não use nenhuma destas 25 piores palavras-passe do ano elencadas pela SlashData:

1.password, 2.123456, 3.12345678, 4.qwerty, 5.abc123, 6.monkey, 7.1234567, 8.letmein, 9.trustno1, 10.dragon, 11.baseball, 12.111111, 13.iloveyou, 14.master, 15.sunshine, 16.ashley, 17.bailey, 18.passw0rd, 19.shadow, 20.123123, 21.654321, 22.superman, 23.qazwsx, 24.michael, 25. football.

Fonte: www.publico.pt

TEMA DA CAPA

iOS, Cocoa Touch & MVC

iOS, Cocoa Touch & MVC

MVC

A grande maioria das plataformas de desenvolvimento de software permitem aos programadores desenvolver software com base nos mais diversos padrões de arquitectura. Essa abordagem oferece um elevado grau de liberdade, permitindo ao programador escolher a melhor solução para o seu problema, com os recursos que dispõe.

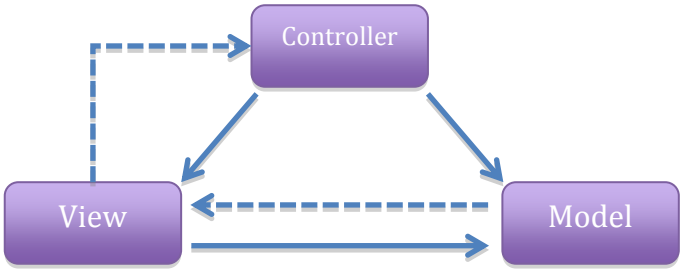


Figura 1 - MVC

Nos últimos anos tem-se vindo a assistir a uma evolução e consolidação neste campo, principalmente quando o tema são os padrões de camada de apresentação e o Cocoa Touch não foge a essa tendência. Apple definiu o padrão de arquitectura de software Model-View-Controller (MVC), como padrão de referência para o desenvolvimento de aplicações para os dispositivos que utilizam iOS (iPod, iPhone, iPad).

O tema do MVC já foi abordado na edição nº27 da revista PROGRAMAR, é pretendido que o leitor tenha em mente o conceito, pois este tema é incontornável no decorrer do artigo.

Cocoa Touch

É uma camada de alto nível do iOS, composta por um conjunto de frameworks que disponibilizam ferramentas que permitem ao leitor utilizar todo o potencial que a plataforma iOS tem para oferecer, ao mesmo tempo, permite desenvolver aplicações que transmitem ao utilizador uma sensação familiar durante a sua utilização graças à utilização transversal nas várias aplicações do Sistema Operativo.

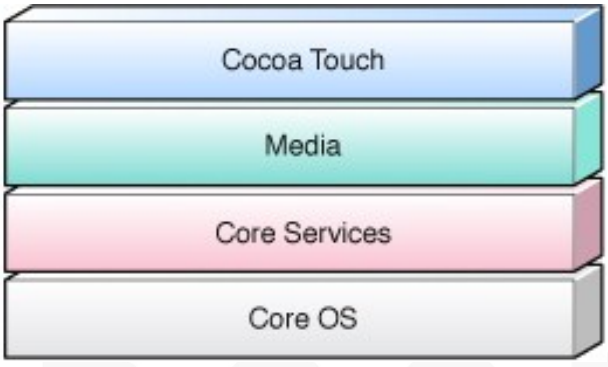


Figura 2 – A arquitectura do iOS

Entre as frameworks do Cocoa Touch de maior relevo estão as seguintes:

- Core Animation
- Core Location
- Core Audio
- Core Data
- UIKit

A mais importante das frameworks que compõem o Cocoa Touch é sem dúvida a UIKit.

Esta framework, baseada em Objective-C, disponibiliza uma vasta quantidade de funcionalidades e ferramentas que permitem ao leitor desenvolver UI, gerir eventos (toque, gestos), aceder ao acelerómetro, câmara fotográfica, bateria, sensor de proximidade e biblioteca de imagens.

Figura 3 – Alguns dos componentes de UI do UIKit

O leitor vai, com toda a certeza, reconhecer alguns dos componentes da figura 3 disponibilizados pela UIKit Framework.

A documentação é acessível e objectiva, permite desenvolver aplicações para iOS num curto espaço de tempo, o design e a experiência de utilização proporcionada pelos componentes de esta framework são transversais a todo o Sistema Operativo, o que garante ao leitor que o utilizador se vai sentir familiarizado com as suas aplicações.

É aconselhado ao leitor, que antes de prosseguir, tenha em mente o artigo *Introdução ao Objective-C e à plataforma iOS* publicado no nº 30 da revista Programar.

Leitor de RSS

Para consolidar os conceitos apresentados anteriormente, o melhor é passar à prática e desenvolver um leitor de RSS.

Execute o XCode, ferramenta incluída no iOS SDK, e crie um novo projecto recorrendo ao template *Navigation-based Application*.

iOS, Cocoa Touch & MVC

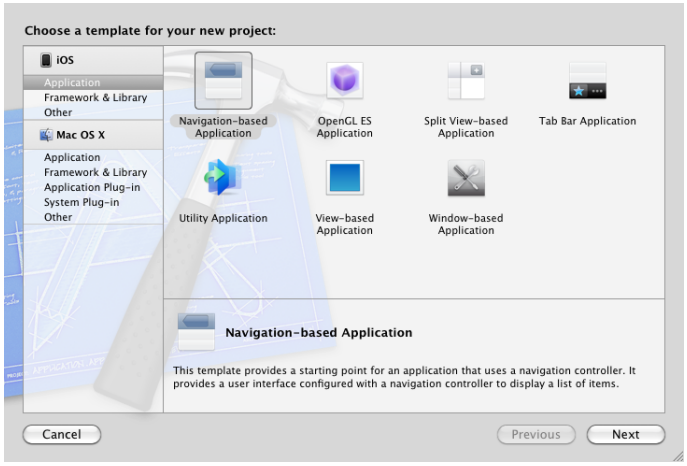


Figura 4 – Template Navigation-based Application

Este template oferece uma estrutura serve como ponto de partida para desenvolver a nossa aplicação.

Sem escrever qualquer tipo de código, se o leitor executar o projecto, vai verificar que a aplicação se apresenta já um formato reconhecível.

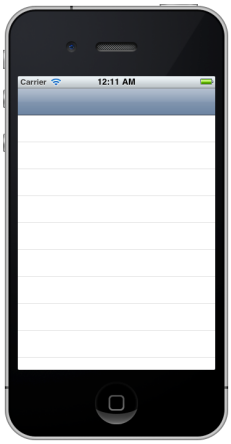
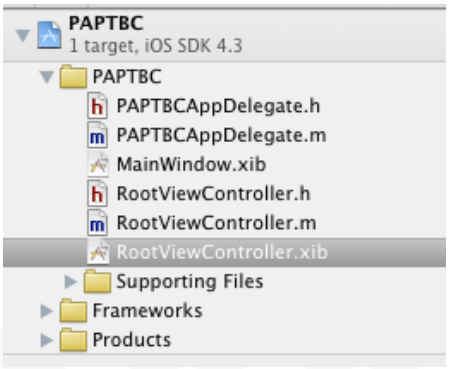


Figura 5 – Primeira execução do projecto

O template escolhido fornece um conjunto de funcionalidades básicas para uma aplicação que pretende ter um interface de navegação, o que é o caso.

Considera-se que um leitor de RSS básico, necessita de pelo menos um ecrã com uma lista de feeds e um outro que permita visualizar o detalhe de um feed.



Este template contém uma View (RootViewController.xib) e um Controller (RootViewController.h/.m).

A View, já contém um controlo do tipo UITableView, que é apresentado na figura 5 e o Controller contém os métodos e eventos necessários para fornecer ao controlo UITableView a informação necessária para apresentar os dados na View.

Sendo assim, torna-se necessário criar um Model que forneça ao Controller o tipo de objectos a serem manipulados e enviados para a View.

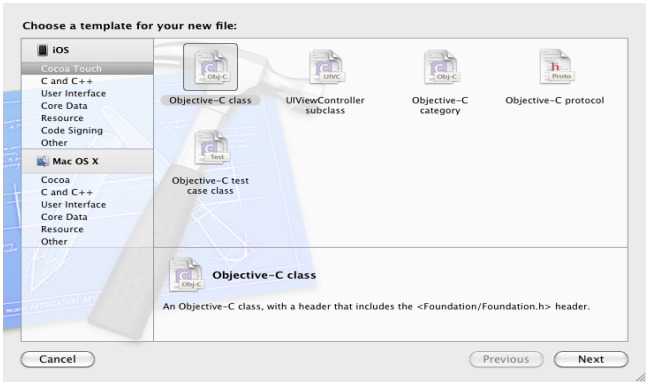


Figura 6 – Adicionar uma nova classe

É necessário adicionar uma nova classe, definida como uma sub-classe de NSObject, ao projecto com o nome RssFeed, onde vão ser definidas as propriedades de uma feed, conforme as figuras abaixo apresentam.

```
#import <Foundation/Foundation.h>
@interface RssFeed : NSObject {}
@property (nonatomic, retain) NSString *FeedTitulo;
@property (nonatomic, retain) NSString
*FeedSumario;
@property (nonatomic, retain) NSString *FeedUrl;
@end
```

Figura 7 – Interface da classe RssFeed

```
#import "RssFeed.h"
@implementation RssFeed
@synthesize FeedSumario;
@synthesize FeedTitulo;
@synthesize FeedUrl;
@end
```

Figura 8 – Implementação da classe RssFeed

Agora que já está definido o modelo, vamos criar um repositório que tem como objectivo fornecer ao Controller a lista de feeds a apresentar, repetindo o passo executado na figura 6, mas desta a classe vai chamar-se Repository.

Nesta classe são definidas uma estrutura de dados para guardar os feeds e um método que devolve uma lista de feeds, conforme as figuras embaixo apresentam.

```
#import "RssFeed.h"
#import <Foundation/Foundation.h>
@interface Repository : NSObject {
    NSMutableArray *FeedsList;
}
+ (NSArray*)getRssFeeds;
@end
```

Figura 9 – Interface da classe Repository

```
#import "Repository.h"
@implementation Repository

+ (NSArray*)getRssFeeds{

    NSMutableArray *feedList = [[NSMutableArray
alloc] init];

 for (int i = 0; i<20; i++) {
  if (i % 2 ==0) {
   RssFeed *item1 = [[RssFeed alloc] init];
   item1.FeedTitulo = @"Revista Programar Nº30";
   item1.FeedSumario = @"Publiquei na edição nº 30
   da Revista Programar um artigo com o nome
   "Introdução ao Objective-C e à plataforma iOS".
   Nesta edição os leitores tiveram 15 dias para
   votar nos seus 3 artigos favoritos. Após a
   conclusão da votação, o meu artigo ficou
   classificado em 2º lugar. Devo confessar que não
   estava à espera, ainda [...] ";
   item1.FeedUrl = @"http://blastersystems.com/
   blog/2011/08/revista-programar-n30/?
   utm_source=rss&utm_medium=rss&utm_campaign=
   revista-programar-n30";

   [feedList addObject:item1];
   [item1 release];

   }
   else{
     RssFeed *item2 = [[RssFeed alloc] init];
     item2.FeedTitulo = @"HTML5 & CSS3 no Visual
     Studio 2010";
     item2.FeedSumario = @"Muito se tem falados nos
     últimos meses nos novos standards para Web
     HTML5 e CSS3, comparações com outras
     tecnologias, milhares de blog posts, artigos
     de opinião e discussões sobre o assunto.
     Embora ainda não sejam um standard fechado, já
     existem projectos desenvolvidos outros
     projectos em desenvolvimento recorrendo a
     estas tecnologias[...]  ";
     item2.FeedUrl = @"http://blastersystems.com/
     blog/2011/06/html5-css3-no-visual-studio-2010-
     2/?utm_source=rss&utm_medium=rss&utm_campaign=
     html5-css3-no-visual-studio-2010-2";

     [feedList addObject:item2];
     [item2 release];

     }
  }
  return [feedList autorelease];
}
@end
```

Figura 10 – Implementação da classe Repository

A classe Repository contém apenas um método que constrói um conjunto de feeds estáticos e os agrega numa estrutura de dados para mais tarde alimentar o controlo UITableView.

De seguida, vamos proceder às alterações necessárias ao Controller para este obter os dados para alimentar a View. Para isso, torna-se necessário criar uma estrutura de dados no RootViewContoller.h para acomodar os dados provenientes do Repository.

```
#import "Repository.h"
#import "DetailViewController.h"
#import <UIKit/UIKit.h>
@interface RootViewController :
UITableViewController {
}
@property (nonatomic, retain) NSArray *feeds;
@end
```

Figura 11 – Interface do Controller RootViewController

Agora que existe onde guardar os dados recebidos, é necessário fornece-los ao controlo UITableViewController. Para isso, é preciso alterar o método viewDidLoad na implementação do Controller (RootViewContoller.m).

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    feeds = (NSArray*)[[Repository getRssFeeds]
retain];
}
```

Figura 12 – Alteração do método viewDidLoad no Controller RootViewController

O Controller alimenta o controlo UITableView, é notificado dos seus eventos porque é uma sub-classe de UITableViewController, o que o torna um Controller especializado em lidar com Views que contêm controlos do tipo UITableView.

A grande vantagem na utilização de este Controller, é que contém um template que ajuda a lidar com o controlo UITableView. Um dos métodos incluído nesse template é o numberOfRowsInSection, que informa o UITableView do número de elementos que vai conter.

Como o leitor já obteve do Repository os feeds, pode agora informar o UITableView do número de elementos que lhe vamos fornecer.

```
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
    return [feeds count] -1;
}
```

Figura 13 – Notificar o controlo do número de elementos que vai conter

Já existem os feeds na estrutura de dados do Controller e já informamos o UITableView do número de elementos que lhe vamos fornecer, apenas resta adicionar as feeds ao UITableView.

O leitor pode aproveitar novamente a vantagem de o Controller ser uma sub-classe de UITableViewController, porque no template já existe um método definido para fornecer os dados ao UITableView, apenas é necessário alterar um pouco o código existente.

```
- (UITableViewCell *)tableView:(UITableView *)
tableView cellForRowAtIndexPath:(NSIndexPath *)
indexPath

{
 static NSString *CellIdentifier = @"myCell";
 UITableViewCell *cell = [tableView

 dequeueReusableCellWithIdentifier:CellIdentifier];

  if (cell == nil) {
    cell = [[[UITableViewCell alloc]
    initWithStyle:UITableViewCellStyleDefault
    reuseIdentifier:CellIdentifier] autorelease];
    }

     if (feeds !=nil)
     {
     [cell.textLabel setText:((RssFeed*) [feeds
     objectAtIndex:indexPath.row]).FeedTitulo];

     }
    return cell;
}
```

Figura 14 – Alimentar o UITableView com os dados a apresentar

Com base no número de feeds vai apresentar, percorre a estrutura de dados que contém as feeds a passa o título da feed à célula.

Estas células são objectos do tipo UITableViewCell, e são altamente extensíveis, podendo o leitor customiza-las e apresentar a informação de outras formas.

Para concluir esta primeira fase do leitor de RSS, dado que a gestão de memória dos objectos criados pelo programador se encontram a seu cargo, torna-se obrigatório libertar a memória alocada à estrutura de dados que contém as feeds quando a View é libertada.

```
- (void)dealloc
{
    [super dealloc];
    [feeds release];
}
```

Figura 15 – Libertação da memória alocada

Agora, apenas resta ao leitor testar o código produzido, para isso basta executar o projecto.

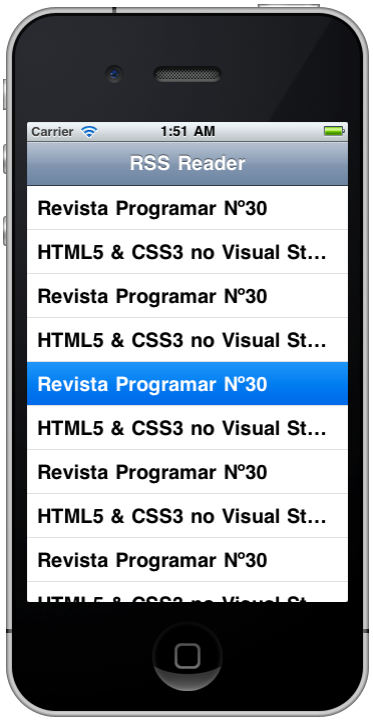


Figura 17 - Primeira execução do projecto

Nesta fase temos um leitor de RSS parcialmente construído, já apresentamos no dispositivo a lista de feeds. No entanto no início do projecto definimos que também iriamos disponibilizar uma View para o detalhe do feed.

Para isso vamos adicionar um novo Controller, o DetailViewController, mas desta vez o novo Controller é uma sub-classe de UIViewController.

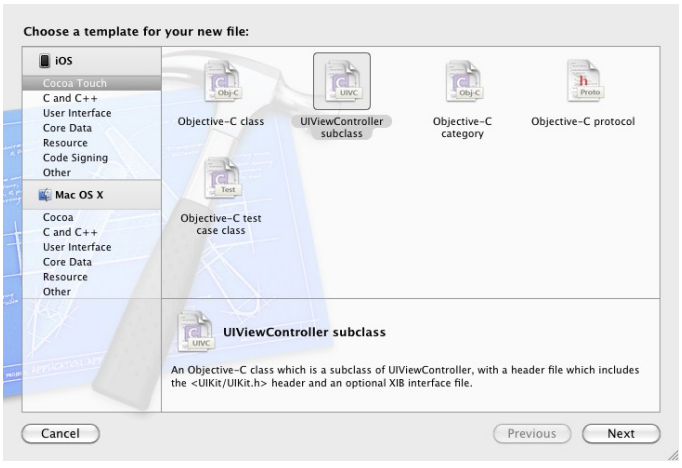


Figura 18 – Adicionar um novo Controller e View para o detalhe do feed

Quando é criado, é composto por 3 ficheiros, um para definir a interface do Controller outro para definir a implementação do Controller e ainda um terceiro para definir a View que vai exibir o detalhe dos feeds.

Para apresentar o detalhe da feed, é proposto criar um UILabel para apresentar o título da feed, um UITextView para apresentar o sumário do feed e um UIButton para abrir a fonte da feed no safari.

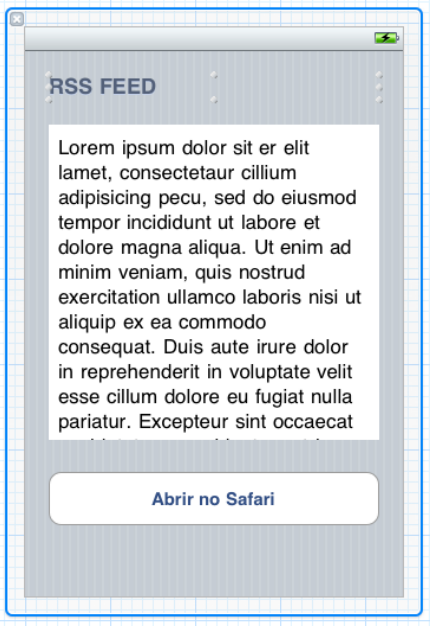


Figura 19 – View de detalhe de feeds

Agora é necessário definir o Controller da View acima apresentada, e para isso basta colocar na interface do DetailViewController a definição das propriedades que compõem a View bem como definir um novo construtor para o Controller, para suportar receber um objecto do tipo RssFeed quando é instanciado.

```
#import "RssFeed.h"
#import <UIKit/UIKit.h>
@interface DetailViewController : UIViewController

@property (retain, nonatomic) IBOutlet UILabel
*FeedTitle;
@property (retain, nonatomic) IBOutlet UIButton
*FeedOnSafari;@property (retain, nonatomic)
IBOutlet UITextView *FeedSummary;

@property (retain, nonatomic) RssFeed *feedDetail;
- (id)initWithFeed:(RssFeed *)feed;
@end
```

Figura 20 – Definição do Interface do Controller DetailViewController

No ficheiro de implementação do DetailViewController é necessário implementar o novo construtor, a gestão de memória dos objectos criados e gerir o evento TouchDown do botão FeedOnSafari.

```
#import "DetailViewController.h"
@implementation DetailViewController
@synthesize FeedOnSafari;
@synthesize FeedSummary;
@synthesize FeedTitle;
@synthesize feedDetail;

- (id)initWithFeed:(RssFeed *)feed{

    self = [super init];
    if (self) {
        feedDetail = feed;
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    [FeedTitle setText:feedDetail.FeedTitulo];
    [FeedSummary setText:feedDetail.FeedSumario];
}

- (void)viewDidUnload
{
    [super viewDidUnload];
}

-(IBAction)linkButtonClick:(id)sender {

    NSString* launchUrl = feedDetail.FeedUrl;;
    [[UIApplication sharedApplication] openURL:
[NSURL URLWithString: launchUrl]];
}

- (void)dealloc
{
    [super dealloc];
    [feedDetail release];
}
@end
```

Figura 21 – Implementação do Controller DetailViewController 7

Para terminar apenas falta referenciar os componentes definidos no Controller com os componentes definidos na View.

Esse processo é executado através de outlets, que nada mais são do que apontadores que permitem ao compilador saber onde se encontra no Controller definido um componente definido na View e qual a acção que o Controller necessita de executar quando um desses componentes dispara um evento.

Neste caso, o botão feedOnSafari, quando a View detecta um toque, pede ao Controller para executar o evento linkButtonClick, que por sua vez vai lançar o Safari cmo o URL da feed.

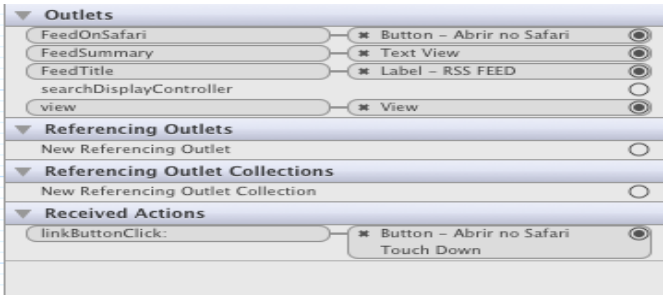


Figura 22 – Outlets referenciadas no Organizer da View

Dá-se por concluído o ecrã de detalhe da feed, apenas falta alterar o ficheiro de implementação do RootViewController para que este navegue para a View de detalhe quando é selecionada uma feed.

É também necessário alterar o método cellForRowAtIndexPath e adicionar um indicador na célula de cada feed para que o utilizador compreenda que existe um ecrã de detalhe para o qual pode navegar e alterar também o método didSelectRowAtIndexPath para este invocar ao navigationController a navegação para um novo ecrã.

```
- (UITableViewCell *)tableView:(UITableView *)
tableView cellForRowAtIndexPath:(NSIndexPath *)
indexPath
{
 static NSString *CellIdentifier = @"myCell";
 UITableViewCell *cell = [tableView
 dequeueReusableCellWithIdentifier:CellIdentifier];
  if (cell == nil) {
      cell = [[[UITableViewCell alloc]
      initWithStyle:UITableViewCellStyleDefault
      reuseIdentifier:CellIdentifier] autorelease];
    }
 cell.accessoryType =
 UITableViewCellSelectionStyleBlue;

 if (feeds !=nil) {
 [cell.textLabel setText:((RssFeed*)
 [feeds objectAtIndex:indexPath.row]).FeedTitulo];
  }
    return cell;
}
```

Figura 23 – Alteração necessária no método já definido na figura 14

```
(void)tableView:(UITableView *)tableView

didSelectRowAtIndexPath:(NSIndexPath *)indexPath

{

 DetailViewController *details =
[[DetailViewController alloc] initWithFeed:

 (RssFeed*) [feeds objectAtIndex:indexPath.row]];
 [self.navigationController

  pushViewController:details animated:YES];

 [details release];

}
```

Figura 24 – Método a alterar para existir navegar para o detalhe da feed selecionada

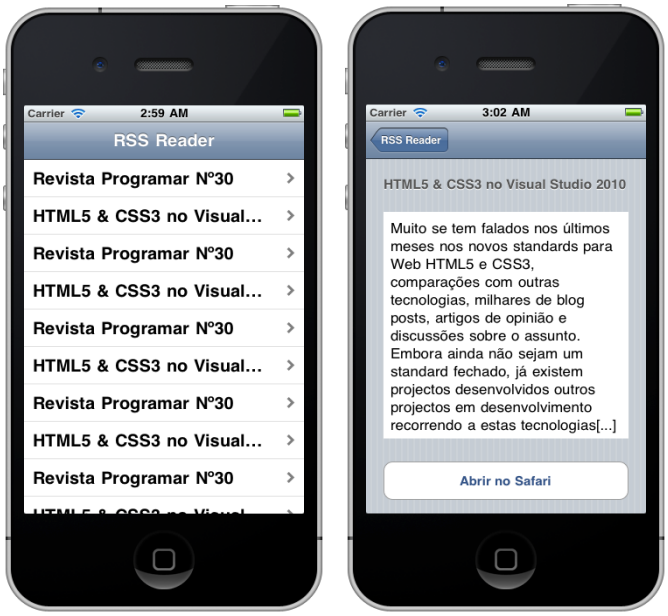
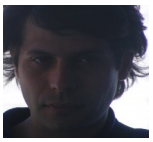


Figura 25 – Comportamento final da aplicação

O código fonte da aplicação encontra-se disponível em http://blastersystems.com/ContentDownload/PAPTBC.zip

AUTOR



Escrito por Bruno Pires
Exerce funções de consultor de IT na Novabase desde 2008, com experiência de maior relevo nas áreas da banca e televisão digital, onde ganhou competências nas mais várias tecnologias. Membro da Comunidade NetPonto (http://netponto.org) e autor do blog http://blog.blastersystems.com - Twitter: @brunoacpires

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://tiny.cc/ProgramarED32_V

A PROGRAMAR

Geração de números aleatórios (Parte 2)

Automatização de tarefas usando atributos

SQL Azure Fedarations

Enigmas de C#: Arrays

GNA - GERAÇÃO DE NÚMEROS ALEATÓRIOS (Parte 2)

No artigo anterior foi indicado o método de geração de números pseudo-aleatórios meio do quadrado. Em continuação será mostrado neste o método do produto do meio.

INTRODUÇÃO

O método de geração de números pseudo-aleatórios produto do meio é indicado em Banks et al. (2009), sendo um algoritmo idêntico ao algoritmo utilizado para o quadrado do meio. A diferença deste método está no fato de pegar-se dois valores como sementes com o mesmo número de dígitos e sobre estes valores traçar a acção do algoritmo.

ALGORITMO DO MEIO DO PRODUTO

Para fazer uso do método do meio do produto pega-se em dois valores de sementes e efectua-se a sua multiplicação. É ideal que os valores de sementes sejam valores com um tamanho de cinco dígitos e que não possuam na sua estrutura valores zero em qualquer uma das posições.

Do produto obtido tira-se o valor do meio contento o mesmo número de dígitos das sementes utilizadas. Para sementes iniciais de cinco dígitos obter-se-á um produto do meio que deve ter um tamanho de onze dígitos. Se o valor obtido não tiver onze dígitos de tamanho torna-se necessário acrescentar á esquerda do valor tantos zeros quantos necessários para completar o tamanho de onze dígitos. Esta atitude é necessária no sentido de que se consiga extrair exactamente cinco dígitos do meio deixando de cada lado três dígitos de distância.

De seguida multiplica-se o valor do meio do produto com o valor da segunda semente e obtém-se assim um novo produto. Tira-se deste novo produto os dígitos do novo meio com a mesma quantidade de dígitos do valor da semente e efectua-se a multiplicação deste valor pelo valor do meio anterior e assim por diante.

A tabela a seguir apresenta a sequência de geração de dez valores obtidos a partir do método do meio do. Assim, considere os como sementes os valores 32534 e 58461

Iteração	Valor Multiplicado	Produto do Meio
0	32534 58461	0190**1970**174
1	58461 01970	00**1151**68170
2	01970 15168	000**2988**0960
3	15168 29880	004**5321**9840
4	29880 53219	015**9018**3720
5	53219 90183	047**9944**9077
6	90183 99449	089**6860**9167
7	99449 68609	068**2309**6441
8	68609 23096	015**8459**3464
9	23096 84593	019**5375**9928

Observe que após aplicação do método são obtidos os valores 01970, 15168, 29880, 53219, 90183, 99449, 68609, 23096, 84593 e 53759.

A aplicação do método pode ser realizada a partir do algoritmo seguinte:

N1/N2 = informação obtida manualmente ou por outro meio para a formação dos números das duas sementes iniciais, tendo o tamanho de cinco dígitos.

p = resultado do produto obtido com a multiplicação dos valores de N1 com N2 (p = N1 x N2).

td = tamanho em dígitos do resultado do produto de N1 e N2: td = tamanho(p).

```
SE (td < 11) ENTÃO
  T = 11 – td
  ZEROS = 0
  PARA I = 1, T, 1
    ZEROS = ZEROS + “0”
  FIM_PARA
  p = ZEROS + p
FIM_SE
```

GNA - GERAÇÃO DE NÚMEROS ALEATÓRIOS (Parte 2)

T = tamanho ajustado para onze dígitos, onde 11-*td* igual ao número de zeros a ser inserido à esquerda de *p*, se *td* for menor que 11.

MP = meio do produto que será utilizado como o valor da próxima semente que é a extração do valor central de cinco dígitos do valor p, ou seja, cinco dígitos a partir do quarto dígito.

N1 = passa a ser o valor de N2.

N2 = passa a ser o valor de MP.

$$MP = \left(\begin{matrix} P \\ 4,5 \end{matrix} \right)$$

CONCLUSÃO

Neste artigo foi apresentado o método de geração de números pseudo-aleatórios meio do produto, sendo este mais eficiente que o método do meio do quadrado, pois demora mais a apresentar valores zero.

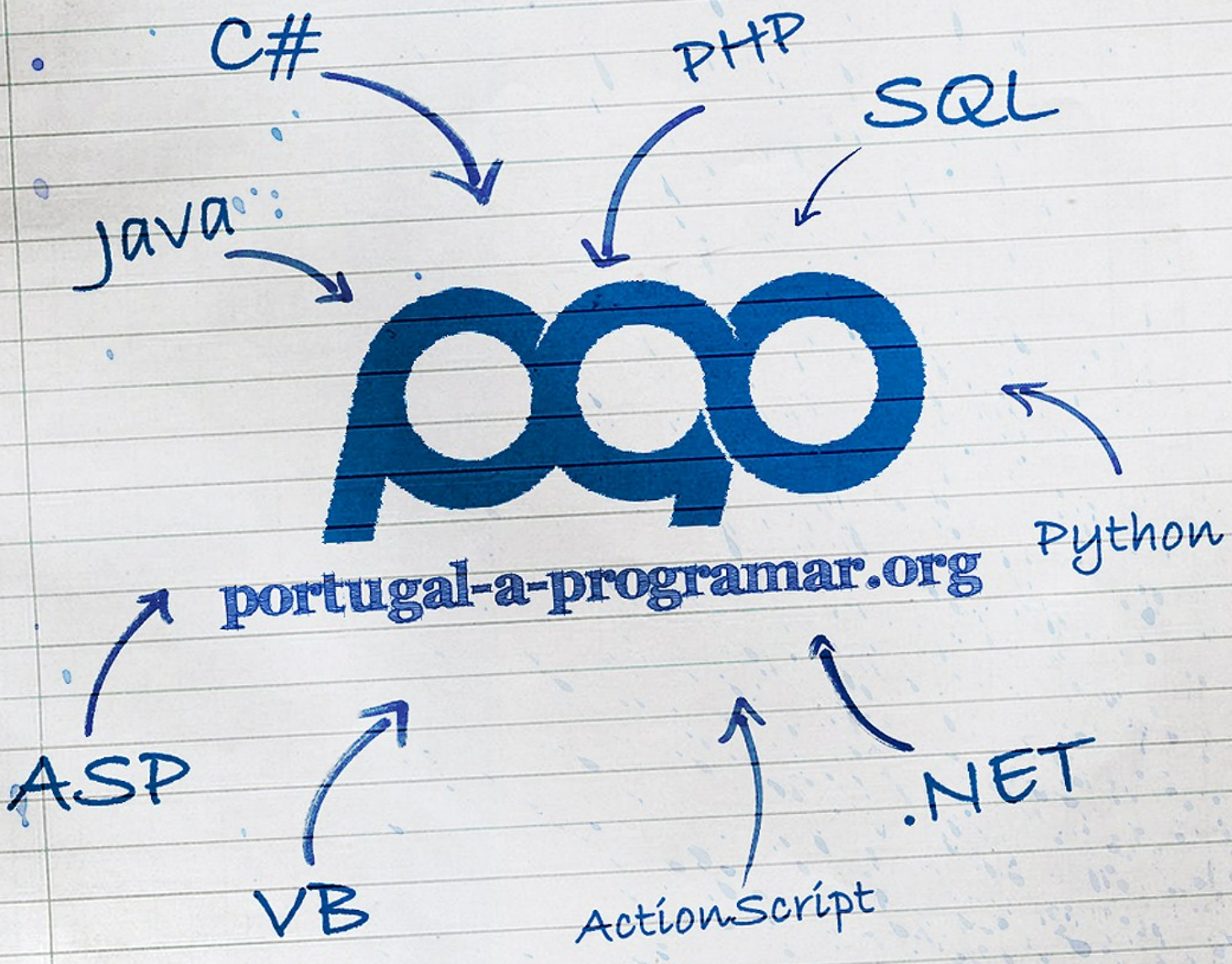
BIBLIOGRAFIA

BANKS, J., CARSON, J. S., NELSON, B. L. & NICOL, D. M. **Discrete Event System Simulation**. 5. ed. USA: Pearson Education. 2010.

AUTOR

Escrito por Augusto Manzano
Natural da Cidade de São Paulo, tem experiência em ensino e desenvolvimento de programação de software desde 1 986. 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.

A maior comunidade portuguesa de programação!



Visite-nos!

Automatização de tarefas usando atributos

Introdução

Na edição número 30 da revista Portugal a Programar falámos de atributos em C#. Vimos como podíamos usar esta funcionalidade para reconhecer um certo tipo de classes, ou mesmo atribuir-lhes certas funcionalidades. Sendo uma funcionalidade muito interessante convenhamos que, na prática, muito raramente a iremos utilizar. Se quisermos caracterizar uma determinada classe ou que esta tenha determinados comportamentos provavelmente iremos usar uma interface ou herança. Se quisermos guardar informação ou implementar certas funcionalidades o mais certo é usarmos um delegate. Ou então implementar na própria classe.

Sendo assim, muito provavelmente, tirando os atributos da framework como o Serializable ou o WebMethod dificilmente iremos usar outros.

No entanto, há um tipo de atributos que pode ser muito útil. Acima de tudo porque, ao contrário daqueles que falámos no último artigo, não precisa de ser chamado por nós. Ao invés disso é automaticamente chamado pela framework quando um objecto nosso – que tenha esse atributo – é usado.

Criar um proxy de objectos

Este funcionamento parte do conceito de contextos. Quando um objecto é criado, é criado também um contexto, onde este objecto está inserido. Este contexto tem uma cadeia de funções, que são executadas cada vez que se entra ou sai do contexto, ou seja, cada vez que se chama um método, ou um método retorna. Aquilo que podemos fazer é colocar a nossa funcionalidade nesse processo. Ou seja, criar uma espécie de proxy.

Com um exemplo prático talvez se perceba mais facilmente. Vamos então fazer um sistema de log's que nos permite saber qual a classe e o método que está a ser chamado, quais os argumentos, e quanto tempo demora a execução do método.

Naturalmente temos de começar por criar o nosso atributo:

```
class LogAttribute : ContextAttribute
    {
        // Temos de passar para o pai o nome do
        //atributo de contexto.
        public LogAttribute(): base("Log"){}
        /*Neste método adicionamos a nossa property
        * ao contexto que está a ser criado para o
        * objecto */

public override void GetPropertiesForNewContext
(IConstructionCallMessage ctorMsg)
        {
            ctorMsg.ContextProperties.Add(new
LogProperty());
        }
    }
```

Como se pode ver, neste caso, não herdamos da mesma classe que no artigo anterior. Em vez de herdarmos de Attribute vamos herdar de ContextAttribute. Isto indica à framework que este é um atributo especial e que queremos que seja tido em conta na criação de contextos.

Um dos métodos a que temos de fazer override é o GetPropertiesForNewContext. Neste método temos como tarefa adicionar uma propriedade nossa à lista de propriedades do novo contexto.

Ora, se usamos a propriedade, temos de a criar:

```
class LogProperty : IContextProperty, IContribute-
ObjectSink
    {
        /*
         * Temos de indicar qual o nome
         * pela qual a property vai ser
         * conhecida dentro do contexto
         */
        public string Name
        {
            get { return "Log"; }
        }

        /*
         * O que fazer quando o contexto fica
         *  bloqueado. Tipicamente não é relevante
         */
        public void Freeze(Context newContext)
        {}
        // Temos de verificar se o novo contexto
        // está OK

       public bool IsNewContextOK(Context newCtx)
        {
        // Temos de verificar se o novo contexto
        // que recebemos tem a nossa property
       LogProperty p = newCtx.GetProperty("Log") as
            LogProperty;
            if (p == null)
            {
              // Se não tiver alguma coisa está mal
                 return false;
            }
            // Se tiver continuamos
            return true;
        }
```

Nos comentários está a explicação para a maior parte do código. No entanto, há um método que é especial. É no método GetObjectSink que a «magia» vai acontecer. Até agora temos estado a criar as classes necessárias para construir a cadeia de objectos que vão ser chamados no contexto. Mas é aqui que efectivamente estamos a colocar o nosso código.

Se precisamos de uma mensagem, vamos então cria-la:. Nos comentários está a explicação para a maior parte do código. No entanto, há um método que é especial. É no método GetObjectSink que a «magia» vai acontecer. Até agora temos estado a criar as classes necessárias para construir a cadeia de objectos que vão ser chamados no contexto. Mas é aqui que efectivamente estamos a colocar o nosso código. O LogSink é uma mensagem que, para todos os efeitos, vai ser o nosso proxy. As mensagens vão sendo chamadas, sequencialmente, até ao objecto final. Neste método recebemos o objecto propriamente dito, e qual a mensagem a ser chamada após a nossa. Aquilo que temos de fazer é criar a nossa mensagem, onde guardamos quem é o próximo na fila, e devolvemos a mensagem para a plataforma.

Se precisamos de uma mensagem, vamos então cria-la:

```
class LogSink : IMessageSink
    {
        // Precisamos de guardar a próxima
        // mensagem para a poder retornar

        private IMessageSink nextSink;
        public LogSink(IMessageSink nextSink) {
            this.nextSink = nextSink;
        }
        public IMessageSink NextSink
        {
            get { return this.nextSink; }
        }

// Comportamento a executar caso a mensagem seja
// chamada de forma assíncrona Vamo-nos limitar a
// chamar a próxima mensagem e devolver o resultado

        public IMessageCtrl AsyncProcessMessage
            (IMessage msg, IMessageSink replySink)
        {
            IMessageCtrl rtnMsgCtrl = nextSink.
            AsyncProcessMessage(msg, replySink);
            return rtnMsgCtrl;
        }


// Aqui é onde vamos fazer a nossa intercepção
// Por agora vamos, simplesmente, chamar
// a próxima mensagem.

    public IMessage SyncProcessMessage(IMessage msg)
        {
            IMessage rtnMsg =
            nextSink.SyncProcessMessage(msg);
            IMethodReturnMessage mrm = (rtnMsg as
            IMethodReturnMessage);
            return mrm;
        }
    }
```

Como se pode ver este objecto é, por agora, relativamente simples. Limitamo-nos a chamar a mensagem seguinte na cadeia, e retornamos a resposta. Um proxy sem lógica de negócio.

Usar o proxy

Mas como se usam estes atributos? Vamos criar uma classe simples que vai ser sujeita a registo através do nosso logger:

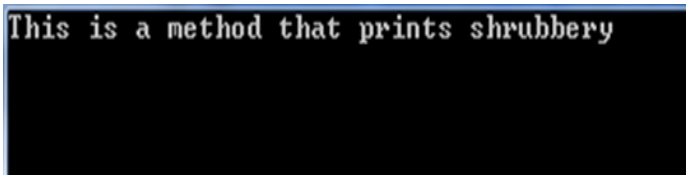
```
[Log]
    public class LoggedObject : ContextBoundObject
    {
        public void metodo(String str)
        {
            Console.WriteLine("This is a method
                        that prints " + str);
        }
    }
```

Como podemos ver é bastante simples. Tal como nos atributos normais, basta colocar o nosso atributo como atributo desta classe. Tem, no entanto, um pequeno truque. Desde o inicio que se tem falado de contextos. Que estes atributos são usados quando entramos ou saímos de um contexto. Temos então de dizer à framework que queremos que este objecto esteja associado a um contexto. Para isso basta herdar da classe ContextBoundObject.

Vejamos então o que acontece se tentarmos usar este objecto:

```
static void Main(string[] args)
        {
            LoggedObject lo = new LoggedObject();
            lo.Metodo("shrubbery");
        }
```

Aquilo que vemos é que acontece muito pouco:



O nosso método imprime aquilo que é esperado dele, e nada mais.

Automatização de tarefas usando atributos

Antes de tentarmos ir mais adiante com o nosso logger, vamos ver se a nossa mensagem está a ser correctamente executada na cadeia. Vamos voltar ao método SyncProcessMessage do objecto LogSink e colocar o nosso debug:

```
public IMessage SyncProcessMessage(IMessage msg)
 {
  Console.WriteLine("Antes de executar o método.");
            IMessage rtnMsg =
            nextSink.SyncProcessMessage(msg);
  Console.WriteLine("Depois de executar o
                    método.");
  IMethodReturnMessage mrm = (rtnMsg as
                   IMethodReturnMessage);
           return mrm;
  }
```

Se corrermos a aplicação novamente já vemos algo de diferente:

```
Antes de executar o método.
This is a method that prints shrubbery
Depois de executar o método.
```

Estamos no bom caminho. A nossa mensagem está a ser correctamente colocada na cadeia de mensagens do objecto e, quando o método do objecto é chamado, a nossa mensagem faz o seu trabalho.

Vamos então tentar transformar o nosso sink num logger com informação util:

```
public IMessage SyncProcessMessage(IMessage msg)
  {
   // A mensagem que recebemos tem
   // a informação que procuramos

    IMethodCallMessage mcm =
           (msg as IMethodCallMessage);

  Console.WriteLine("Tipo do objecto: " +
                    mcm.TypeName);
  Console.WriteLine("Nome do método: " +
                    mcm.MethodName);

  Console.WriteLine("Parametros: " +
              String.Join(", ", mcm.Args));

  IMessage rtnMsg = nextSink.SyncProcessMessage
           (msg);

  IMethodReturnMessage mrm = (rtnMsg as
                    IMethodReturnMessage);

  return mrm;

  }
```

Se tentarmos correr este código já vemos algo mais interessante:

```
Tipo do objecto: ProxyAttributesArticle.LoggedObject, ProxyAttributesArticle, Ve
rsion=1.0.0.0, Culture=neutral, PublicKeyToken=null
Nome do método: Metodo
Parametros: shrubbery
This is a method that prints shrubbery
```

Podemos ver que executámos um LoggedObject, qual o nome do método que foi executado, e o parâmetro que foi passado. Falta-nos apenas do tempo que demorou a correr:

```
public IMessage SyncProcessMessage(IMessage msg)
   {
    // A mensagem que recebemos tem a
   // informação que procuramos
   MethodCallMessage mcm = (msg as
    IMethodCallMessage);

   Console.WriteLine("Tipo do objecto: " +
                              mcm.TypeName);
   Console.WriteLine("Nome do método: " +
                             mcm.MethodName);
   Console.WriteLine("Parametros: " + String.Join
                          (", ", mcm.Args));
   DateTime before = DateTime.Now;
   IMessage rtnMsg = nextSink.SyncProcessMessage
                          (msg);
   DateTime after = DateTime.Now;

   Console.WriteLine("Demorou " + (after - before)
                  + " a correr.");
   IMethodReturnMessage mrm = (rtnMsg as
                  IMethodReturnMessage);
   return mrm;
   }
```

No entanto, visto que o nosso método é demasiado básico, vamos faze-lo demorar mais tempo artificialmente:

```
public void Metodo(String str)
   {
    Console.WriteLine("This is a method that prints
           " + str + " once");
    Thread.Sleep(1500);
    Console.WriteLine("And prints " + str + "
           again");
    }
```

Se corrermos isto vemos que, para além do segundo e meio que inserimos, demorou mais uns milésimos de segundo:

```
Tipo do objecto: ProxyAttributesArticle.LoggedObject, ProxyAttributesArticle, Ve
rsion=1.0.0.0, Culture=neutral, PublicKeyToken=null
Nome do método: Metodo
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:01.5010858 a correr.
```

Temos assim uma forma de registar todos os acessos a um determinado objecto.

Aqui, para efeitos demonstrativos, imprimimos a informação para o ecrã, no entanto pode muito facilmente ser guardada, por exemplo, numa base de dados. Pode-se até ir um pouco mais longe e registar também qual é o utilizador que está a fazer o acesso. E até criar um sistema de permissões. Facilmente, sabendo qual o utilizador que está a tentar usar a funcionalidade, e que tipo de permissões tem, se pode permitir ou impedir que use essa funcionalidade da aplicação.

Não há soluções, há compromissos

No entanto, como em quase todas as decisões de arquitectura deste tipo, é necessário ter em conta aquilo que se tem de abdicar quando se quer certas regalias.

Esta funcionalidade tem um grande peso em termos de performance. Só para se ter uma ideia, vamos fazer algumas medições.

Para começar, vamos ver quanto tempo demora para imprimir 10000 vezes a mesma frase, usando o nosso logger. Vamos alterar o método Main:

```
static void Main(string[] args)
 {
   LoggedObject lo = new LoggedObject();
   DateTime before = DateTime.Now;
   int sum = 0;
   for (int i = 0; i < 10000; i++)
     {
       lo.Metodo("shrubbery");
     }
    DateTime after = DateTime.Now;

    Console.WriteLine("\n\n\nDemorou " +
        (after - before) + " a correr.");

           WaitForKeyPressAndExit();
 }
```

```
rsion=1.0.0.0, Culture=neutral, PublicKeyToken=null
Nome do método: Metodo
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:00.0010001 a correr.
Tipo do objecto: ProxyAttributesArticle.LoggedObject, ProxyAttributesArticle, Ve
rsion=1.0.0.0, Culture=neutral, PublicKeyToken=null
Nome do método: Metodo
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:00 a correr.
Tipo do objecto: ProxyAttributesArticle.LoggedObject, ProxyAttributesArticle, Ve
rsion=1.0.0.0, Culture=neutral, PublicKeyToken=null
Nome do método: Metodo
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:00 a correr.


Demorou 00:00:11.4136528 a correr.
```

Vemos que este código demorou 11 segundos a correr (NOTA: foi retirado o sleep do método que imprime as duas linhas).

Agora, criamos um objecto igual, mas que não herda de ContextBoundObject nem usa o atributo de log, e alteramos novamente o Main:

```
static void Main(string[] args)
   {
    NotLoggedObject nlo = new NotLoggedObject();

    DateTime before = DateTime.Now;
    for (int i = 0; i < 10000; i++)
     {
      String str = "shrubbery";
      Console.WriteLine("Tipo do objecto: " +
            nlo.GetType().ToString());
      Console.WriteLine("Nome do método: " +
            "Soma");
      Console.WriteLine("Parametros: " + str);
      DateTime beforeSum = DateTime.Now;
      nlo.Metodo(str);
      DateTime afterSum = DateTime.Now;

      Console.WriteLine("Demorou " + (afterSum -
                        beforeSum) + " a correr.");
     }
     DateTime after = DateTime.Now;
     Console.WriteLine("\n\n\nDemorou " +
         (after - before) + " a correr.");
     WaitForKeyPressAndExit();
   }
```

Temos o seguinte resultado:

```
And prints shrubbery again
Demorou 00:00:00 a correr.
Tipo do objecto: ProxyAttributesArticle.NotLoggedObject
Nome do método: Soma
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:00.0010001 a correr.
Tipo do objecto: ProxyAttributesArticle.NotLoggedObject
Nome do método: Soma
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:00 a correr.
Tipo do objecto: ProxyAttributesArticle.NotLoggedObject
Nome do método: Soma
Parametros: shrubbery
This is a method that prints shrubbery once
And prints shrubbery again
Demorou 00:00:00 a correr.


Demorou 00:00:07.0424028 a correr.
```

Com programação tradicional temos um código quase 40% mais rápido.

Isto, em sistemas críticos, pode ser o suficiente para não termos a disponibilidade que esperamos.

Conclusão

Utilizando o contexto do objecto podemos injectar um proxy que vai ser chamado sempre que esse objecto seja utilizado. Desta forma podemos criar um proxy que nos permite adicionar lógica de controlo antes e depois da execução de um método. Este proxy é chamado automaticamente pela framework fazendo com que deixe de ser uma preocupação nossa. Teremos assim menos repetição de código e um controlo mais fino na execução da nossa aplicação.

Mas isso não vem sem um custo. Neste caso um custo de performance. Este é um processo pesado e que pode levar a que uma determinada aplicação deixe de ser útil, pois não faz o seu serviço no tempo definido. Há sempre formas de aumentar a capacidade de um serviço, adicionar máquinas a um cluster, aumentar a capacidade de processamento, discos/memórias mais rápidas... No entanto tudo isto custa dinheiro. Além de que, muitas vezes, há optimizações que têm de ser feitas no código, ao nível dos algoritmos e das bibliotecas usadas. No entanto, nos casos em que é possível de ser usado, temos aqui uma forma de ter código bastante limpo, que simplifica em muito certas tarefas que são necessárias de fazer de forma repetida.

AUTOR

Escrito por Flávio Geraldes

Licenciou-se em Engenharia Informática e Computadores no Instituto Superior Técnico tendo-se especializado na área de Programação e Sistemas de Informação. Após a finalização do curso juntou-se à Sybase SBS Software onde teve oportunidade de trabalhar com várias tecnologias focando-se particularmente em .NET. Actualmente é consultor da Sybase SBS Software numa empresa de telecomunicações onde é responsável pelo desenho e desenvolvimento de várias aplicações.

Enigmas de C# Arrays

por Paulo Morgado

Dadas as seguintes classes:

```
public class Posição
{
}

public class Posição2D : Posição
{
    public int X { get; set; }
    public int Y { get; set; }
}

public class Posição3D : Posição2D
{
    public int Z { get; set; }
}
```

Qual é a saída do seguinte código?

```
public static void Exec()
{
    var array = new Posição3D[]
    {
        new Posição3D
        {
            X = 1,
            Y = 2,
            Z = 3
        },
        new Posição3D
        {
            X = 2,
            Y = 1,
            Z = 3
        },
        new Posição3D
        {
            X = 3,
            Y = 2,
            Z = 1
        }
    };

    try
    {
        Process(array);
        Console.WriteLine("Dados processados!");
    }
    catch (Exception ex)
    {
        Console.WriteLine("{0}: {1}",
            ex.GetType().FullName,
            ex.Message);
    }
}
```

Veja a resposta na página 29

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://tiny.cc/ProgramarED32_V

Introdução às SQL Azure Federations

O cloud computing tem sido um assunto muito debatido, no mundo das tecnologias de informação não só pela capacidade de computação de alta disponibilidade, escalabilidade e elasticidade bem como pela redução de custos que o modelo baseado em serviços proporciona.

O SQL Azure da Microsoft fornece um conjunto de serviços, que implementa as capacidades de armazenamento e processamento de dados relacionais na cloud. O SQL Azure é um serviço de alta disponibilidade e tolerante a falhas, onde os programadores não têm que se preocupar com instalação, configuração ou manutenção dos servidores de bases de dados.

SQL Azure é baseado numa versão especial do Microsoft SQL Server pelo que permite ainda ao programador ser bastante produtivo dado que usa o mesmo modelo relacional baseado em T-SQL e permite usar as mesmas ferramentas de desenvolvimento e gestão usadas nas bases de dados locais.

Actualmente, dentro da Web Editions, podemos adquirir bases de dados até 5Gb onde se paga cerca de 9.99$ se a base de dados tiver até 1Gb ou 49.95$ se o tamanho for entre 1 e 5Gb. Dentro da Business Edition podemos ter uma base de dados até 50Gb onde o pagamento é fraccionado em blocos de 10Gb sendo que cada 10Gb custa 99.99$.

Escalabilidade

Scale Up

Actualmente o modelo de escalabilidade mais simples de utilizar em SQL Azure é a escalabilidade vertical (scale up), ou seja, aumentar o tamanho máximo de armazenamento na base de dados.

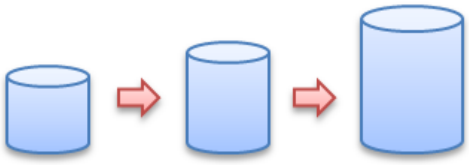


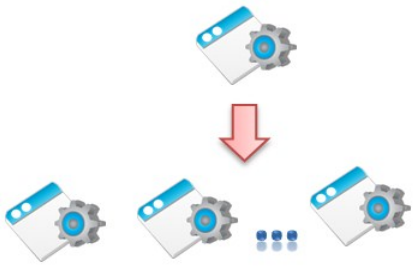
Figura 1.1 – Escalabilidade Vertical (Scale Up)

Como vimos anteriormente a escalabilidade em SQL Azure está, em termos de capacidade de armazenamento limitada a 50Gb. Brevemente irá haver um aumento deste limite para 150Gb mas o que se pretende que o leitor retenha é que a capacidade de armazenamento em cada base de dados é limitada. Poderemos chegar a uma altura em que a capacidade de armazenamento disponível não é suficiente.

Scale Out

Outra limitação natural, quem em muitos cenários se revela mais importante ainda, é a capacidade de processamento do servidor. Quantas vezes as necessidades de escalabilidade são provocadas pelo aumento da carga de processamento? Imagine um website de vendas de bilhetes onde devido ao lançamento de um evento o número de pedidos ao servidor aumenta drasticamente durante alguns dias.

Numa aplicação multicamada suportada pela plataforma Windows Azure, a gestão da capacidade de resposta a pedidos na camada de apresentação ou na camada lógica de negócio é simples, podemos por exemplo aumentar ou diminuir o número de servidores (web e worker roles) consoante a carga a que o website está a ser sujeita (scale out).



O problema é que este modelo de escalabilidade elástica não estava presente na camada de acesso a dados (base de dados) sem que houvesse a necessidade de o implementar.

As SQL Azure Federations vêm fornecer à camada de acesso a dados essa capacidade, ou seja, será possível aumentar ou diminuir o número de nós que constitui a camada de acesso a dados sem que a mesma fique indisponível durante essas alterações. Vêm ainda permitir um aumento mais granular do espaço de armazenamento e consequente poupança de custos.

Outro "cliente" muito importante das federations serão as aplicações multi-tenant, vamos tentar perceber porquê.

Microsoft® SQL Azure™

Multi-tenant

Quando se desenha uma aplicação multi-tenant (multi-cliente), uma das primeiras e grandes opções que se tem que tomar tem a ver com a distribuição ou não dos dados dos clientes por várias base de dados.

De uma forma geral podemos optar por meter todos os dados dos clientes numa única base de dados ou ter uma base de dados para cada cliente. Existe ainda uma opção intermédia que é ter vários schema dentro de uma única base de dados onde cada cliente tem um schema. Não iremos considerar esta opção por se enquadrar dentro da opção de guardar todos os dados numa única base de dados relativamente à análise que pretendemos fazer.

Quando o alojamento é on-premisses (local), a opção mais comum é o fornecimento de uma base de dados para cada cliente.

Uma base de dados por cliente

Figura 2.1 – Uma base de dados por cliente (tenant)

Esta opção tem as suas vantagens e desvantagens mas o que importa reter neste caso é que, no alojamento local, não existe o limite mínimo em relação ao custo de uma base de dados. Na cloud este limite mínimo é actualmente de 1Gb, ou seja, o custo mínimo de uma base de dados em SQL Azure é de 9.99$.

Dependendo da aplicação esta pode ser uma opção demasiado cara e, nesse caso, a opção por juntar todos os dados dos clientes numa única base de dados pode ser a opção mais correcta.

Tudo numa base de dados

Figura 2.2 – Todos os clientes numa única base de dados

Tanto uma opção como a outra têm problemas de escalabilidade. Ter todos os dados numa única base de dados, além do limite de 50Gb para dados pode ter problemas de capacidade de processamento. Ter uma base de dados por cliente pode ser bastante dispendioso ou, no limite, os 50Gb de armazenamento podem não chegar para um cliente que tenha muita informação para armazenar.

Sharding

Para resolver este problema era necessário recorrer à implementação manual de sharding. Sharding é um padrão que permite aumentar a escalabilidade e a performance de grandes bases de dados. Aplicar o padrão a uma base de dados significa "partir" essa base de dados em pedaços mais pequenos e distribui-los por vários servidores de modo a obter escalabilidade. A cada pedaço resultante chamamos de shard.

Neste padrão são as linhas das tabelas que são divididas pelos vários servidores. Uma outra opção seria realizar o particionamento vertical. No particionamento vertical os dados são separados por distribuição de tabelas completas entre os servidores. Meter a tabela de clientes num servidor e a tabela de vendas noutro servidor seria um exemplo de partição vertical. A partição de dados por valor (sharding) permite atingir maior escalabilidade porque, por exemplo, não está limitada ao número de tabelas existentes na aplicação e permite a divisão de tabelas com mais acessos.

Figura 2.3 – Sharding de uma base de dados

Nas aplicações multi-tenant podemos começar com uma única base de dados e, à medida que o volume de carga vai crescendo, usar o padrão de sharding para distribuir os dados dos clientes por várias bases de dados.

Figura 2.4 – Sharding numa aplicação multi-tenant

Com SQL Azure Federations vamos poder usar este padrão com "muito pouco esforço" e sem ter downtimes.

SQL Azure Federations

Distribuição dos dados

Um conceito muito importante antes de entrarmos na arquitectura das SQL Azure Federations é a distribuição dos dados. Conforme pode verificar na figura seguinte existem dados centralizados, particionados e replicados.

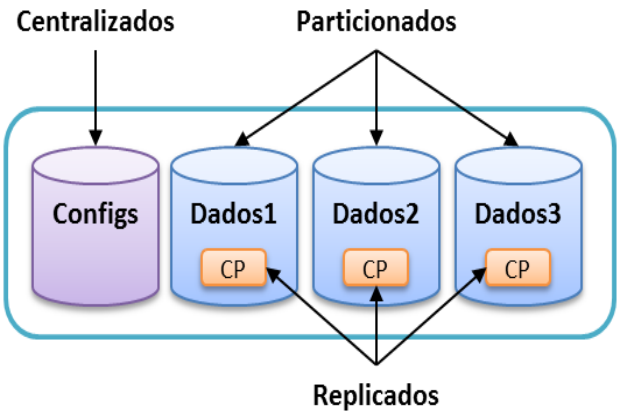


Figura 3.1 – Distribuição dos dados

Centralizados

Como exemplo de dados centralizados poderemos ter as tabelas de configurações. São dados sobre os quais não existem muitas operações de leitura e escrita e que estão disponíveis apenas num único local.

Replicados

Neste caso imagine a tabela de códigos postais dos CTT. São dados sobre os quais apenas existem operações de leitura (as escritas são possíveis mas raras) e que por questões de performance estão disponíveis em todas as bases de dados da aplicação. Dados replicados (reference data) são muito importantes relativamente a operações de join.

Particionados

Nesta categoria estarão a grande maioria dos dados onde cada "pedaço" de dados reside numa e apenas numa base de dados.

Arquitectura

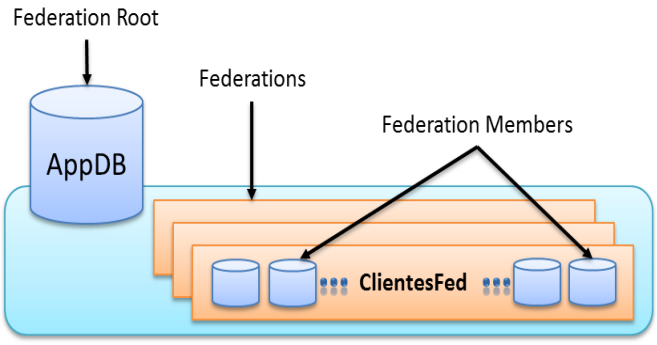


Figura 4.1 – Arquitectura das federations

Federation Root

Refere-se à base de dados central que contêm todas as informações sobre o particionamento dos dados.

Federation

Federations são os objectos que suportam a funcionalidade de distribuição dos dados entre as várias partições. Estes objectos representam o schema da federation e contêm a chave da distribuição, o tipo de dados e o estilo da distribuição. Na imagem anterior a base de dados AppDB representa uma base de dados com federations. Uma base de dados pode ter várias federations.

Federation Member (Shard)

A cada partição de dados dentro de uma federation chamamos federation member. Na imagem acima podemos observar uma federation para a tabela Clientes com vários membros. Cada federation member contêm uma parte dos dados e é suportado por uma instância de base de dados SQL Azure que fornece lhe capacidade de armazenamento e computação.

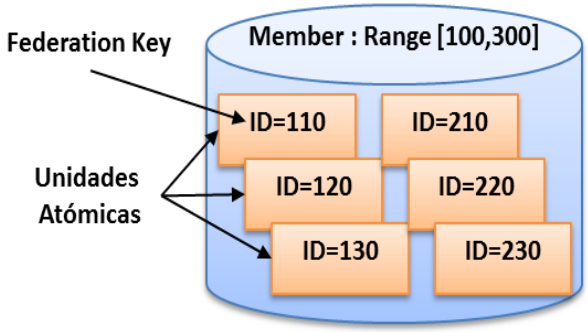


Figura 4.2 – Arquitectura das federations member

Federation Distribution Key

Os dados são distribuídos pelos vários federation members com base na federation key. Esta chave deve ser um campo que identifica univocamente cada partição possível para os dados. Na imagem anterior podemos verificar que o campo ID é a federation key.

Atomic Unit

Uma unidade atómica representa todo o conjunto de dados com a mesma federation key. Na imagem anterior podemos observar que todos os clientes com ID=100 são uma unidade atómica. Como o próprio nome indica, não é possível dividir uma unidade atómica, ou seja, dentro de uma federation, todas as tabelas com a mesma federation key ficam garantidamente no mesmo federation member.

Exemplo

Não é objectivo deste artigo ensinar a criar e a manipular federations mas para não defraudar alguns leitores mais curiosos vamos ver alguma ver um exemplo para que possamos ficar desde já familiarizados com alguma sintaxe.

```
-- Criar a base de dados root

CREATE DATABASE AppDB
(EDITION='business',MAXSIZE=50GB)


-- Criar uma federation

CREATE FEDERATION ClientesFed(ID BIGINT RANGE)
```

A federation foi criada com o tipo de dados BIGINT e um esquema de partição RANGE. Os tipos de dados possíveis na versão actual são INT, BIGINT, UNIQUEIDENTIFIER e VARBINARY(900).

```
-- Ligação à ClientesFed
USE FEDERATION ClientesFed (ID=0) WITH RESET,
FILTERING=OFF
```

Repare que a sintaxe utiliza a palavra reservada USE seguida de FEDERATION na sintaxe de ligação a uma federation. Temos que indicar qual a unidade atómica que nos queremos conectar, o que neste caso é indiferente porque acabámos de criar a federation e portanto todos ID são válidos.

```
CREATE TABLE Cliente(
ClienteID bigint,
Nome nvarchar[100],
primary key (ClienteID)
FEDERATED ON (ID = ClienteID)
```

A sintaxe de criação de uma tabela neste caso não tem nada de novo à excepção da última linha que permite anotar a tabela como fazendo parte da federation e indicar qual a coluna que serve de federation key é a coluna ClienteID. Repare que o tipo de dados dessa coluna tem que ser igual ao tipo de dados definido para a federation key.

```
CREATE TABLE Distrito(DistritoID
tinyint primary key, Nome nvarchar(128))
```

Se não for indicado o FEDERATED ON estamos a criar uma tabela com dados de referência que irá ser replicada por todos os federation members.

```
USE FEDERATION ClientesFed (ID = 110) WITH RESET,
FILTERING=ON
```

Temos agora um exemplo de uma ligação a uma unidade atómica. Neste exemplo estamos a criar uma ligação à unidade atómica com ID=110 dado que a opção FILTERING está ON.

SPLIT

Neste momento a única operação suportada é a operação de Split (divisão) com um parâmetro.

```
ALTER FEDERATION ClientesFed SPLIT (ID=200)
```

No futuro termos mais operações disponíveis tais como:

```
ALTER FEDERATION ClientesFed SPLIT AT
(ID=200,400,...,800)
ALTER FEDERATION ClientesFed MERGE AT (ID=200)
ALTER FEDERATION ClientesFed MERGE AT
(ID=200,400,...,800)
```

Vamos então ver como é feita a divisão de um federation member de modo a que o sistema não fique indisponível e a operação seja completamente transparente para o utilizador.

A operação de split é executada em duas fases. A primeira fase tem como objectivo preparar a transição e a segunda fase será a realização da transição.

SPLIT is executed in 2 phases; first phase is executed synchronously and focuses on setting up the operation in motion. The second phase happens asynchronously and typically is where bulk of the time is spent.

Fase 1

Nesta primeira fase instanciadas duas novas base de dados para alojar, uma para cada parte dos dados. Essas bases de dados terão o estado SPLITTING na tabela sys.databases.

São criadas as entradas nas tabelas de metadata que irão suportar o progresso da operação (sys.dm_federation_operation*) e preparadas as operações de cópia dos dados.

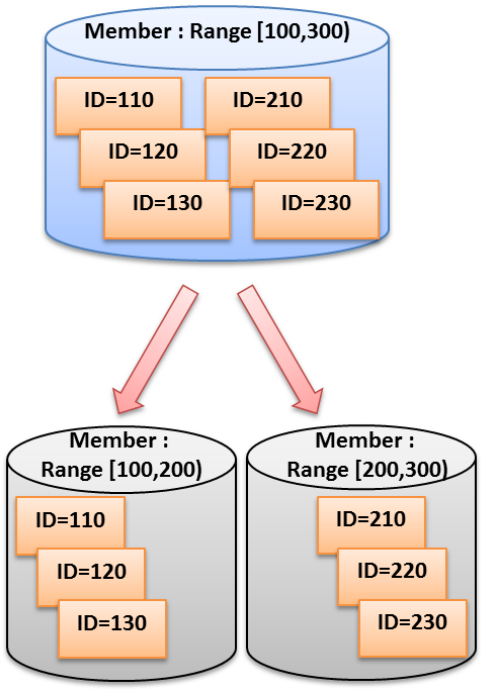


Figura 5.1 – Primeira fase do SPLIT

Enquanto esta fase corre a aplicação continua a funcionar sobre a base de dados original e além das operações de CRUD podem ainda ser realizar alterações ao schema.

Fase 2

É nesta fase que ocorre a transacção do schema e dos dados para as novas base de dados.

O Schema e todas as tabelas com dados replicados (tabela de códigos postais dos CTT por exemplo) são copiados para ambas as bases de dados novas. Os dados a particionar serão filtrados e copiados para a respectiva base de dados. Além das bases de dados novas os dados serão também copiados para as respectivas cópias secundárias.

Após todos os dados terem sido copiados dá-se a troca. A base de dados original passa para o estado offline quebrando assim todas as ligações existentes e imediatamente após isso as novas bases de dados passam para o estado online e começam a aceitar os novos pedidos de ligação.

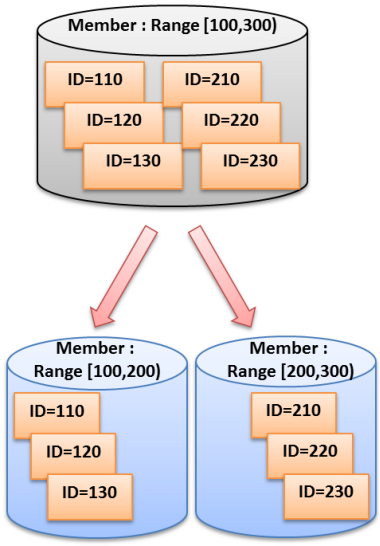


Figura 5.2 – Segunda fase do SPLIT

Repare que o downtime que existe é muito pequeno porque trata-se apenas de alterações na metadata e é mascarado pela retry logic que devemos ter nas aplicações que usam SQL Azure e portanto o erro não chega a ser visível aos utilizadores da aplicação.

Conclusão

A plataforma Windows Azure já permitia a construção de camadas de apresentação e camadas de lógica de negócio altamente escaláveis mas estava ligeiramente atrás no que respeita a camadas de acesso a dados sobre bases de dados. As camadas de acesso a dados sobre storage já são altamente escaláveis mas agora as SQL Azure Federations vêm trazer ao SQL Azure a mesma possibilidade sem que tenha que existir uma implementação “manual” do padrão de sharding.

Referências

Building Scale-Out Database Solutions on SQL Azure - Lev Novik

http://bit.ly/dUS2P6

Building Scalable Database Solutions Using Microsoft SQL Azure Database Federations - Cihan Biyikoglu

http://bit.ly/mSL5nW

Your Data in the Cloud - Cihan Biyikoglu

http://bit.ly/fnxTNn

Windows Azure

AUTOR

Escrito por Vítor Tomaz
Consultor independente na área das tecnologias de informação. Tem especial interesse por cloud computing, programação concorrente e segurança informática. É membro de algumas comunidades tais como Portugal-a-Programar, NetPonto, AzurePT, HTML5PT e GASP.

Enigmas do C#: Arrays

(continuação da página 22)

Resultado

System. ArrayTypeMismatchException: Attempted to access an element as a type incompatible with the array.

Explicação

Aquando do lançamento da plataforma .NET, a única forma ter uma coleção tipada, sem ter de a desenvolver, era o array.

A plataforma (e as suas linguagens – C# e Visual Basic) permite a conversão entre arrays de tipos referência. Para dois tipos referência A e B, se existir uma conversão de referências (implícita ou explicita) de A para B, a mesma conversão de referências existe de um array de tipo A[R] para um array do tipo B[R], em que R é uma especificação de dimensão (mas a mesma para ambos os tipos).

Por isso, é possível escrever o seguinte código:

```
Posição2D[] a1 = new Posição3D[5];
Posição[] a2 = a1;
```

Mas porque isto apenas é válido caso exista uma conversão de referências, o seguinte código não é válido:

```
int[] a3 = (int[])(new long[5]);
```

Esta relação é designada por covariância de arrays. A covariância de arrays, em particular, significa que um valor de um array do tipo A[R] pode ser uma referência para um array do tipo B[R], desde que exista uma conversão implícita entre B e A.

Devido a esta covariância de arrays, as atribuições de valor a elementos de arrays de tipos referência inclui uma validação em tempo de execução para garantir que o valor atribuído ao elemento do array é de um tipo permitido. Se tal não acontecer, será lançada uma

System.ArrayTypeMismatchException

Conclusão

Pelo demonstrado, a passagem de arrays entre interfaces programáticas (APIs) é desaconselhada (e desnecessária após a versão 2.0 da plataforma).

A utilização de arrays deve ser mantida privada ou em situações em que a sua errada manipulação não pode levar à inconsistência dos sistemas, como é o caso de leitura ou escrita de streams.

Para os restantes casos, deve ser usada a mais simples interface que satisfaça os requisitos. Por exemplo, se apenas se pretende enumerar os elementos, IEnumerable<T> é o suficiente. Se ICollection<T> for suficiente, não se deve usar IList<T>.

Ligações

C# Reference http://bit.ly/tfDPKa

Covariance and Contravariance in C#, Part Two: Array Covariance http://bit.ly/sicYL9

Covariância e Contravariância em Genéricos http://bit.ly/sMiv5M

AUTOR

Escrito por Paulo Morgado
É licenciado em Engenharia Electrónica e Telecomunicações (Sistemas Digitais) pelo Instituto Superior de Engenharia de Lisboa e Licenciado em Engenharia Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Pelo seu contributo para a comunidade de desenvolvimento em .NET em língua Portuguesa, a Microsoft premeia-o com o prémio MVP (C#) desde 2003. É ainda co-autor do livro "LINQ Com C#" da FCA.

COLUNAS

VISUAL (NOT) BASIC – Um pouco mais de Windows Phone 7

CoreDump – Free as in Beer

Um pouco mais de Windows Phone 7

Em uma edição anterior, na edição número 27, foi abordado o desenvolvimento VB.NET para Windows Phone 7 (WP7) na coluna, de uma forma básica, como que a fornecer o essencial para acesso ao meio, começar a desenvolver algumas brincadeiras, despertar vontades e quem sabe, mudar opiniões.

Desta vez, e assumindo que o essencial foi absorvido na edição 27, gostaria de passar a apresentar acessos às funcionalidades e características mais pertinentes destes aparelhos no geral, e do sistema operativo em particular e não tanto na construção da aplicação em si.

Estas funcionalidades e características ajudam a produzir possível interesse do público na aplicação, por ter capacidade de interagir não só com o aparelho, mas com o meio que o rodeia, das mais variadas formas e para os mais variados fins.

Existem vários complementos externos às aplicações mas um dos mais preciosos terá de ser a possibilidade de fazer a aplicação interagir com os "tiles", ou os mosaicos do ecrã de espera (a Microsoft gosta de lhe chamar Start screen, consequentemente, os Start Tiles. Eu vou chamar só “tiles”).

É precioso na medida em que é para estes tiles que o utilizador tem de olhar praticamente cada vez que desbloqueia o telefone. Se a aplicação puder de alguma forma dar feedback através desses tiles, conseguimos chamar a atenção do utilizador sem precisar de correr a aplicação, ou de lá estar. É um local onde é suposto estarem todas as notificações de todos os sistemas dos quais se esperam notificações.

Para além disto, os tiles funcionam como atalhos, que nos levam directamente para determinada página da nossa aplicação. Tudo isto pode jogar a nosso favor.

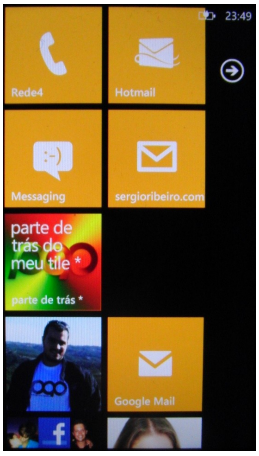
Vamos começar precisamente por aí.

Os assuntos abordados foram executados na versão 7.1 do Windows Phone SDK.

É possível que a maioria não esteja disponível na versão 7.0.

Integração com o ambiente

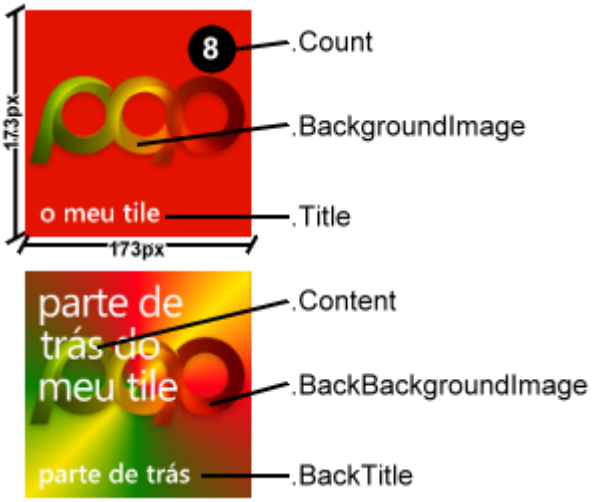
Tiles



A forma mais simples de interagir com o menu "mosaico", para além de alterar o tile da aplicação, é através da criação de "standard tiles".

Estes tiles possuem duas faces 173x173 que podemos usar para apresentar informação, da forma que nos for mais conveniente.

Podemos optar por utilizar apenas uma das faces, ou ambas. Neste último caso, a tile alterna da frente para trás de alguns em alguns segundos, automaticamente.



Um pouco mais de Windows Phone 7

ShellTile é a API responsável por a interacção com o Start Menu.

A propriedade "ActiveTiles" da ShellTile é a colecção que contém todos os tiles associados à aplicação e onde o índice 0 (zero, o primeiro) é sempre o tile da própria aplicação (usado para quando se cria um atalho para a aplicação no Start Menu).

Todos os outros na colecção, se alguns, são tiles secundários.

Assim, podemos alcançar e alterar a tile principal da seguinte forma:

```
Dim tile As New StandardTileData
tile.Title = "o meu tile"
tile.BackTitle = "aplicação dos tiles!"
meusTiles.Update(tile)
```

Com pouco mais, podemos utilizar a API para criar novos tiles, ou tiles secundários:

```
Dim tile As New StandardTileData
tile.Title = "o meu tile"
tile.BackTitle = "parte de trás"
tile.BackContent = "parte de trás do meu tile"
tile.Count = 8
tile.BackgroundImage = New
Uri("/imagens/tile.png", UriKind.Relative)
tile.BackBackgroundImage = New
Uri("/imagens/backtile.png", UriKind.Relative)
ShellTile.Create(New
Uri("/MainPage.xaml", UriKind.Relative), tile)
```

Os tiles secundários podem, não só, ser completamente personalizados, mas também podem apontar para uma outra página na aplicação, com ou sem parâmetros adicionais.

Imagine-se por exemplo um calendário onde poderíamos criar um tile por cada dia, e esse tile arrancasse o calendário naquele dia específico.

Se for necessário alterar um tile secundário, o método é em todo semelhante ao de alterar o tile principal, mas temos de referenciar um tile secundário.

Por exemplo, apontando para o último tile, no caso de termos apenas tile principal e um secundário:

```
Dim tiles As ShellTile = ShellTile.ActiveTiles.Last
Dim tile As New StandardTileData
tile.Title = "o meu tile *"
tile.BackTitle = "parte de trás *"
tile.BackContent = "parte de trás do meu tile *"
tile.Count = 0
tile.BackgroundImage = New
Uri("/imagens/tile.png", UriKind.Relative)
tile.BackBackgroundImage = New
Uri("/imagens/backtile.png", UriKind.Relative)
tiles.Update(tile)
```

Para utilizações mais específicas, também é possível programar uma actualização a um tile.

A actualização automática é possível com ShellTileSchedule, mas apenas é possível actualizar a imagem de fundo, a partir de um URL:

```
Dim sTS As New ShellTileSchedule
sTS.Recurrence = UpdateRecurrence.Interval
sTS.RemoteImageUri = New
Uri("http://www.portugal-a-programar.org/staff/
imagens/banners/pap_2011.png")
sTS.Interval = UpdateInterval.EveryHour
sTS.StartTime = DateTime.Now
sTS.Start()
```

Não é garantido que o tile actualize imediatamente, mesmo que a nossa programação de tempo seja para o alterar apenas uma vez, sem repetição.

Estas actualizações são feitas em grupos e com frequência pré-determinada, e não dependem da aplicação.

Pode demorar até uma hora para que uma actualização surta efeito.

Em baixo, o resultado da programação programada do código exemplo:

Para actualizar mais do que a imagem de fundo, seria necessário recorrer a Push Notifications, que não vou cobrir no artigo.

Este facto força-me a omitir outras funcionalidades interessantes como a Toast Notification.

À lupa:

StandardTileData

- **BackBackgroundImage**

  Imagem de fundo da face traseira

- **BackContent**

  Texto a apresentar no corpo da face traseira

- **BackgroundImage**

  Imagem de fundo da face dianteira

- **BackTitle**

  Texto a apresentar no título da face traseira

- **Count**

  Número a apresentar no contador da face dianteira

- **Title**

  Texto a apresentar no título da face dianteira

ShellTileSchedule

- **Interval**

  Define a frequência da actualização. Definida por um Enum

- **MaxUpdateCount**

  Número máximo de actualizações a efectuar, desde o último Start. Zero para actualizar indefinidamente, até ordem de Stop.

- **Recurrence**

Define se a actualização é única ou se deverá repetir

- **RemoteImageUri**

  Endereço da imagem a utilizar para a actualização do tile

- **Start**

  Inicia a actualização programada

- **StartTime**

  Define a data a partir da qual o programa tem efeito

- **Stop**

  Interrompe e termina a actualização programada

Resources

Para manter a coerência, durante a execução da aplicação, é importante existir um local a partir do qual se podem consultar uma série de opções transversais ao aparelho.

A colecção Resources, em Application.Current.Resources pode não só ser usada como repositório temporário de alguns parâmetros que queremos ter disponíveis em qualquer página, mas também pode ser usada para consultar parâmetros do aparelho em si, ou do sistema operativo, bastante importantes para a integração da aplicação.

Um bom exemplo são por exemplo as cores e os brushes que estão a ser utilizados no sistema.

É a partir desta colecção que se pode consultar, entre dezenas de outros parâmetros, a “Accent” color que o utilizador escolheu e se o tema é escuro ou claro.

A colecção é composta por chaves únicas e o seu respectivo valor.

```
Application.Current.Resources.Add
("urlPAP", New Uri
("http://www.portugal-a-programar.org"))
MessageBox.Show(DirectCast
Application.Current.Resources("urlPAP"),
Uri).AbsoluteUri)
```

Um pouco mais de Windows Phone 7

```
Application.Current.Resources.Add
("parametro1", "teste!")

MessageBox.Show
(Application.Current.Resources
("parametro1"))
```

Tanto a chave como o valor são do tipo "Object", para uma elevada versatilidade, pelo que é sempre boa ideia proceder a casts ou conversões de tipo para trabalhar com a colecção.

As chaves correspondentes às cores do tema mais pertinentes são:

Tipo SolidBrush

- PhoneAccentBrush (brush com cor dos destaques, igual à cor dos tiles. Conhecida como a "accent color")
- PhoneForegroundBrush
- PhoneBackgroundBrush

Tipo Color

- PhoneAccentColor (cor dos destaques, igual à cor dos tiles. Conhecida como a "accent color")
- PhoneBackgroundColor
- PhoneForegroundColor

Tipo Double para tamanho de fonte

- PhoneFontSizeSmall
- PhoneFontSizeNormal
- PhoneFontSizeMedium
- PhoneFontSizeMediumLarge
- PhoneFontSizeLarge
- PhoneFontSizeExtraLarge
- PhoneFontSizeExtraExtraLarge
- PhoneFontSizeHuge

Tipo Visibility

- PhoneDarkThemeVisibility (no tema light, assume Visibility.Collapsed)
- PhoneLightThemeVisibility (no tema dark, assume Visibility.Collapsed)

Tipo Double para opacidade

- PhoneDarkThemeOpacity (no tema dark, assume 1.0, opaco)
- PhoneLightThemeOpacity (no tema light, assume 1.0, opaco)

Para as utilizar basta usar uma das chaves expostas no dicionário Resources.

Será devolvido o tipo especificado, com a informação pretendida.

Por exemplo:

```
Application.Current.Resources_
("PhoneDarkThemeOpacity")
```

DeviceStatus

Para além dos parâmetros de cor, tamanho e outros, que nos permitem a integração visual da nossa aplicação, é necessário também conhecer o aparelho para determinar de que forma se deverá adaptar e como se deverá comportar.

Muita da informação básica do aparelho encontra-se em Microsoft.Phone.Info

```
AppliMicrosoft.Phone.Info.Device.
Status.DeviceFirmwareVersion
Microsoft.Phone.Info.DeviceStatus.
DeviceManufacturer.Microsoft.Phone.
Info.DeviceStatus.DeviceName
Microsoft.Phone.Info.DeviceStatus.PowerSource.
ToString
```

Depois, para além dessa informação geral, existe também informação em âmbitos mais reduzidos, por exemplo, a disponibilidade ou o "nome" da ligação de dados, que nos remete a Microsoft.Phone.Net:

```
Microsoft.Phone.Net.NetworkInformation.
NetworkInterface.NetworkInterfaceType.ToString
Microsoft.Phone.Net.NetworkInformation.
NetworkInterface.GetIsNetworkAvailable
```

Ou a disponibilidade física de alguns sensores, em Microsoft.Devices.Sensors, por exemplo:

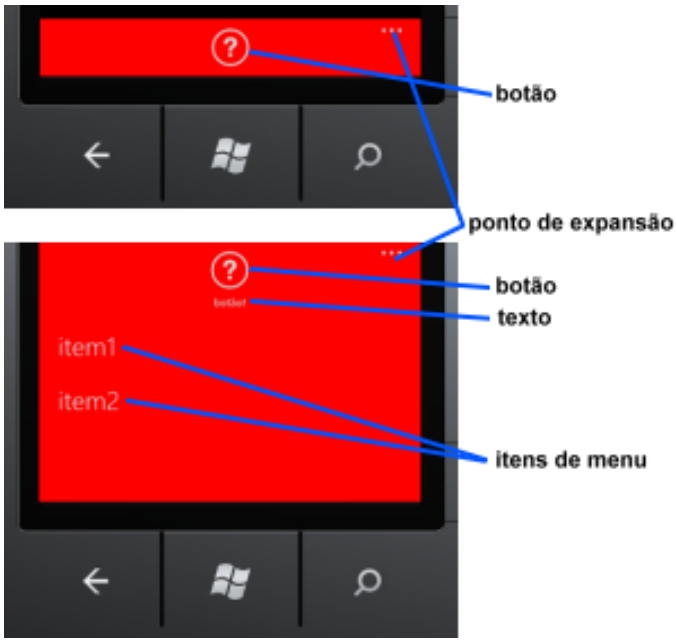
```
Microsoft.Devices.Sensors.Compass.IsSupported
Microsoft.Devices.Sensors.Gyroscope.IsSupported
```

Entre outros.

ApplicationBar

Todas as aplicações têm a possibilidade de usufruir de um menu configurável, e de comportamento e contornos transversais ao sistema operativo.

Este componente é de extrema importância quando a nossa aplicação compreende uma maximização de espaço útil em paralelo com um elevado número de funcionalidades.



A ApplicationBar é constituída por 4 Botões com ícone e “n” Itens em lista vertical.

Não é obrigatório utilizar ambos, assim como não é obrigatório utilizar apenas um em detrimento do outro: o seu uso está ao nosso critério.

Podem existir um máximo de 4 botões, e os botões podem ou não utilizar uma pequena descrição.

Finalmente, podem assumir qualquer cor, qualquer nível de opacidade e ainda suportam dois modos de recolha: o normal, onde os botões estão visíveis, e o minimizado onde recolhida apenas expõe o ponto de expansão.

```
ApplicationBar.BackgroundColor = Colors.Red
ApplicationBar.ForegroundColor = Colors.LightGray
ApplicationBar.Opacity = 1D
ApplicationBar.Buttons.Clear()
ApplicationBar.MenuItems.Clear()
Dim appBot As New ApplicationBarIconButton
appBot.Text = "botão!"
appBot.IconUri = New Uri("/imagens/perg.png",
UriKind.Relative) 'build como content
ApplicationBar.Buttons.Add(appBot)
```

```
ApplicationBar.MenuItems.Add
(New ApplicationBarMenuItem("item1"))
ApplicationBar.MenuItems.Add
(New ApplicationBarMenuItem("item2"))
```

A ApplicationBar pode ser pré-configurada via XAML.

No código acima, os constituintes são adicionados em runtime a uma ApplicationBar adicionada via XAML.

Não está incluído qualquer handling dos eventos necessários para o funcionamento, como o click.

Basta acrescentar um handler por cada botão ou item, por exemplo:

```
AddHandler appBot.Click, AddressOf BotaoTocado
```

Caso não seja adicionada via XAML, a propriedade ApplicationBar estará nula até que uma instância de ApplicationBar lhe seja atribuída.

As imagens a utilizar como ícones dos botões, deverão ser PNG, e deverão ser marcadas com a Build Action “Content” para que possam ser utilizadas.

IsolatedStorage

Em todas as aplicações, por mais pequenas que sejam, é muito provável que exista uma situação em que é ideal armazenar algum tipo de informação de forma a que possa ser recuperada independentemente da aplicação terminar ou do aparelho ser desligado.

```
'Preparar o “túnel” de ligação com a nossa
'isolatedstorage
Dim ISTF As IsolatedStorageFile = _
IsolatedStorageFile.GetUserStoreForApplication()
MessageBox.Show(“A quota para a aplicação é de “
                & ISTF.Quota.ToString & “ bytes”)

ISTF.CreateDirectory("pasta")

'Escrever linhas para um ficheiro na
'isolatedstorage
Using SW As New IO.StreamWriter(New
        IsolatedStorageFileStream("pap.txt",
        FileMode.Create, FileAccess.Write, ISTF))
    SW.WriteLine("linha1")
    SW.WriteLine("linha2")
End Using
'Ler dados a partir de um ficheiro na
'isolatedstorage
Using SR As New IO.StreamReader
               (New IsolatedStorageFileStream
               ("pasta\pap.txt",
               FileMode.OpenOrCreate,
               FileAccess.ReadWrite, ISTF))
```

```
    While Not SR.EndOfStream
        MessageBox.Show(SR.ReadLine)
    End While
End Using
```

A IsolatedStorage responde desde essa necessidade, até à necessidade de uma aplicação que apenas implique pesadas operações com ficheiros, ou que exija um elevado grau de organização no seu sistema de dados.

Podemos encarar a IsolatedStorage como uma pasta que reside algures no aparelho, não interessa onde, e que está de alguma forma relacionada com uma aplicação em específico.

Neste local, é nos possível criar pastas, ficheiros, copiar ficheiros, consultar datas de modificação, virtualmente todas as operações com ficheiros que teríamos em um outro sistema de ficheiros, incluindo listagens. Assim como qualquer outro sistema de ficheiros, a IsolaredStorage também tem um limite. Esse limite está pré-definido para cada aplicação, com uma quota.

Eis o exemplo de uma operação com ficheiros que engloba os essenciais:

Quando a quota por defeito não é suficiente, é possível pedir um aumento da quota, e a concessão é da exclusiva gestão do sistema operativo.

Se for possível, o método devolve True e a quota é aumentada. Caso contrário, devolve False.

Se o pedido para aumentar a quota for de valor inferior à actual quota, é disparada uma excepção.

As quotas são expressas em bytes.

ISTF.IncreaseQuotaTo(2000000)

Se a aplicação for desinstalada, a IsolatedStorage associada é destruída.

Background Transfers

A propósito das IsolatedStorages, sendo esses os locais onde se devem operar ficheiros, é importante referir as Background Transfers.

As Background Transfers são unidades de um serviço de transferências de ficheiros existente na framework, que utilizam os meios e circunstâncias à sua disposição, quer sejam ligações de dados celulares ou WiFi, bateria ou alimentação externa para descobrir a melhor forma e a melhor altura para transferir ficheiros.

São independentes da aplicação, e funcionam sempre. Por esta razão, estão aptas para analisar a situação do telefone e decidir, por exemplo, que não deverá transferir um ficheiro de elevadas dimensões sem estar ao alcance de uma WiFi, ou ligado à uma fonte de energia externa.

O BackgroundTransferService recebe BackgroundTransferRequest, e cada um destes pedidos pode implicar o download de um ficheiro para a IsolatedStorage, ou o upload de um ficheiro da IsolatedStorage.

Em ambos os casos, é importante que os handlers TransferProgressChanged e TransferStatusChanged sejam registados para que se consiga dar feedback ao utilizador.

As transferências são identificadas inequivocamente por o seu RequestID, que é automaticamente gerado a cada pedido.

```
Dim ISTF As IsolatedStorageFile =
   IsolatedStorageFile.GetUserStoreForApplication()
If Not ISTF.DirectoryExists("shared\transfers")
Then
    ISTF.CreateDirectory("shared\transfers")
End If
Dim BGT As New Microsoft.Phone.BackgroundTransfer.
    BackgroundTransferRequest(New Uri
    ("http://www.portugal-a-programar.org/revista-
    programar/edicoes/download.php?e=31&t=forum"))

AddHandler BGT.TransferProgressChanged,
                    AddressOf repProgresso
AddHandler BGT.TransferStatusChanged,
                    AddressOf repEstado

BGT.DownloadLocation = New Uri("shared/transfers/
     revista.pdf", UriKind.RelativeOrAbsolute)
Microsoft.Phone.BackgroundTransfer.
BackgroundTransferService.Add(BGT)

Private Sub repProgresso(ByVal sender As Object,
ByVal e AsMicrosoft.Phone.BackgroundTransfer.
BackgroundTransferEventArgs)
    ' e.Request.BytesReceived
    ' e.Request.TotalBytesToReceive
    ' acções de actualização
End Sub

Private Sub repEstado(ByVal sender As Object, ByVal
e As Microsoft.Phone.BackgroundTransfer
.BackgroundTransferEventArgs)
ApplicationTitle.Text =
e.Request.TransferStatus.ToString
```

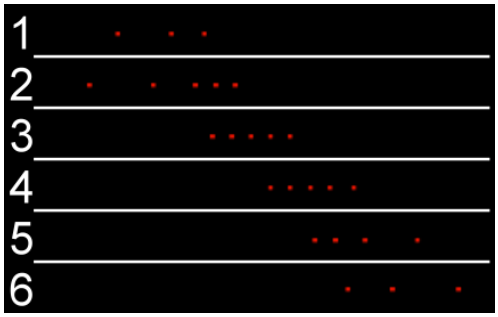
Quanto uma transferência completa, não desaparece. Esta terá que ser removida por aplicação.

Por esta razão, uma aplicação que lide com transferências desta natureza, tem a responsabilidade de verificar as filas de transferências de cada vez que arrancar, para determinar se a transferência precisa de alguma acção ou se precisa de ser removida.

```
For Each BGTR As Microsoft.Phone.
    BackgroundTransfer.BackgroundTransferRequest
In Microsoft.Phone.BackgroundTransfer.
    BackgroundTransferService.Requests

Select Case BGTR.TransferStatus
   Case Microsoft.Phone.BackgroundTransfer.
              TransferStatus.Completed
   Microsoft.Phone.BackgroundTransfer.
       BackgroundTransferService.Remove(BGTR)
   MessageBox.Show("Transferencia com RequestID="
           & BGTR.RequestId & " terminou enquanto
           não estava na aplicação!")
         'outras acções, como dar feedback ao UI
         para saber que terminou
Case Microsoft.Phone.BackgroundTransfer.
            TransferStatus.Paused,
   Microsoft.Phone.BackgroundTransfer.
          TransferStatus.Transferring,
          Microsoft.Phone.BackgroundTransfer.
          TransferStatus.WaitingForExternalPower,
        'outras acções, como feedback ao UI para
        'continuar
End Select

Next
```

Indeterminate ProgressBar

Em algumas operações onde não é possível prever o tempo de execução ou não se possa quantificar o progresso as Indeterminate ProgressBars são importantes porque oferecem um feedback visual ao utilizador, que indica que alguma coisa está a executar e fará consequentemente com que aguarde.



Na verdade, uma Indeterminate ProgressBar é uma progressbar comum, mas com a propriedade IsIndeterminate verdadeira.

```
Dim IPB As New ProgressBar
IPB.IsIndeterminate = True
ContentPanel.Children.Add(IPB)
```

Basta utilizar a visibilidade para mostrar ou ocultar a barra.

Leitura do exterior

Acelerómetro, giroscópio e bússola

Os aparelhos WP7 estão, tipicamente, munidos de componentes que lhe permitem perceber como é que estão a ser orientados e movimentados no espaço físico.

Estes três componentes funcionam todos de forma semelhante, mas dão leituras distintas.

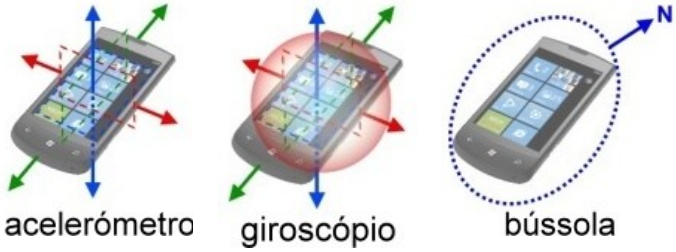
Estas leituras podem ser utilizadas em separado, ou em conjunto para se complementarem em determinadas aplicações.

A classe Motion do namespace Sensors utiliza as leituras destes componentes para calcular movimentos e orientações com mais precisão (por exemplo para realidade aumentada).

```
If Sensors.Accelerometer.IsSupported Then
      Dim Accel As New Sensors.Accelerometer
      Accel.TimeBetweenUpdates =
                  New TimeSpan(0, 0, 0, 0, 100)
      AddHandler Accel.CurrentValueChanged,
            AddressOf dadosAccel
      Accel.Start()
End If
```

Cada sensor é preparado de maneira semelhante, configurando qual o intervalo de tempo entre as actualizações (ou seja, qual a frequência com que a leitura é feita), adicionando o handler para reportar as leituras, e no caso da bússola um handler que dispara caso se detecte que a precisão da bússola está afectada de forma a que seja necessário calibrar.

A captura começa assim que o método Start for chamado, e termina quando for chamado o Stop.



Eis um exemplo de obtenção de dados da bússola:

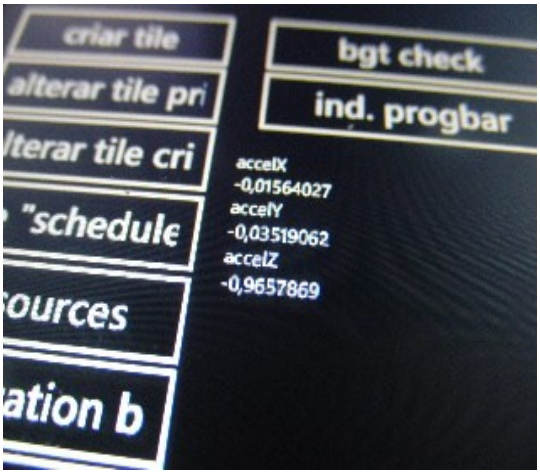
```
Private Sub dadosComp(ByVal sender As Object,
ByVal e As Sensors.SensorReadingEventArgs(Of _
Sensors.CompassReading))

Deployment.Current.Dispatcher.BeginInvoke(
    Sub()
   'caso a precisão da leitura esteja abaixo de
   '15, as letras do botão ficam brancas (OK)
   'as letras ficam vermelhas se for disparado o
   'evento Calibration. Para o exemplo, esse
   'handler colocava as letras do botão vermelhas
   'o que indica que a calibração era necessária
  If e.SensorReading.HeadingAccuracy < 15 Then
      Button8.Foreground = New SolidColorBrush
                                   (Colors.White)
      Dim precisão As Double =
            e.SensorReading.HeadingAccuracy
      Dim orientacaoMag As Double =
            e.SensorReading.MagneticHeading
      Dim magnetometro As _
        Microsoft.Xna.Framework.Vector3 =
        e.SensorReading.MagnetometerReading
      Dim verdadeiraOri As Double =
        e.SensorReading.TrueHeading
    End Sub
End Sub)
```

Do giroscópio:

```
Private Sub dadosGyro(sender As Object, e As _
Sensors.SensorReadingEventArgs(Of _
Sensors.GyroscopeReading))
    Deployment.Current.Dispatcher.BeginInvoke(
    Sub()
        Dim rrX As _
         Microsoft.Xna.Framework.Vector3 =
         e.SensorReading.RotationRate
    End Sub)
End Sub
```

E do acelerómetro:

```
Private Sub dadosAccel(ByVal sender As Object,
ByVal e As Sensors.SensorReadingEventArgs(Of _
Sensors.AccelerometerReading))
    Deployment.Current.Dispatcher.BeginInvoke(
    Sub()
        Dim acel As _
            Microsoft.Xna.Framework.Vector3 =
            e.SensorReading.Acceleration
    End Sub)
End Sub
```



Leitura do acelerómetro

Leitura da bússola

Não tive oportunidade de utilizar um aparelho com giroscópio para apresentar leituras.

É necessário o invoke, pois não é garantido que as leituras voltem no thread da UI.

As leituras são adaptadas ao tipo de sensor e terão de ser interpretadas no seu contexto, o que requer conhecimentos matemáticos no campo da física.

Vector3 é um tipo de dados que armazena essencialmente 3 valores, XY e Z, representando consequentemente um vector no espaço.

A classe Vector3 pode ser encontrada no namespace

Microsoft.Xna.Framework.

Cada leitura é acompanhada por uma estampa de tempo (SensorReading.Timestamp).

Estes dados podem ser necessários para cruzar com leituras de outros sensores no decorrer de um intervalo, e determinar rotas ou melhorar médias.

GPS

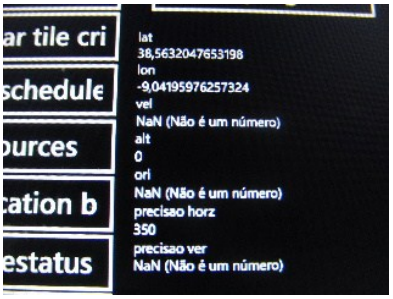
Os dados da geo-localização, através de satélites de posicionamento, não é considerado um sensor, tendo sido abrigada no namespace System.Device.Location, classe GeoCoordinateWatcher.

É relativamente fácil começar a capturar a localização, bastando apenas registar um handler para as mudanças de posição.

```
Dim GPS As New
System.Device.Location.GeoCoordinateWatcher
AddHandler GPS.PositionChanged,
      AddressOf dadosGPS
GPS.Start()
```

O handler é parecido com os dos sensores:

```
Private Sub dadosGPS(ByVal sender As Object,
ByVal e As System.Device.Location.
GeoPositionChangedEventArgs
(Of System.Device.Location.GeoCoordinate))
    Deployment.Current.Dispatcher.BeginInvoke(
    Sub()
        Dim latitude As Double =
            e.Position.Location.Latitude
        Dim longitude As Double =
            e.Position.Location.Longitude
        Dim velocidade As Double =
            e.Position.Location.Speed
        Dim altitude As Double =
            e.Position.Location.Altitude
        Dim orientacao As Double =
            e.Position.Location.Course
        Dim precisaoVert As Double =
            e.Position.Location.VerticalAccuracy
        Dim precisaoHorz As Double =
            e.Position.Location.HorizontalAccuracy
    End Sub)
End Sub
```



Leitura do GPS

A velocidade é dada em metro por segundo, orientação em graus relativos a norte, altitude e precisões em metros.

Cada relatório de posição também é acompanhado por uma estampa de tempo (Location.Timestamp).

Em suma

Apenas a criatividade em conjunto com a utilidade ditam o sucesso de uma aplicação, mas é importante ter conhecimento das possibilidades que nos são oferecidas.

Existem muito mais pequenas e grandes funcionalidades que merecem ser exploradas: apenas foquei as que considero de maior interesse e que podem ser integradas num maior número de aplicações, tratem elas do que tratarem.

AUTOR

Sérgio Ribeiro, curioso e auto-didacta com uma enorme paixão por tecnologias de informação e uma saudável relação com a .NET framework.

Moderador do quadro de Visual Basic.NET na comunidade Portugal@Programar desde Setembro de 2009.

Alguns frutos do seu trabalho podem ser encontrados em http://www.sergioribeiro.com

Free as in Beer

Open Source Software (OSS), free as in speach, not as in beer...

Por muito altruísta que pretenda ser, a verdade é que muito do OSS proliferou por ser "free as in beer"... E se hoje em dia soluções OSS têm um grau de maturidade que ninguém questiona, a verdade é que muitas organizações continuam a optar por OSS não pelas suas caraterísticas, não por ser livre mas sim por ser grátis!

Este erro comum de confundir software livre com software gratuito continua a existir. Um excelente exemplo é a recente adoção por parte do governo português de OSS precisamente por não querer renovar licenças de software proprietário. Ou seja, um caso claro e assumido de escolha de OSS por ser "free as in beer"...

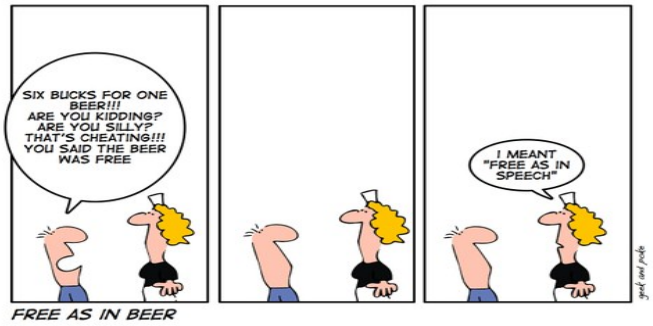


Imagem de Geek and Poke.

Se nos anos 70 e 80 alguém tinha um problema num sistema que havia sido comprado a um concorrente da IBM, estava em maus lençóis e tinha de responder "Why not IBM?". A IBM dominava a informática empresarial e todos os departamentos de IT eram IBM. Os que não eram, quando havia um problema tinham de explicar à administração porque não haviam comprado IBM, mesmo que a solução adotada servisse melhor a organização. Nos anos 90 o mesmo se começou a passar com a Microsoft. Hoje em dia há situações onde essa questão se coloca ao nível do OSS. Em particular em soluções maduras como sistemas operativos para servidores ou servidores web, é comum observar alguém questionar porque não se optou por uma distribuição de Linux ou pelo Apache.

Atualmente as organizações tecnologicamente maduras vêm o OSS como "*free as in speach*" e não como "*free as in beer*". Essas organizações compreendem a diferença e, acima de tudo, fazem escolhas de software conscientes, colocando lado-a-lado várias soluções proprietárias e open source para medir a sua eficácia na resolução dos seus problemas. Esta abordagem agnóstica e independente permite às organizações efetuarem escolhas baseadas no mérito e não por outros critérios mais duvidosos aos olhos de um técnico competente.

As organizações imaturas continuam, e acredito que continuarão, a escolher o OOS por o conceberem como gratuito. Mas quando a escolha de um produto se foca no preço de aquisição do mesmo, corre-se o risco da escolha não se adequar à realidade ou às necessidades da organização. Escolher OSS, ou um software proprietário gratuito, nesta base revela uma falha enorme ao nível da gestão, fazendo com que a organização incorra em custos não previstos e alguns dissabores. Esse custos são, muitas vezes, operacionais e escondidos, ou varrido para debaixo do tapete. Por exemplo, quando alguém tem de fazer uma quantidade de trabalho manual adicional para completar uma tarefa que deveria ser totalmente automatizada.

Por esta altura muitos estão a pensar "**mas isso é OSS, qualquer um pode mexer no código e adequá-lo às necessidades da empresa**". Na verdade não. E porquê? Porque o fator de escolha foi o preço, e uma organização que escolhe assim não está disposta a pagar a uma empresa ou a um profissional para adaptar a solução às necessidades da organização. Mesmo que isso fosse bastante mais barato do que todo o custo operacional adicional que as falhas implicam. Não nos podemos esquecer que esta é uma organização tecnologicamente imatura, se não o fosse, teria efetuado a sua escolha de forma consciente.

Felizmente as organizações têm evoluído e os departamentos de IT têm hoje competências que não tinham à 10 anos atrás. Isso faz com que cada vez mais os departamentos de IT sejam integrados na estratégia de negócio das organizações e consigam responder com soluções tecnológicas cada vez melhores.

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.

COMUNIDADES

AzurePT - Multiplas identidades no seu WebSite usando Windows Azure AppFabric ACS

NetPonto - Biztalk Server - Princípios Básicos dos Mapas

Multiplas identidades no seu WebSite usando Windows Azure AppFabric ACS

Introdução

Uma das questões que actualmente mais se tem falado é o facto de existirem demasiadas identidades actualmente na internet, pois quem não passou até hoje pelas seguintes questões quando chega a um website, “será que já me registei neste website?”, ”qual será o utilizador que utilizei no registo?”, para já não falar na dificuldade de lembrar da palavra-chave que foi utilizada para o mesmo. Com todas estas questões o que em muitas situações acontece é que começamos a ter quase um padrão para o nome do utilizador e palavra-chave que utilizamos no registo nos websites, o que não é claramente uma boa prática de segurança, mas é isso ou andar com uma aplicação em que descrevemos todos os websites em que nos registamos, juntamente com o nome do utilizador e palavra-chave utilizada. Isto é sempre uma “mina de ouro” porque se alguém apanhar esta informação estamos completamente agarrados, porque nem sequer sabemos todos os locais em que determinada identidade existe.

Também bastante problemático em termos de segurança é que muitas vezes nós registamos os nossos dados num determinado website, sem utilizar os dados reais porque em primeiro lugar, não sabemos exactamente o que vai acontecer no mesmo, nem o que o próprio website vai fazer com os mesmos. Isto é importante porque quantos de nós já não vimos aplicações em que as próprias palavras-chave se encontravam completamente acessíveis e desencriptadas na base de dados?

Imaginemos o seguinte cenário,

“Enquanto me encontrava a navegar na internet deparei-me com um website muito interessante que tem uma rádio online e músicas muito recentes, aquilo que eu fiz imediatamente foi registar-me. Após isso descobri que existe uma subscrição que por um pequeno valor consigo ter acesso a um conjunto maior de músicas e ainda mais recentes, e a possibilidade de efectuar o descarregamento de algumas para dispositivos como iPod, PC, etc. Para efectuar essa subscrição necessito de colocar os dados relativos ao meu cartão de crédito no site (número e código de segurança), para que eles possam todos os meses debitar o meu cartão de forma a pagar a subscrição.”

Este é um cenário relativamente comum de acontecer, a questão é que todos nós já estamos devidamente preparados para dizer logo que essa não parece uma boa opção por uma questão simples, a confiança. O website pode ser excelente, pode ter música excelente mas esse é o seu negócio, mas quando falamos de questões relacionadas com pagamentos ou outras questões sensíveis já pensamos melhor e podemos inclusivamente deixar de utilizar o mesmo se não existir alternativa, mais uma vez porque a confiança é absolutamente crucial quando falamos de questões relacionadas com cartões de crédito, identidades, e outras informações sensíveis. A situação seria completamente diferente se o website nos desse a possibilidade de não colocar os dados directamente no mesmo, mas sim redireccionar-nos para um website de um banco, ou PayPal ou outro qualquer sistema no qual eu tenha uma elevada confiança para tratar os meus dados.

Por tudo isto cada vez mais são as soluções e websites que em vez de terem a sua própria gestão de identidades têm relações de confiança com fornecedores de identidades bem conhecidos, como por exemplo o LiveId, GoogleId, Twitter, Facebook, entre outros. Mas a questão que se coloca é como conseguimos nós criar um website que tem exactamente o mesmo tipo de funcionamento, e não querendo pedir demasiado, como é que o conseguimos efectuar com o mínimo de trabalho possível?

Conceitos base

Para que consigamos responder a esta questão necessitamos de em primeiro lugar compreender os conceitos que se encontram na base desta solução. E para iniciarmos vamos olhar para o conceito de IdP – Identity Provider, que é uma aplicação/solução que tem como função principal gerir identidades. Neste caso como fornecedores de identidade teríamos o LiveId, GoogleId, TwitterId, FacebookId, etc. Uma das vantages de utilizarmos estes fornecedores de identidades é que a maioria das pessoas já têm um deles ou mais, e isso irá permitir que o utilizador para poder utilizar a nossa solução não passe a ter mais uma identidade digital.

Outro conceito muito importante é o **RP - Relaying Party** que representa a aplicação/solução que confia a parte de validação das identidades num **IdP**. Quando existe este tipo de relação de confiança entre um **RP** e um **IdP**, diz-se que estamos perante uma Identidade Federada, uma vez que não é a solução que trata das identidades mas sim, confia/federa essa responsabilidade num **IdP**.

Também bastante importante é o conceito de **FG - Federation Gateway** que básicamente nos define quais as identidade que a nossa solução vai suportar, e que irá permitir ao utilizador seleccionar qual o **IdP** que deseja utilizar e efectuar o reencaminhamento para esse mesmo **IdP** sempre que for necessário identificar um utilizador desse tipo.

Para finalizar existe um outro conceito que é o Tipo de Federação (Federation Type) que define a forma como o

processo de routeamento entre o **FG** e o **IdP** é efectuado, sendo que o mesmo poderá ser efectuado de duas formas:

- Passivo – Necessita de redireccionamento. Exemplo seria um browser, em que nós quando vamos ao site da XBOX, quando pedimos para fazer o login, somos imediatamente redireccionados para o site do LiveId, e apenas quando formos correctamente validados pelo mesmo retornamos ao site da XBOX.

- Activo – Não necessita de redireccionamento, efectua a passagem das informações de identidade de uma forma proactiva. Exemplo deste tipo de federação seria uma aplicação Windows Forms ou uma qualquer solução rica que não corre dentro de um browser, e na qual não poderemos redireccionar para um site. Neste caso o que irá acontecer é que a validação das informações será efectuada de uma forma proactiva utilizando serviços para que seja efectuada a validação e respectiva identificação do utilizador.

Problema

Tendo todos estes conceitos em conta, o problema é que nós queremos criar uma relação de confiança para com diversos I**dPs** e por isso mesmo necessitamos de um **FG**, e uma vez que estamos num ambiente Web, dentro do browser iremos utilizar a Federação Passiva, uma vez que será possível efectuar o redireccionamento.

Até aqui tudo bem, agora só falta compreender como é que poderemos efectuar a relação de confiança entre o **RP** e o **IdP** e como poderemos utilizar um **FG** para que possamos ter mais do que um **IdP**.

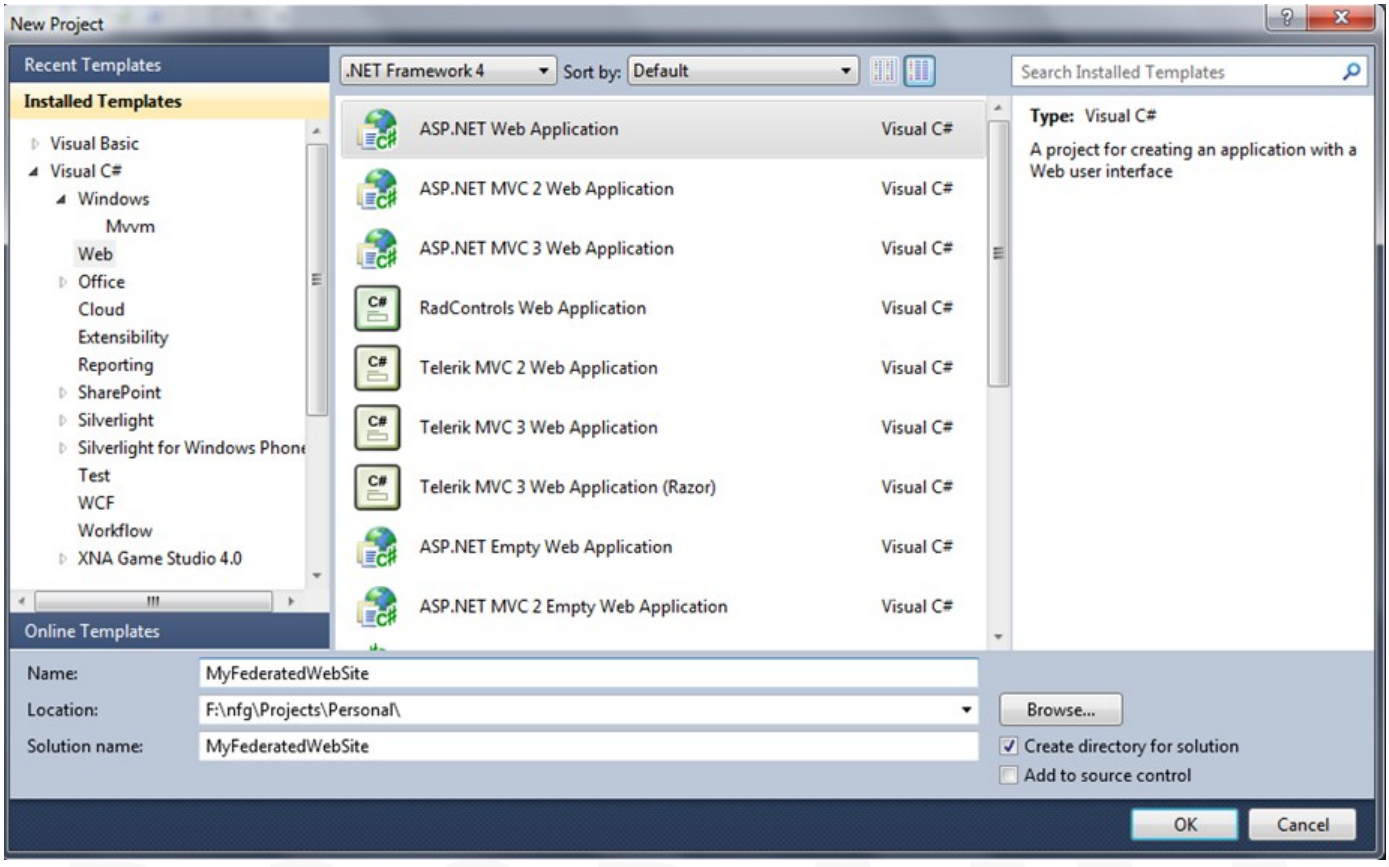
Windows Azure AppFabric Access Control Service (ACS)

Uma das formas de resolvermos este problema é utilizando o Windows Azure AppFabric Access Control Service, que é nada mais do que um serviço que nos é fornecido dentro da plataforma de Cloud Computing da Microsoft, o Windows Azure, e que desde logo funciona como **Federation Gateway**. Para além disso a Microsoft tem também um bloco denominado **WIF – Windows Identity Foundation** que é o responsável por simplificar todo o processo de trabalho com **Federação de Identidade** e **Claim-based Identity.**

Solução

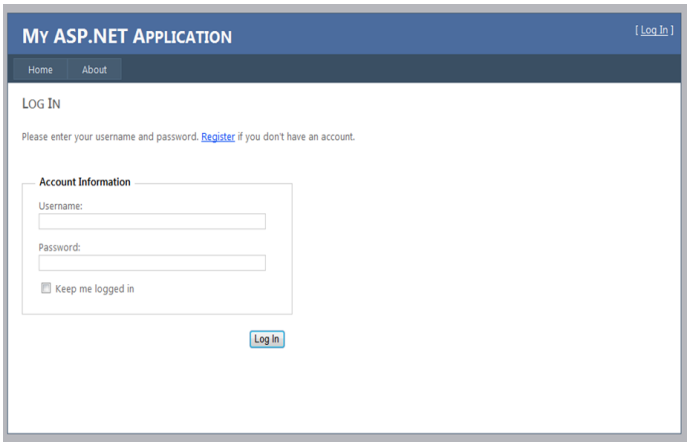
Mas como poderemos solucionar este problema com o Windows Azure AppFabric ACS? Para isso necessitamos de ter instalado o Visual Studio 2010 SP1, as Windows Azure Tools for Visual Studio 2010, preferencialmente a versão 1.6 que saiu ainda à muito pouco tempo e o WIF Runtime e SDK.

Vamos iniciar o processo com a criação de um WebSite normal em Visual Studio:

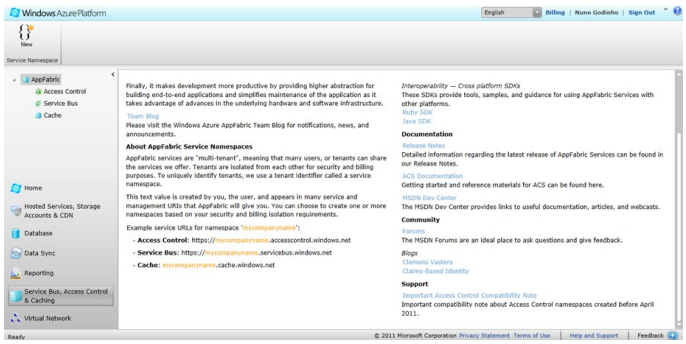


Windows Azure AppFabric ACS

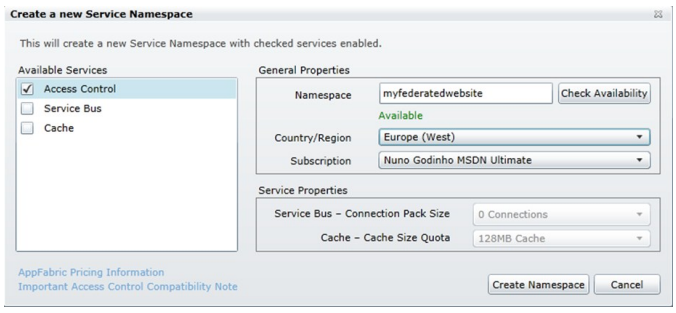
Se lançarmos agora o WebSite temos que um site normalíssimo com uma gestão de identidade tradicional.



Vamos agora preparar o Windows Azure AppFabric Access Control Service e definir quais os **Identity Providers (IdP)** que vamos querer que sejam possíveis de utilizar no nosso WebSite. Para isso vamos ao site http://windows.azure.com, ao tab de "Service Bus, Access Control & Caching".

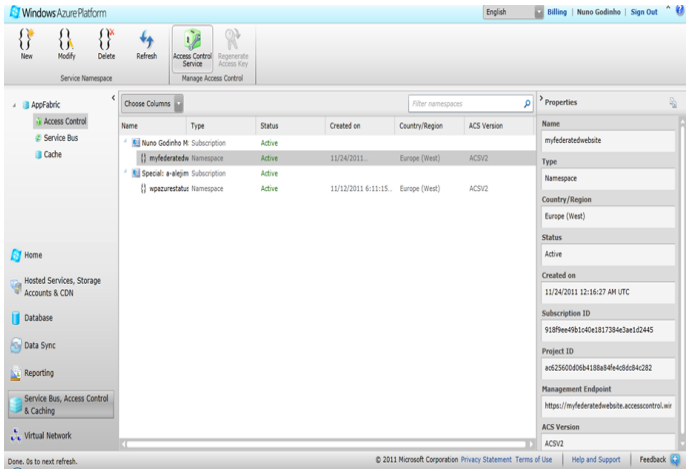


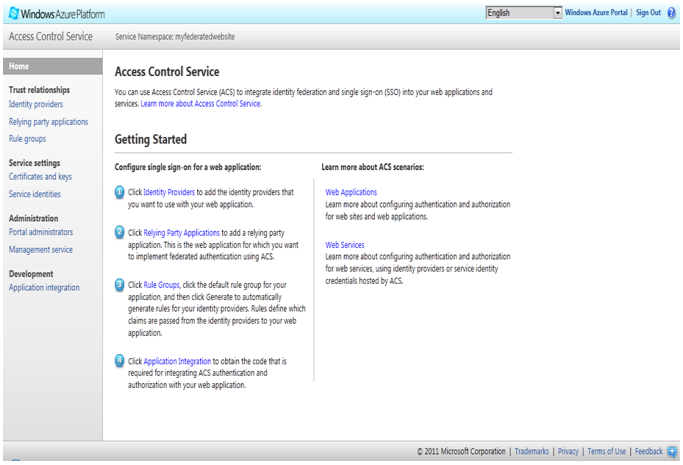
Onde escolhemos o nó Access Control. Vamos para este exemplo criar um novo namespace com o nome MyFederatedWebSite.



O que este passo irá fazer é criar uma nova entrada no DNS do Windows Azure com este nome para o tratamento de federação de identidade. O que acontece é que além da entrada no DNS é também provisionado para nós um ambiente com todos os elementos necessários para efectuar este tipo de processo já devidamente configurados, bem como a disponibilização de um backOffice de gestão do mesmo.

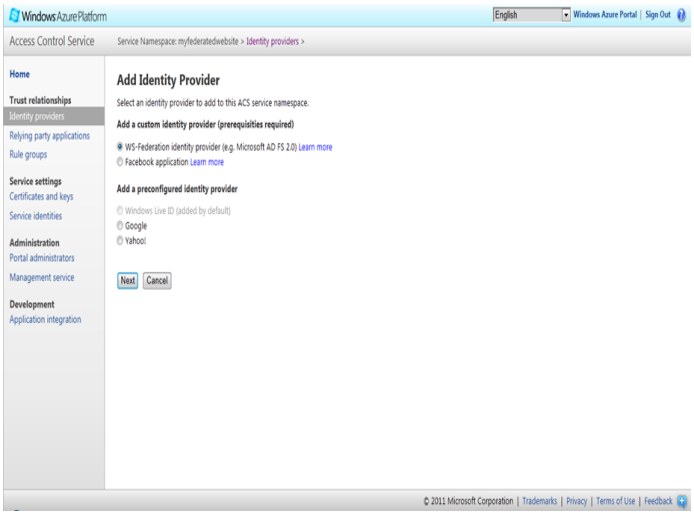
Ao escolhermos o botão Access Control Service que se encontra no topo do portal passamos imediatamente ao portal de gestão deste mesmo serviço.



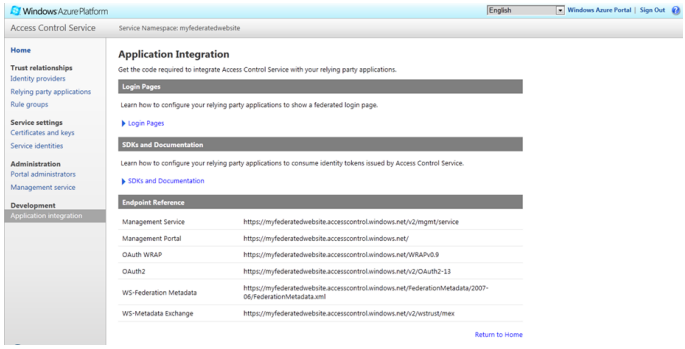


Neste portal a primeira operação que iremos efectuar é definir os Identity Providers que iremos disponibilizar, e neste caso vamos escolher Google e Yahoo.

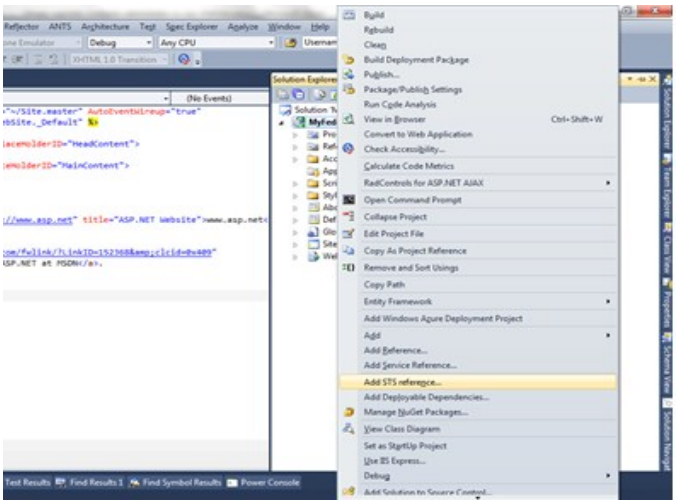
Ficando com o seguinte:



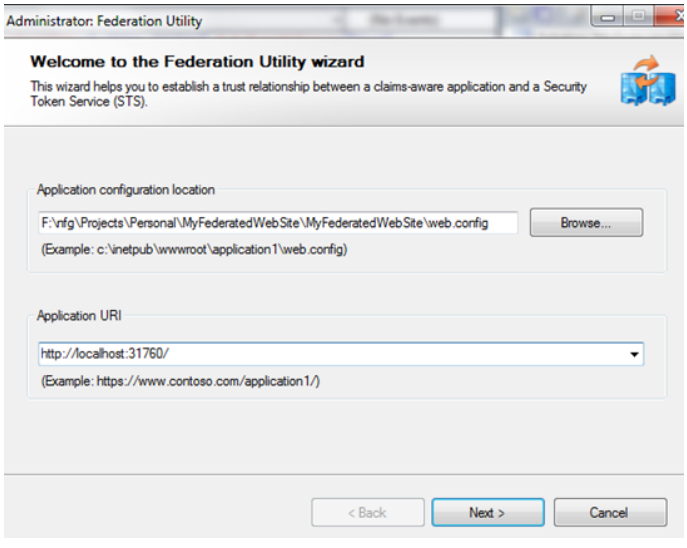
Vamos agora efectuar a relação de confiança entre o nosso WebSite e o Windows Azure AppFabric ACS. Para isso vamos seleccionar a opção Application Integration onde nos são fornecidos os dados necessários paraefectuar a dita relação.



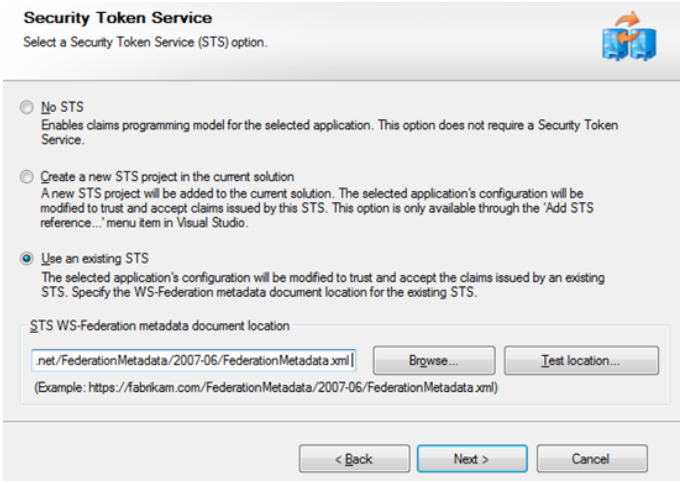
Agora voltemos ao Visual Studio 2010 e vamos associar uma referência ao ACS. Para isso vamos adiccionar uma referência ao STS (Security Token Service).



Vamos definir que o Application Uri a utilizar será o uri de acesso ao nosso WebSite

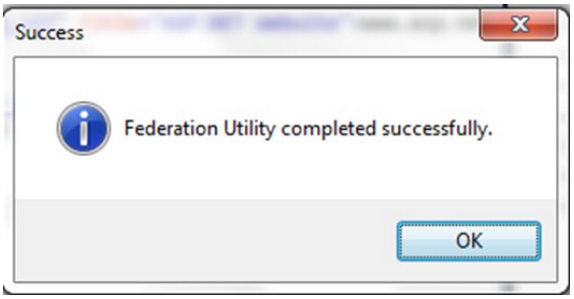


E que neste caso desejamos utilizar um STS já existente, que será o que foi criado no momento em que efectuamos o pedido de geração do namespace para Access Control Service no Windows Azure.



Vamos para já efectuar sempre Next nos restantes passos, uma vez que estamos num ambiente de testes e não queremos ter uma grande entropia no mesmo para que possamos rpidamente ver tudo em funcionamento, mas claramente em produção não seria esta a melhor opção.

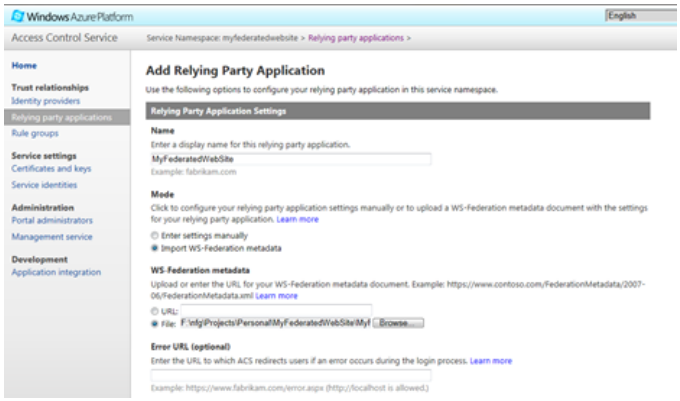
No final recebemos uma notificação muito importante.



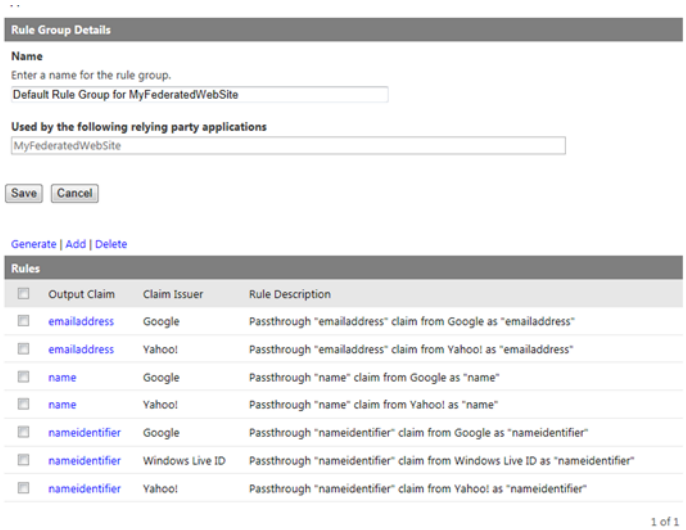
A partir deste momento já existe uma relação de confiança entre o WebSite e o ACS. Para finalizar apenas falta registarmos ao nível do ACS também este RP bem como gerar as regras de conversão para as diversas informações dos IdP para algo comum e visível para o WebSite, uma vez que cada um destes IdP funciona de forma diferente, e contm informações diferentes, teremos de transformar num formato comum para que o nosso WebSite independentemente do IdP utilizado tenha sempre a noção de quem é o utilizador, email, etc.

Para isso vamos novamente ao portal de gestão do ACS, e escolhemos a opção Relying Party Applications, onde iremos adiccionar o nosso WebSite. Para simplificar vamos importar o ficheiro FederationMetadata.xml que foi gerado no WebSite no momento da criação da referência ao STS.

Windows Azure AppFabric ACS



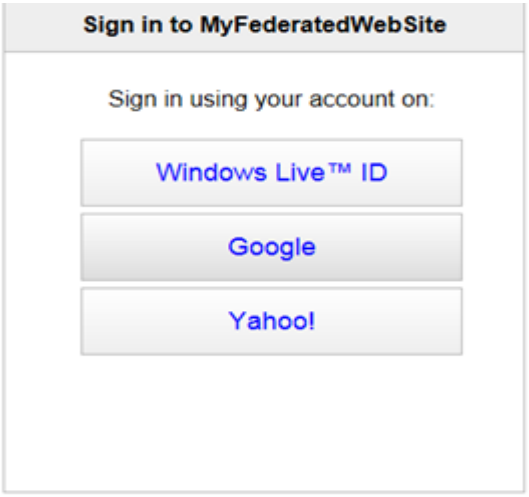
Para finalizar vamos à opção Rule Groups e seleccionamos o grupo que foi criado predefenidamente para o nosso WebSite e seleccionamos o Generate, para que todas as regras de conversão dos diversos IdPs utilizados sejam criadas.



A partir deste momento o nosso WebSite está registado no ACS e também a relação de confiança está efectuada. Além de tudo isso, as regras de conversão dos diversos IdPs utilizados foram efectuadas automticamente.

Agora vamos terminar as configurações do lado do nosso WebSite, e para isso iremos retirar do web.config todas as entradas relativas a authentication, membership, profile e roleManager, uma vez que tudo isto será da responsabilidade do ACS a partir deste momento.

Agora lançamos novamente a solução e tudo funciona, sendo que antes de chegarmos ao nosso WebSite temos o seguinte quadro que nos permite escolher o IdP que queremos utilizar.



Conclusão

Em resumo, todos concordamos que é importante que cada vez existam menos identidades digitais espalhadas pela Internet, e por isso mesmo a Federação de Identidade é uma solução muito interessante, o problema é que nem sempre é simples. Neste caso, o Windows Azure AppFabric Access Control Service vem ajudar-nos substancialmente na medida em que simplifica muito o processo tratando de toda a configuração e instalação necessária para este tipo de sistemas, e com o auxilio da WIF – Windows Identity Foundation, torna muito simples conseguir o resultado esperado.

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://bit.ly/ufTstn (Português e Inglês) e http://bit.ly/s6f5ec (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)."

Biztalk Server - Princípios Básicos dos Mapas

Os mapas, ou transformações, são um dos componentes mais comuns nos processos de integração. Funcionam como tradutores essenciais no desacoplamento entre os diferentes sistemas a interligar. Neste artigo, à medida que exploramos o editor de mapas do BizTalk Server, exploramos os seus principais conceitos enquanto abordamos temas como a arquitectura deste servidor e alguns dos padrões mais usados na tradução de mensagens.

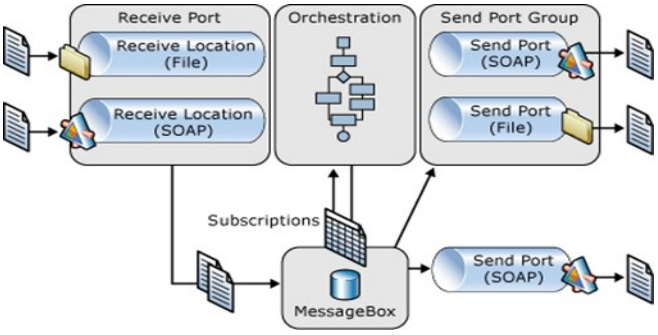
Este artigo pretende ser uma nota introdutória e destinada a quem está a dar os primeiros passos nesta tecnologia.

Podemos definir o BizTalk como um servidor de encaminhamento de mensagens, capaz de tratar, validar, transformar e controlar inúmeros processos, simplificando as necessidades de adaptação de cada sistema a interligar. Ou seja, um componente de infra-estrutura essencial nas ligações entre empresas (B2B - Business-to-Business) e cada vez mais, usado para ligar sistemas, também eles, cada vez mais complexos dentro das organizações (EAI - Enterprise Application Integration). Para além dos padrões “Fire & Forget”, o BizTalk é também usado para cenários mais complexos onde o workflow depende de várias mensagens que precisam de ser correlacionadas para orquestração processos de negócio (BPM - Business Process Management). Neste artigo vamos focar no processo de mapeamento e transformação de mensagens apenas.

De maneira simples podemos definir que o BizTalk é um servidor de integração, projectado para trabalhar com mensagens, 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).

Arquitectura

As mensagens entram no BizTalk através de uma porta lógica (portas de recepção ou Recieve Port) que são compostas por 1 ou varias portas físicas (locais de recepção ou Recieve Locations). Cada local de recepção possui uma configuração específica para um adaptador, como podemos verificar no exemplo seguinte:



Um adaptador FILE poderia ser por exemplo uma pasta de rede (\\fileshare.local\Encomendas\) e um filtro (*.edifact) e, paralelamente poderíamos também estar a receber encomendas por um Web Service (SOAP/REST/XML).

Quando o servidor recebe uma mensagem num adaptador, este executa um pipeline. O pipeline é simplesmente uma composição sequencial de componentes que tem como principal objectivo:

- **Converter as mensagens** que podem estar em diferentes formatos (arquivos de texto (Flat File), arquivos compactados - ZIP), para o formato que o BizTalk usa internamente para processar as mensagens: XML (Extensible Markup Language).

- **Validar as mensagens recebidas**. No seu normal funcionamento, o BizTalk só processa mensagens reconhecidas internamente, para isso utiliza esquemas XML (XML Schema) que permitem descrever a estrutura (record, elemento, atributo, nome, tipo de dado) e define as regras de validação (se é ou não obrigatório, numero de vezes que o elemento pode aparecer, hierarquia) das mensagens XML.

De seguida as mensagens são despejadas internamente na MessageBox (base de dados) onde são os diferentes subscritores (1 ou mais interessados nessa mensagem) a vão receber. Estes subscritores podem ser outras portas de saída (Routing) ou entram em orquestrações lançando novos processos, ou acordando os que estavam à espera (via campos correlacionáveis).

O que são os mapas de BizTalk e onde podem ser utilizados?

Os mapas de BizTalk são representações gráficas de documentos XSLT (Extensible Stylesheet Language Transformation) que permitem efectuar, de forma simples e visual, transformações às mensagens XML.

Biztalk Server - Princípios Básicos dos Mapas

Podemos enumerar os standards usandos no BizTalk Mapper:

- **XML** (Extensible Markup Language) – contêm os dados das mensagens;
- **XML Schema** (XSD - XML Schema Definition) – define o formato das mensagens;
- E **XSLT** (Extensible Stylesheet Language Transformation) – define as regras de tranformação das mensagens;

De realçar que todos eles são é uma recomendação da W3C (Worldwide Web Consortium) - consórcio internacional, que agrega empresas, órgãos governamentais e organizações independentes, e que visa desenvolver standards para a criação e a interpretação de conteúdos para a Web.

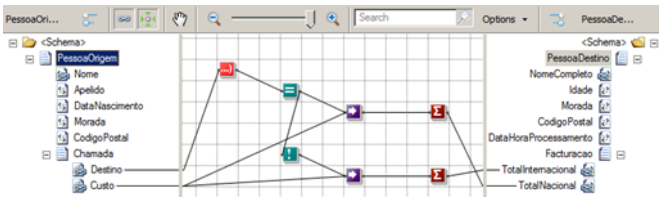
Podemos caracterizar dois tipos de transformações existentes:

- **Transformações de Sintaxe:** este tipo de transformações ocorrem nas pipelines de recepção ou envio e têm como objectivo transformar um documento noutra representação, por exemplo de CSV para XML. Aqui o documento mantem os mesmos dados (semântica), mas muda a sintaxe com que é representado. Ou seja traduzimos o documento mas, normalmente, não o modificamos em termos de estrutura. Por norma este tipo de transformação é bidireccional, uma vez que continuamos a ter o mesmo conteúdo semântico. Podemos aplicar a mesma lógica de transformação e voltar a obter um documento no seu formato original.

```
Sandro;Pereira;1978-04-04;Crestuma;4415 Crestuma
```

```
<ns0:PessoaOrigem xmlns:ns0="http://ComoFuncinam
  <Nome>Sandro</Nome>
  <Apelido>Pereira</Apelido>
  <DataNascimento>1978-04-04</DataNascimento>
  <Morada>Crestuma</Morada>
  <CodigoPostal>4415 Crestuma</CodigoPostal>
</ns0:PessoaOrigem>
```

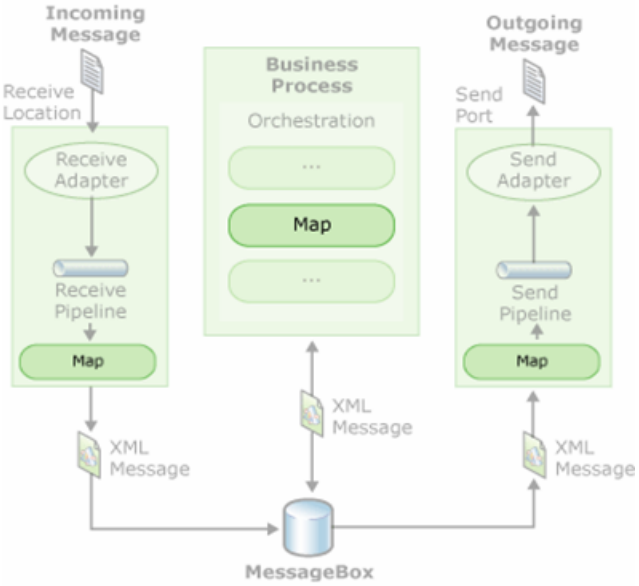
- **Transformações de Semântica:** este tipo de transformações ocorre por norma apenas nos mapas de BizTalk. Aqui o documento mantem a mesma sintaxe com que é representado (XML), mas muda a sua semântica. Tipicamente são operações One-way, uma vez que quando extraímos e agregamos partes de informação de um documento e compomos um outro documento diferente, podendo perde detalhes importantes para a sua reconstrução.



Nota: Neste artigo vamos falar apenas nas transformações de semântica, ou seja, nos mapas de BizTalk.

Onde podem ser utilizados os mapas?

Conforme a imagem a baixo demonstra, os mapas de BizTalk podem ser utilizados à entrada, nas orquestrações, ou nas portas de saída.

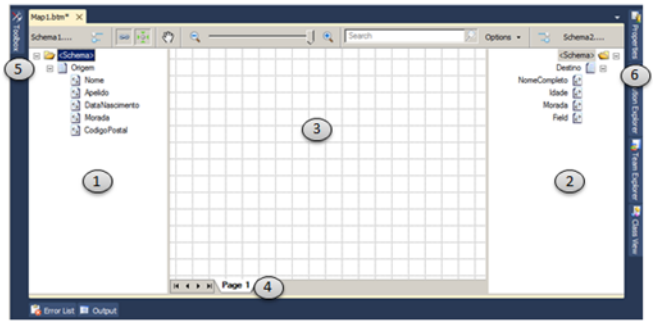


A grande diferença entre utilizar nas portas ou em orquestrações, é que a utilização na última pode ter múltiplos inputs de mensagens (transformações de vários documentos para um documento final – transformações N↔1) e nas portas apenas permite uma única mensagem de input (transformações 1↔1).

Introdução ao editor de mapas - *BizTalk Mapper Designer*

O editor de mapas, BizTalk Mapper Designer, possibilita efectuar transformações de mensagens XML complexas de forma visual e extremamente simples, expressas em associações gráficas de ligações (links) que definem as relações entre os vários elementos das mensagens.

Estas relações entre elementos são internamente implementadas como transformações XSLT (Extensible Stylesheet Language Transformation) que é o standard recomendado pela Worldwide Web Consortium (W3C) para efectuar transformações entre esquemas XML



Esta ferramenta encontra-se integrada no Visual Studio e é composta essencialmente por 3 módulos:

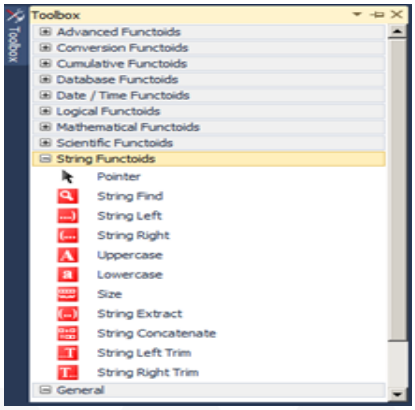
- **Esquema de Origem** (source schema): trata-se da estrutura de dados da mensagem de origem e encontra-se na parte esquerda da janela principal – **ponto 1;**

- **Esquema de Destino** (destination schema): trata-se da estrutura de dados da mensagem final após ser efectuada a transformação e encontra-se na parte direita da janela principal – **ponto 2**;

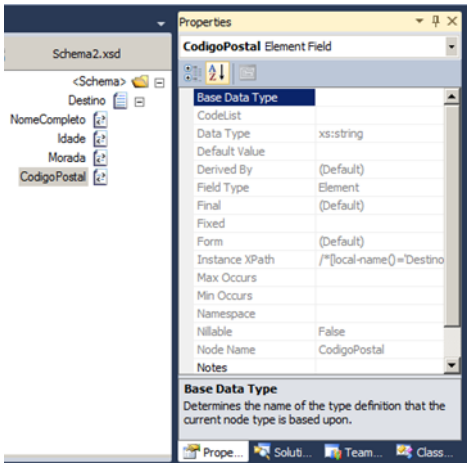
- **Grelha de mapeamento** (mapper grid): encontra-se no meio da janela principal, entre as duas estruturas de dados (origem e destino) - **ponto 3**; Esta zona desempenha um papel crítico na definição de mapas, contendo as ligações e as functoids que iram controlar a forma como os dados de origem da mensagem são transformados, de acordo com o esquema de destino, para a mensagem final. Cada mapa pode ter até 20 páginas (mapper grids), acedíveis através dos separadores (tabs) que se encontram no fundo da grelha de mapeamento – **ponto 4.**

Para além destes 3 módulos, existem 2 janelas de extrema importância para o programador:

- **Janela de Ferramentas** (toolbox window): normalmente encontra-se no lado esquerdo do esquema de origem – **ponto 5**; Providencia acesso a todas as *functoids* que podemos utilizar nos mapas.

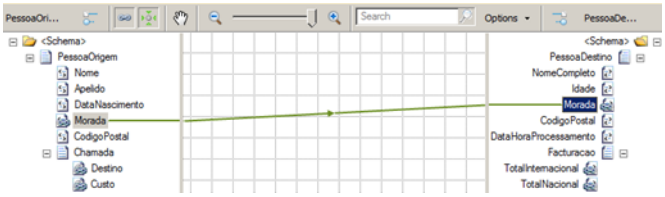


- Janela de Propriedades (properties window): nesta janela podemos ver e modificar as propriedades de um objecto seleccionado na grelha ou nos esquemas, normalmente encontra-se disponível à direita do esquema de destino – **ponto 6**.



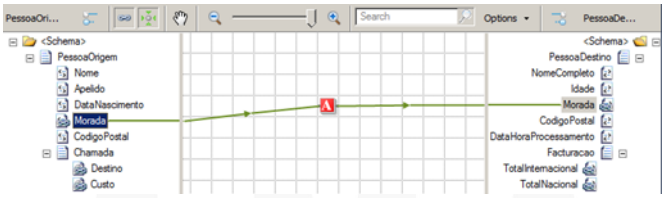
Ligações e *Functoids*

As transformações num mapa podem ser definidas na forma de relações simples, como copiar um nome ou um endereço de um documento para outro. Podemos expressar uma cópia directa dos dados usando uma ligação, que é representada no BizTalk Mapper Designer como uma linha que liga os elementos da origem para os elementos de destino.



O utilizador também pode especificar transformações mais complexas usando *functoids*. Podemos considerar uma functoid como funções pré-definidas que podemos utilizar para efectuar mapeamentos ou transformações complexas.

Tipicamente num mapa, os dados são copiados da origem para o destino, arrastando ligações entre os elementos dos dois esquemas. As functoids ficam no meio destas operações e aplicam uma operação sobre os dados de entrada, de modo a transformá-los às exigências do destino. BizTalk Mapper Designer representa uma functoid como uma caixa no meio da ligação ou ligações entre os elementos de transformação.



Biztalk Server - Princípios Básicos dos Mapas

BizTalk fornece um extenso conjunto de functoids que podem ser usadas nos mapas para executar uma variedade de operações nos dados que estão a ser transformados a partir de uma mensagem de origem para uma mensagem de destino.

Por defeito, as functoids estão organizadas em 9 categorias com base nas suas funções:

- **Advanced Functoids:** Usadas para criar vários tipos de manipulação de dados, como a implementação de script personalizado (C#, Visual Basic .NET, XSLT), mapear valores ou gerir e extrair dados de elementos recursivos.
- **Conversion Functoids:** Funções típicas de conversão de valores como: conversão de caracteres para ASCII ou de números de uma base para outra (Hexadecimal, decimal).
- **Cumulative Functoids:** Usadas para realizar vários tipos de operações de acumulação de valores que ocorrem várias vezes numa mensagem.
- **Database Functoids:** Utilizadas principalmente para pesquisar dados existentes em base de dados.
- **Date and Time Functoids:** Trata-se de um conjunto de operações sobre datas como: adicionar data, hora, data e hora, ou adicionar dias a uma data específica.
- **Logical Functoids:** Estas funções permitem controlar de forma condicional o mapeamento dos valores de origem ou o comportamento de outras functoids, determinando assim se os dados de saída são criados ou não.
- **Mathematical Functoids:** Utilize as functoids matemáticas para executar cálculos numéricos específicos, como adição, multiplicação ou divisão.
- **Scientific Functoids:** Usadas para realizar cálculos científicos específicos, como funções logarítmicas, exponenciais ou trigonométricas.
- **String Functoids:** Usadas para manipular dados alfanuméricos (texto) através de funções bem conhecidas tais como: calcular comprimento, concatenação de elementos, extrair bloco de texto, converter para maiúsculas ou minúsculas.

No entanto, a plataforma permite que sejam criadas novas functoids pelos programadores assim como organizar e criar novas categorias.

Projecto de referência para criação e instalação de novas functoids: "BizTalk Mapper Extensions UtilityPack".

Grelha de Mapeamento

A grelha de mapeamento desempenha um papel crítico na definição de mapas, ela irá conter as ligações e functoids que irão controlar a forma como os dados de origem de uma mensagem são transformados numa mensagem de destino de acordo com o seu esquema.

A grelha poderá ter múltiplas camadas, num máximo de 20, chamadas de páginas. Particionar os mapas em diferentes páginas, além de ser uma boa prática, pode-se tornar extremamente útil por forma a organizar-nos e desta forma torná-los mais legíveis. Apesar de em pequenos mapas, uma página ser suficiente, quando lidamos com esquemas complexos como EDI, "infestar" uma página com inúmeras ligações e functoids torna-os ilegíveis, chegando ao ponto de não conseguirmos distinguir um elemento do outro.

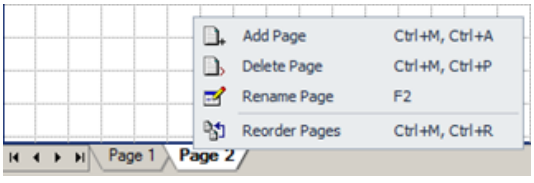
Podemos definir as páginas como contentores de ligações e functoids, que servem apenas para organizar os mapas, isto porque ao nível do compilador não têm qualquer impacto uma vez que são invisíveis.

Operações possíveis nas páginas

Por defeito os mapas são criados com apenas uma página, com o nome "Page 1", apesar das operação mais frequentes serem a de criação e renomeação, são 4 operações as operações que podemos efectuar sobre as páginas:

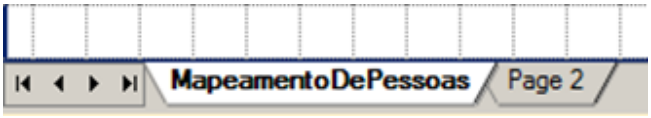
- **Adicionar nova página:** esta é a funcionalidade mais comum, adicionar novas páginas permite-nos organizar diferentes zonas do mapa em contentores.

⇒ Pressionar botão direito sofre o separador da página, e seleccionar a opção "Add Page"



- **Renomear uma página existente:** muita das vezes esquecida, esta opção permite-nos renomear os separadores de forma a tornar as diferentes páginas mais legíveis visualmente.

⇒ Pressionar botão direito sofre o separador da página, e seleccionar a opção “Rename Page”

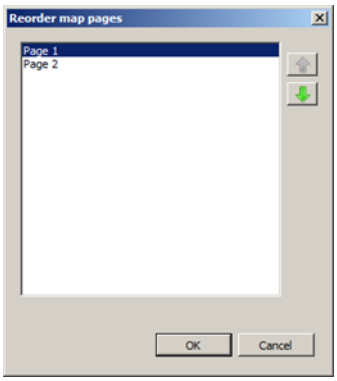


- Eliminar uma página existente: eliminação de páginas desnecessárias ou obsoletas.

⇒ Pressionar botão direito sofre o separador da página, e seleccionar a opção “Delete Page”

- Reorganizar as páginas existentes: muita das vezes temos a necessidade de organizar a disposição das páginas numa sequência diferente, para isso basta:

⇒ Pressionar botão direito sofre o separador da página, e seleccionar a opção “Reorder Pages”



Transformações - Funcionalidades básicas dos mapas

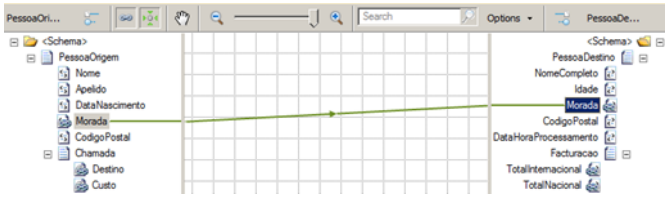
Quando estamos a efectuar uma transformação de mensagens são 5 as funcionalidades básicas que normalmente nos surgem:

- Mapeamento simples de um determinado valor (cópia directa)
- Concatenação de valores
- Selecções condicionadas
- Scripts customizados
- Adicionar novos dados

Mapeamento simples de um determinado valor (cópia directa)

Esta é a operação mais básica de todas as operações, onde pretendemos mover um valor da origem para um determinado destino, sem efectuarmos qualquer tipo de operação sobre os valores (cópia directa).

Para isso apenas de necessitamos de efectuar drag-and-drop do elemento de origem para o esquema destino. Na imagem seguinte está exemplificado o elemento Morada.



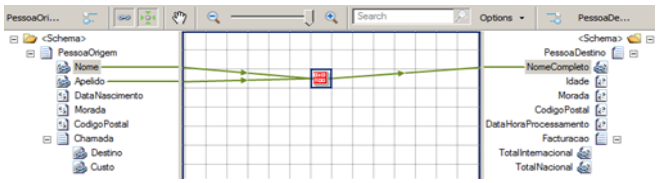
Concatenação de valores

Concatenar diferentes valores da origem para um determinado elemento no esquema de destino é outra das operações quotidianas em transformações, para isso apenas necessitamos de:

Abrir a janela de ferramentas e arrastar para a grelha a *functoid* “String Concatenate”;

Conforme foi explicado no ponto anterior, arrastar uma ligação entre os elementos pretendidos e a *functoid* “String Concatenate”, por exemplo os elementos “Nome” e “Apelido”;

Arrastar a ligação da *functoid* “String Concatenate” para o elemento no esquema de destino, neste caso o “NomeCompleto”;



Nota: a ordem de entrada dos elementos na functoid é importante, uma vez que a concatenação é efectuada pela ordem de entrada dos elementos (iremos aprofundar este tema mais à frente).

Selecções condicionadas

Muitas das vezes não queremos simplesmente mover valores da origem para o destino, às vezes necessitamos de gerar a saída dos dados de acordo com certas condições. Podemos efectuar estas condições de muitas formas distintas através de functoids ou scripts customizados, aqui fica um exemplo: Testar se o valor na origem é uma string válida, se sim mapeá-la para o esquema de origem, para isso necessitamos:

- Arrastar a functoid “Logical String” para a grelha, esta functoid permite validar se um determinado valor é uma string válida, semelhante a função C# String.IsNullOrEmpty

⇒ Retorna “False” se a string for vazia ou nula;

⇒ Retorna “True” caso seja uma string válida;

- Arrastar uma ligação do elemento pretendido da origem para a functoid “Logical String”, neste caso o elemento “CodigoPostal”;

- Arrastar a functoid “Value Mapping” para a grelha, esta functoid providencia, através de um valor booleano, controlar a saída dos valores da origem para o destino, ou seja, recebe dois valores:

  ⇒ O primeiro terá de ser um booleano (True/ False);

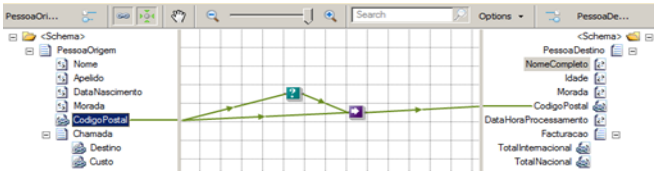
  ⇒ O segundo será o valor a ser mapeado;

  ⇒ Caso o primeiro seja verdadeiro, o valor é mapeado para o esquema de destino, caso seja falso, o mesmo não é mapeado.

- Arrastar uma ligação da functoid “Logical String” para a functoid “Value Mapping”;

- Arrastar uma ligação do elemento “CodigoPostal” do esquema de origem para a functoid “Value Mapping”;

- Arrastar uma ligação da functoid “Value Mapping” para o elemento “CodigoPostal” do esquema de destino;



Scripts customizados

Scripts customizados são utilizados normalmente em transformações mais complexas ou por forma a facilitar algumas condições. Basicamente existem dois cenários para utilizarmos esta functoid:

- Quando nenhuma das functoids existentes permite efectuar o que pretendemos, o exemplo que vamos ver é converter uma data de nascimento em idade.

- Ou quando a utilização das functoids existentes tornam-se extremamente complexas para resolver um problema do mapeamento.

- Existe uma “regra” que normalmente usa-mos para definir se devemos utilizar functoids ou utilizar scripts customizados que nos diz: se para um determinado mapeamento utilizarmos mais de 6 functoids, então deveremos utilizar scripts customizados.

- Eu gosto de utilizar esta regra de forma ponderada e não à letra, ou seja, se as functoids existentes me ajudarem de forma fácil a resolver o problema, eu utilizo as functoids existentes, se a utilização das mesmas, se torna extremamente complexa, então opto por utilizar scripts customizados.

- Estão ao dispor do programador 6 tipos de scripts:

- External Assembly: Permite-nos associar e invocar esta functoid com uma função existente numa assembly existente na Global Assembly Cache (GAC).

- Inline C#: Esta opção permite-nos associar e invocar código C# directamente na functoid (Inline Script Buffer property).

- Inline JScript .NET: Igual à anterior, mas com uso de código JScript .NET

- Inline Visual Basic .NET: Igual às anteriores, mas com uso de código Visual Basic .NET

- Inline XSLT: Esta opção permite-nos associar e invocar scripts XSLT directamente na functoid (Inline Script Buffer property).

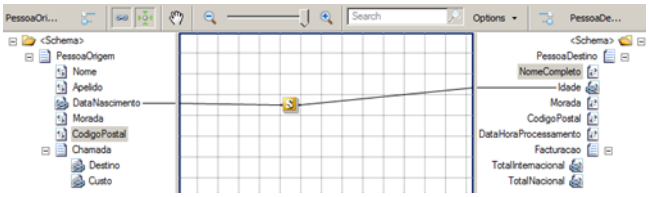
- Inline XSLT Call Template: idêntico à anterior, no entanto esta permite-nos invocar templates XSLT directamente da functoid.

Neste exemplo, queremos transformar a data de nascimento na idade da pessoa, neste cenário devemos:

- Arrastar a functoid “Scripting” para a grelha;

- Arrastar uma ligação do elemento “DataNascimento” do esquema de origem para a functoid “Scripting”;

- Arrastar uma ligação da functoid “Scripting” para o elemento “Idade” do esquema de destino;

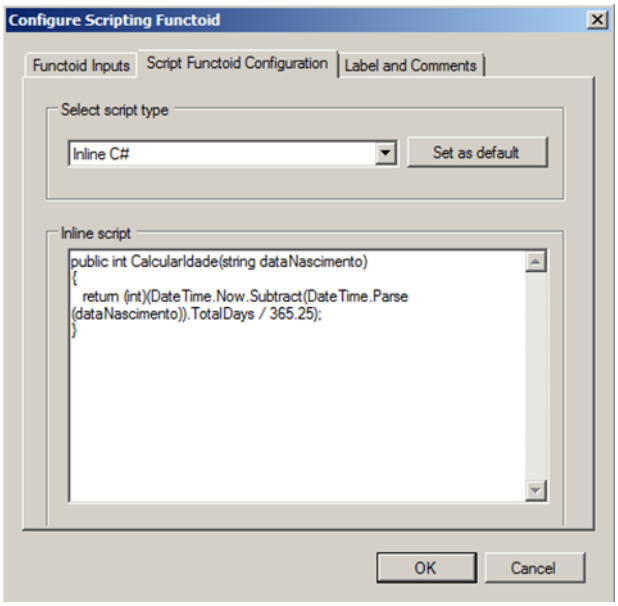


O mapeamento encontra-se efectuado, faltando-nos apenas configurar a script customizado. Para este cenário vamos utilizar um script do tipo “Inline C#”, e para isso devemos:

- Efectuar duplo click na functoid “Scripting” e seleccionar o separador “Script Functoid Configuration”;

- Na opção “Select script type”, seleccionar da lista a opção “Inline C#”;

- Na opção “Inline script” colocar o seguinte script:

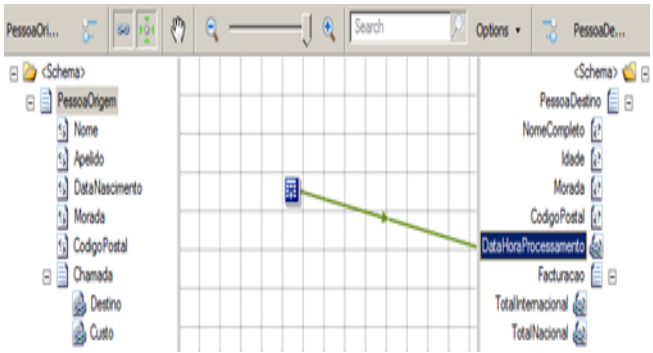


Adicionar novos dados

Em muitos cenários temos a necessidade de adicionar novos dados à mensagem final que não existem na mensagem de origem. É normal encontrarem situações em que, com base em alguns dados existentes na mensagem de origem, necessitaremos de consultar um sistema externo, por exemplo base de dados, por forma a adquirimos mais informações para preencher os dados necessários na mensagem final.

Um exemplo mais básico e simples é adicionar à mensagem final um carimbo de data/hora actual em que a mensagem é processada, para isso devemos:

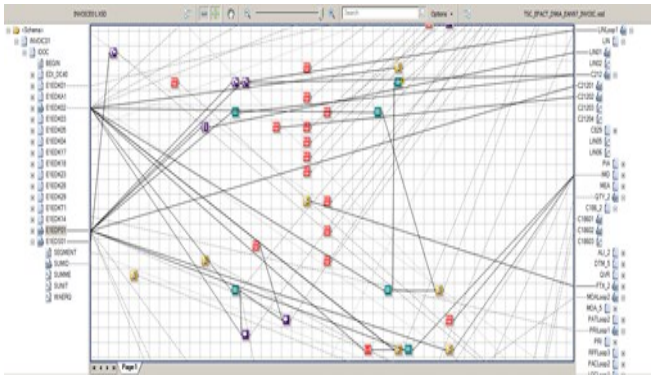
- Arrastar a functoid “Date and Time” para a grelha;
- Arrastar uma ligação da functoid “Date and Time” para o elemento no esquema de destino “DataHoraProcessamento”;



Nota: Como podem verificar, esta functoid não requer nenhum dado de entrada, retornando a data e hora actual do sistema.

Como organizar os mapas BizTalk

Os mapas podem tornar-se extremamente complexos, dificultando assim a sua leitura e manutenção.

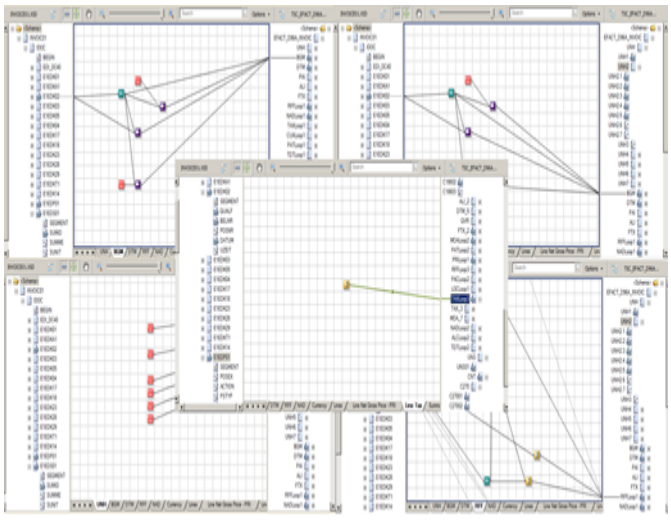


O editor de mapas do BizTalk providencia duas importantes funcionalidades que nos permitem minimizar este problema:

- Páginas
- Etiquetas para as ligações

Páginas

As páginas permitindo organizar os mapas, especialmente os mais complexos, em subdivisões lógicas de mapeamentos. Esta funcionalidade já foi descrita anteriormente.



Etiquetas para as ligações

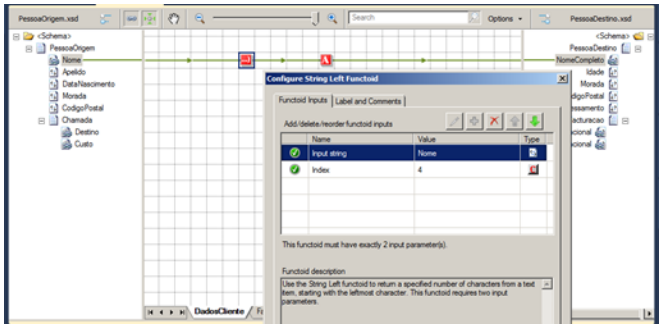
Nas versões anteriores do produto, por defeito, era apresentado a query XPATH se uma ligação fosse estabelecida a uma functoid:

- /*[local-name()='PessoaOrigem' and namespace-uri()='http://ComoFuncinamOsMapas.PessoaOrigem']/*[local-name()='Nome' and namespace-uri()='']

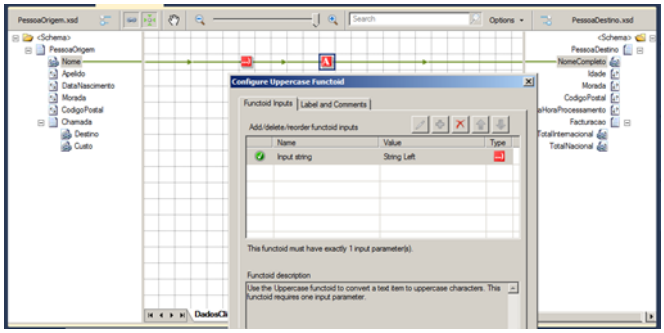
Ou o nome da functoid caso a ligação proviesse de outra functoid, o que poderá dificultar a leitura dos mapas.

Nesta versão do produto este comportamento foi ligeiramente melhorado, sendo que actualmente o valor defeito é o

nome do elemento do esquema de origem de onde provem a ligação:

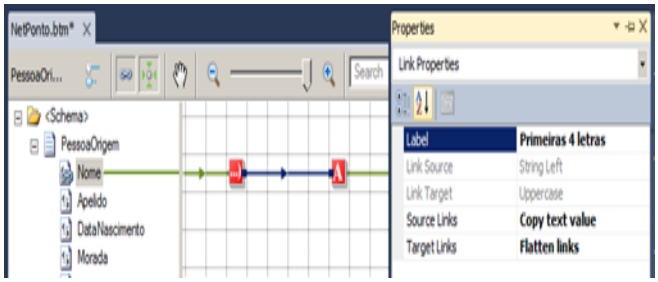


Ou o **nome da functoid** anterior:



No entanto, o editor de mapas permite-nos colocar etiquetas nas ligações, originando a que a mesma substitua a query XPATH (nas versões antigas) ou o nome do elemento do esquema de origem, para isso devemos:

- Seleccionar a ligação e com o botão direito do rato seleccionar a opção “Properties”;
- Preencher a propriedade “Label” com a etiqueta pretendida.



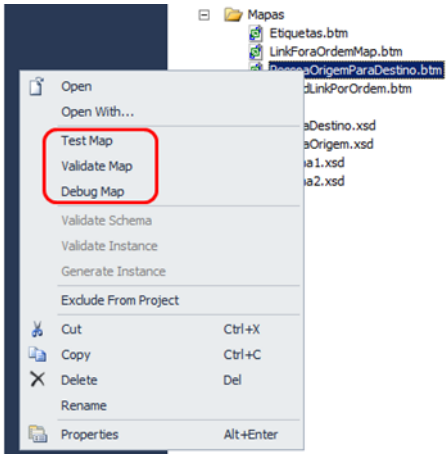
Normalmente esta funcionalidade é esquecida pelos programadores.

Apesar de parecer uma funcionalidade banal e sem impacto significativo, a meu ver, trata-se de uma funcionalidade importante a longo prazo. Enquanto as ideias estão frescas sabemos o que estamos a fazer, mas se for necessário intervir passado algum tempo e rever os mapeamentos, esta funcionalidade irá tornar a nossa tarefa mais facilitada.

Operações possíveis de efectuar sobre os mapas

Na fase de desenvolvimento, temos ao dispor 3 funcionalidades, incluídas no Visual Studio, que nos permite testar e validar os mapas:

- Testar mapas
- Validar mapas
- Depurar mapas



Estas funcionalidades estão ao dispor dos programadores de uma forma fácil e directamente da ferramenta de desenvolvimento, Visual Studio, não necessitando assim de executar (build) e publicar (deploy) os mapas ou até mesmo criar e configurar portas.

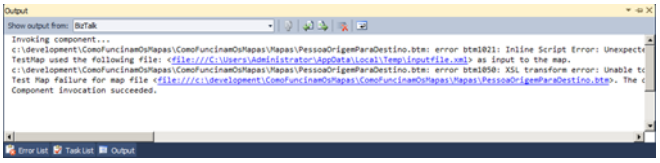
Testar mapas (Test Map)

Deveremos testar o nosso mapa de uma forma continua, não apenas no final do desenvolvimento, mas sempre que necessário ou quando um bloco de transformação importante seja concluído.

Para isso apenas necessitamos de:

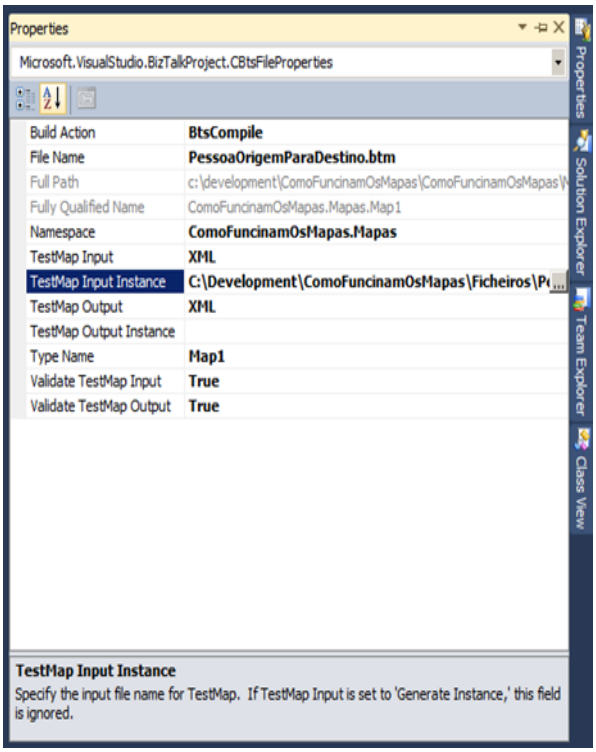
- Abrir a janela do “Solution Explorer”
- Seleccionar o mapa em causa
- Premir o botão direito do rato sobre o mapa e seleccionar a opção “Test Map”

Por defeito uma instância do esquema de entrada é gerada automaticamente com valores, também eles por omissão, de acordo com o seu tipo e testada no mapa seleccionado. No final é apresentado o resultado gerado ou os erros ocorridos na execução do mesmo na janela de saída (Output).



No entanto, muita das vezes, este cenário não é o ideal, e o que pretendemos é testar um documento existente com um mapa. Para isso apenas necessitamos de configurar as propriedades do mapa antes de corremos o teste:

- Efectuar o procedimento anterior, mas em vez de seleccionar a opção “Test Map”, iremos seleccionar a opção “Properties”;
- Na janela de propriedades configurar a propriedade “TestMap Input Instance” com o caminho para o ficheiro de entrada.



Nesta janela também podemos configurar diversas propriedades como: o formato de entrada e saída dos documentos, assim como o caminho final do documento de saída, mas mais importante podemos configurar se pretendemos validar os formatos de entrada e saída com os respectivos esquemas.

Esta segunda opção: **validar formato do ficheiro de saída** (Validate Test Map Output), é extremamente importante para os testes intermédios. Ao configurar esta opção como “False” permite-nos testar um mapa incompleto sem ser apresentados erros de testes ou falta de dados obrigatórios, muito deles associados a zonas por mapear, permitindo-nos assim apenas validar o trabalho efectuado até à data.

Nota: Esta opção deverá ser configurada como “True” para os testes finais.

Validar mapas (Validate Map)

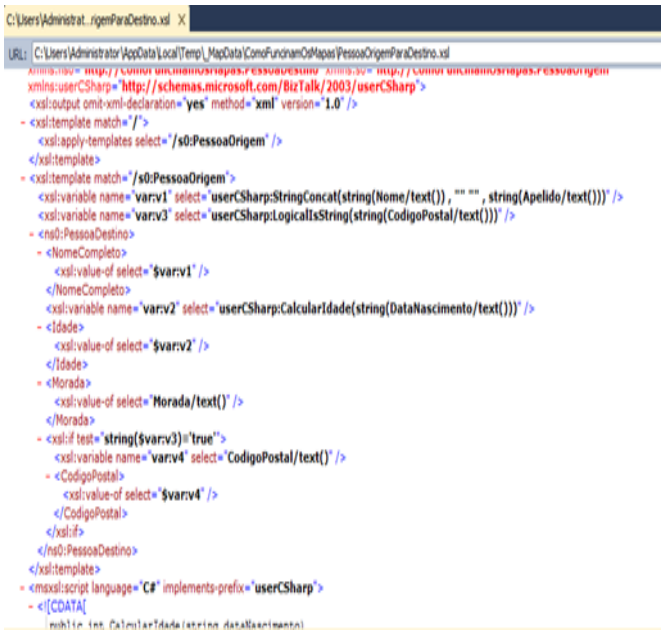
Esta opção permite-nos validar a estrutura do mapa. Podemos desta forma inspeccionar e analisar o código XSLT gerado pelo compilador, fornecendo-nos informações

adicionais de como os mapas funcionam assim como mais uma outra opção de depuração (debug).

Para executar esta opção basta:

- Aceder ao “Solution Explorer”;
- Botão direito do rato sobre o mapa (arquivo *. btm)
- E seleccione a opção "Validar Map".

Uma ligação para o XSLT gerado será apresentada na janela de saída (Output).



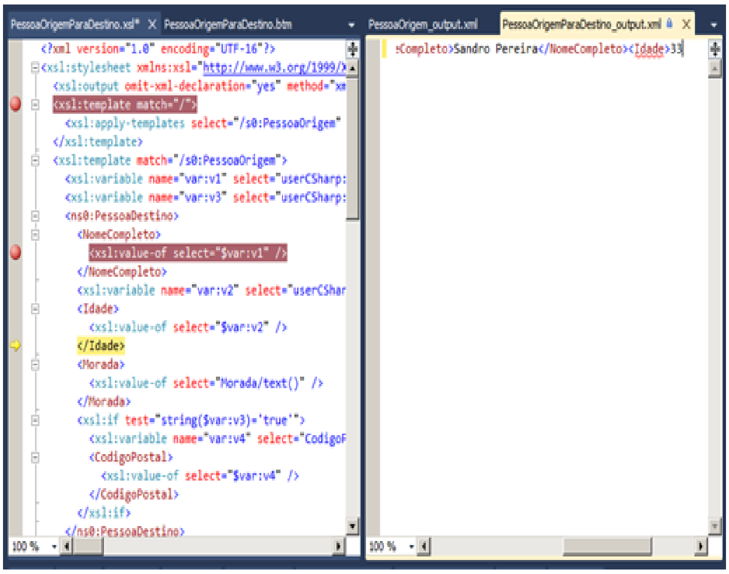
Depurar mapas (Debug Map)

Esta opção permite-nos efectuar a depuração (debug) de mapas e assim identificar e corrigir problemas complexos de mapeamento na fase de desenvolvimento.

Esta funcionalidade faz uso de algumas propriedades dos mapas, tais como “TestMap Input Instance” e “TestMap Output Instance”. Portanto, antes de depurar o mapa, será necessário configurar as suas propriedades dos mapas, com especial foco nas duas anteriores.

Para executar esta opção, teremos de:

- Aceder ao “Solution Explorer”
- Botão direito do rato sobre o mapa (arquivo *. btm)
- E seleccione a opção "Debug Map".
- Pressione a tecla F10 ou F11 para depurar o código XSL.
    - Quando você pressiona F11 numa chamada a uma functoid, o Visual Studio irá entrar para o código C# associado à functoid.



> Podemos definir o BizTalk como um servidor de encaminhamento de mensagens, capaz de tratar, validar, transformar e controlar inúmeros processos

Conclusão

Devido ao inúmero número de sistemas e aplicação distintas existentes nas organizações, é imperativo usar boas ferramentas e técnicas para produzir soluções que funcionem durante muitos anos de uma forma controlada e fácil de manter. Ao mesmo tempo novos processos são adicionados e os existentes vão sofrendo pequenas melhorias, tudo sem perder o rasto ao que está a acontecer em produtivo.

O BizTalk ajuda-nos a resolver muitos destes problemas, dispondo de inúmeras funcionalidades “out of the box” com o produto. Neste artigo acho que consegui explicar de uma forma intuitiva os principais conceitos básicos dos mapas de BizTalk, à medida que exploramos o seu editor.

AUTOR

Escrito por Sandro Pereira

Actualmente Senior Software Developer na empresa DevScope. É Microsoft Most Valuable Professional (MVP) em Microsoft BizTalk. O seu principal foco de interesse são as tecnologias e plataformas de Integração (EAI): BizTalk e SOAP / XML / XSLT e Net, que utiliza desde 2002. É um participante bastante activo nos fóruns da Microsoft (MSDN BizTalk Server Forums), contribuidor no MSDN Code Gallery e autor do Blog: http://bit.ly/oFLwB4- Twitter: @sandro_asp – Membro da comunidade BizTalk Brasil: http://bit.ly/9NI7ie

Veja também as edições anteriores da Revista PROGRAMAR

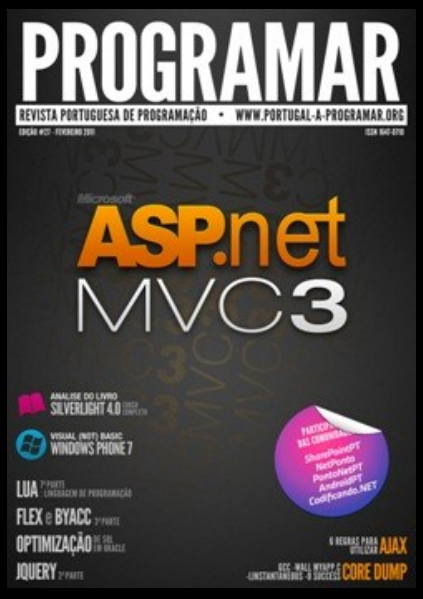
31ª Edição - Outubro 2011

30ª Edição - Agosto 2011

29ª Edição - Junho 2011

28ª Edição - Abril 2011

27ª Edição - Fevereiro 2011



26ª Edição - Dezembro 2010

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar.org