# PROGRAMAR

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

EDIÇÃO #28- ABRIL 2011

ISSN 1647-0710


# BUSINESS CONNECTIVITY SERVICES

COLUNAS

**ENTITY FRAMEWORK** MODEL-FIRST CODE-FIRST

**CORE DUMP** O "X" NO QUADRADO CERTO

A PROGRAMAR

**LUA** 7ª PARTE : LINGUAGEM DE PROGRAMAÇÃO

**ORACLE** PLANOS DE EXECUÇÃO

**WPF** DATAGRIDS NA PRATICA

**NOVIDADES** JQUERY 1.5 E AJAX

**SMARTY PHP** TEMPLATE ENGINE

COMUNIDADES

NETPONTO

AUTOMATIZAÇÃO DE DEPLOYMENTS **WINDOWS AZURE**

ANDROIDPT

INTER-PROCESS COMMUNICATION **ANDROID**

SHAREPOINTPT

BUSINESS CONNECTIVITY SERVICES **SHAREPOINT**

# EDITORIAL

## EQUIPA PROGRAMAR



Contacto
revistaprogramar@portugal-a-programar.info

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

ISSN
1647-0710

## FMI - Fim da Maravilhosa Internet?

Aconteceu. O ICANN (Internet Corporation for Assigned Names and Numbers) distribuiu o último bloco de endereços IPV4 pelos 5 RIRs (Regional Internet Registries). Ou seja após esses endereços serem atribuídos aos ISP, poderá estar em causa a aceitação de novos clientes por parte dos ISP e não haverá no IPV4 mais lugar a expansão. Ou haverá? O facto é que a norma IPV6 já está preparada (foi descrito como um standard da Internet num documento publicado em Dezembro de 2008), e já em Agosto de 2008, na edição #15 da Revista PROGRAMAR, foi publicado um artigo que falava um pouco sobre esta nova norma. Contudo a verdade é que parece não existir uma vontade comercial muito grande para adoptar esta nova norma, porque é financeiramente desagradável, uma vez que é necessário aos ISP fazerem a actualização do seu hardware de rede, aos programadores actualizarem as suas aplicações…

No dia 8 de Junho, algumas empresas do sector da Internet e não só disponibilizarão durante 24 horas os seus serviços em IPV6, para o utilizador testar os seus serviços com esta nova norma e para pressionar um pouco os ISP a fazerem a migração. Pode testar a sua ligação neste site: http://test-ipv6.com/

Nos EUA existem já soluções que tentam contornar este problema, sem no entanto ser necessário passar a usar o IPV6. Tecnicamente dois ou mais clientes de um ISP estarão numa rede mais pequena (possivelmente ligados pela central mais próxima), e existirá um router, e esse sim terá o verdadeiro endereço IP da internet, enquanto os PCs dos clientes têm apenas endereços de redes internas. Na prática dois ou mais clientes podem ter o mesmo IP ao mesmo tempo (algo idêntico ao que acontece nos hotspots). Mas não será isto simplesmente um remedeio? O ser humano tem tendências para rejeitar mudanças, prolongando até ao limite em que já não pode adiar mais essa mudança.

Um estudo mundial recente mostra que uma grande parte dos jovens dos denominados países desenvolvidos está dependente de dispositivos com acesso à Internet, sofrendo sintomas de abstinência comuns quando estão afastados muito tempo desses dispositivos. Traduzindo não podem viver sem a Internet. Mas e se os ISP não derem “o salto” para que a Internet continue a existir como a conhecemos? E se os programadores não adaptarem as suas aplicações a este salto? Poderá isto matar a Internet, pelo facto de o utilizador deixar de confiar nela?

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

# INDICE

## TEMA DE CAPA


6 Business Connectivity Services


## A PROGRAMAR


17 Lua – Linguagem de Programação (Parte 8)
21 jQuery 1.5 e AJAX
27 Datagrid em Windows Presentation Foundation
37 Planos de Execução em ORACLE
47 Smarty PHP Template Engine


## COLUNAS


48 CORE DUMP - O X No Quadrado Certo
50 VISUAL (NOT) BASIC - Entity Framework 4.0: Model-First e Code-First


## COMUNIDADES


58 AndroidIPC - Inter Process-Communication
62 Automatização de deployments em Windows Azure


## EVENTOS

**15 Abr.** **SQL Saturday - Portugal 2011**
**15 Abr.** **6º Encontro Nacional de Estudantes de Informática ENEI'2011**
**16 Abr.** **SharePointPT - 10ª Reunião da SPUGPT**
**16 Abr.** **19ª Reunião Presencial da Comunidade NetPonto - Lisboa**
**03 Mai.** **Fim das Inscrições ONI'2011**
**06 Mai.** **Microsoft WebCamp Portugal - Lisboa**
**06 Mai.** **Prova de Qualificação na Internet ONI'2011**
**09 Mai.** **CLOUD para DEVELOPERS**
**13 Mai.** **StopNplay Lan Party 2011**
**27 Mai.** **Final Nacional ONI'2011**

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

# NOTICIAS

## Resultados sobre o Panorama Nacional Tecnológico

O Sapo Developers Blog fez recentemente um inquérito sobre um conjunto de questões sobre o Panorama Nacional Tecnológico. Responderam 421 pessoas sendo a maioria dos inquiridos jovens com idades entre os 21 e os 35 anos.
Os resultado apresentados, revelam que os recursos online mais influentes são:

Portugal a Programar
Aberto até de Madrugada
SAPO Tek
Zwame
Planet Geek
Pplware
Exame Informática
SAPO Developers Blog
Globais: Engadget, Gizmodo, Mashable, ReadWriteWeb, outras.

Mais info

## Lançamento do GNOME 3.0

O GNOME 3.0 é o principal marco na história do projeto GNOME. O lançamento introduz um excitante novo ambiente de trabalho que foi desenhado para utilizadores comuns e adequado para uma grande quantidade de modernos dispositivos computacionais. As tecnologias de desenvolvimento do GNOME foram substancialmente melhoradas para o GNOME 3.0. Modernizadas e padronizadas, elas irão permitir aos desenvolvedores promover melhorias na experiência do utilizador com menos tempo e esforço. E o GNOME 3.0 vem com as mesmas aplicações GNOME que os utilizadores conhecem e confiam, muitos dos quais receberam melhorias significantes.

Mais info

## Lei das Normas abertas aprovada na AR

Foi votada no Plenário da Assembleia da República a Lei das Normas Abertas, uma proposta que já tinha passado na especialidade e prevê a garantia de interoperabilidade e adopção de normas abertas nos sistemas informáticos do Estado.

Depois de já ter conseguido gerar consenso, e muitas vezes unanimidade, na discussão na especialidade, a Lei foi aprovada com votos a favor do PEV, PCP, BE, CDS e PS, abstendo-se o PSD.

Bruno Dias, deputado do PCP, já tinha adiantado que "este foi um processo legislativo de enorme abertura e espírito construtivo de todas as partes. Não houve "ideias fixas" porque as opiniões que foram surgindo contribuíam para aperfeiçoar o texto, e dessa maneira foram tidas em conta".

## Windows 8 poderá ter App Store

om a disponibilização da primeira versão do Windows 8 para os RTM's, começaram a surgir as primeiras imagens do que será o novo sistema operativo da Microsoft. Estas imagens, não oficiais mostram um sistema operativo que seguirá a continuidade do que o Windows 7 nos trouxe, mas também novidades que farão as delicias de todos os que o forem usar.
As últimas imagens que surgiram mostram que o Windows 8 poderá ter uma store incorporada e onde os utilizadores poderão aceder às aplicações que pretenderem. Ainda não existem muitas informações sobre que software esta store conterá e se apenas existirá software gratuito ou se existirá a possibilidade de os utilizadores adquirem aplicações.
O conceito de store dentro do sistema operativo não é novo, mas nos últimos tempos tem ganho uma importancia grande, com a disponibilização da Apple Store dentro do Mac OS e até com o Ubuntu Software Center.

# TEMA DE CAPA

## Business Connectivity Services

# Business Connectivity Services

O SharePoint 2010 é uma plataforma complexa e com um impressionante conjunto de funcionalidades nativas que lhe permitem adaptar-se a uma enorme variedade de situações. Uma das novas funcionalidades com mais potencial designa-se Business Connectivity Services e este artigo é uma introdução a esta tecnologia e às suas potencialidades.

## O que são os Business Connectivity Services e para que servem?

Business Connectivity Services (BCS) é o nome da tecnologia integrada no SharePoint 2010 que permite ler e escrever informação em sistemas externos a partir do SharePoint 2010 e do Office 2010. Trata-se de uma evolução da tecnologia Business Data Catalog (BDC) introduzida no SharePoint 2007, e sobre a qual foram feitas várias melhorias, nomeadamente:

- Possibilidade de leitura e escrita sobre a fonte de dados externa;
- Suporte para cenários de autenticação mais complexos;
- Suporte para múltiplas fontes de dados;
- Integração com aplicações Office;
- Novas e melhores formas de apresentar a informação;
- Ferramentas destinadas à criação e manipulação dos modelos;
- Extensibilidade através de assemblies .Net.

O objectivo desta tecnologia é permitir a integração de informação proveniente de sistemas externos e apresentá-la em SharePoint e aplicações Office com o mínimo de esforço possível e, idealmente, sem ser necessário escrever qualquer linha de código. Há, de facto, um conjunto de cenários em que é possível a utilização da tecnologia BCS apenas por configuração mas é a sua extensibilidade que lhe permite adequar-se a praticamente qualquer necessidade de integração.

## Arquitectura

A tecnologia BCS não se limita a apenas um serviço ou API dentro do SharePoint 2010. É, na realidade, um conjunto de componentes, serviços e ferramentas tal como apresentado no esquema abaixo.

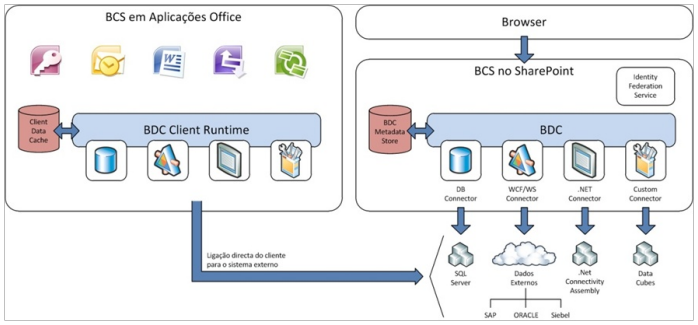



Figura 1 - Arquitectura dos Business Connectivity Services

## Business Data Connectivity (BDC) Service

O Business Data Connectivity Service é uma das peças mais importantes dos BCS. Trata-se do componente que permite, através do seu repositório central de metainformação, guardar as descrições da informação à qual se pretende aceder bem como do próprio sistema externo onde esta está armazenada.

## Metadata Store

O repositório de metainformação é a base de dados utilizada pelo Business Data Connectivity Service para armazenar as descrições da informação e dos sistemas externos onde esta está armazenada. Este repositório não contém qualquer informação proveniente dos sistemas externos, apenas a metainformação necessária para a obter.

## Connectors

Os connectors são as peças que permitem ao Business Data Connectivity Service ligar-se às fontes de dados externas descritas nos modelos armazenados no seu Metadata Store. São fornecidos três conectores com o produto:

- Database Connector – permite a ligação a bases de dados SQL Server.
- WCF/Web Services Connector – permite a ligação a serviços WCF ou web services.
- .Net Assembly Connector – permite a ligação utilizando um assembly .Net desenvolvido à medida. Uma vez que se trata de um assembly desenvolvido à medida, este conector permite a ligação a virtualmente qualquer fonte de dados externa, incluindo até a ligação a múltiplas fontes em simultâneo.

Este mecanismo de connectors é extensível, sendo ainda possível desenvolver conectores à medida, para casos em que os conectores existentes não são suficientes.

## BDC Client Runtime

As aplicações cliente que fazem parte do Office 2010 conseguem também expor informação proveniente de sistemas externos através dos BCS. Isso é possível porque o Office 2010 inclui o BDC Client Runtime, um componente que faz no contexto da aplicação cliente o que o BDC Service faz no contexto do
servidor SharePoint, ou seja, acede ao repositório de metainformação e, através das definições que este contém, acede à informação propriamente dita.

## Client Data Cache

No sentido de acelerar o acesso à informação, bem como para suportar cenários de acesso offline à informação, os BCS utilizam uma cache para guardar a informação externa obtida através dos mesmos. Esta cache é baseada numa base de dados SQL Server 2005 Compact Edition e possui um mecanismo de sincronização automático que permite que todas as alterações efectuadas sobre a informação em modo offline sejam replicadas assim que o sistema externo fica disponível.

## Principais Conceitos

Uma vez conhecida a arquitectura dos Business Connectivity Services, é importante que se perceba em que consiste a metainformação que é armazenada na Metadata Store pelo Business Data Connectivity Service.

## Modelo

A metainformação utilizada pelo BDC Service e armazenada na Metadata Store materializa-se em ficheiros XML que descrevem Modelos, normalmente designados por BDC Metadata Models. No SharePoint 2007, estes ficheiros de metainformação eram designados por application definition files.

Um modelo contém, de forma declarativa, toda a informação necessária para que os BCS consigam ligar-se a um sistema externo e obter a informação que se encontra armazenado no mesmo.

## Lob System

No contexto dos BCS, o Lob System (ou Line-of-Business System) refere-se ao sistema externo no qual está armazenada a informação a que se pretende aceder. Este sistema pode ser uma base de dados relacional, ou qualquer outro sistema que exponha essa informação através de web services ou serviços WCF.

## External Content Type

O External Content Type (ECT) é o conceito central e mais importante dos BCS, uma vez que descreve uma entidade de negócio, ou seja, descreve a estrutura e comportamento da informação a que se pretende aceder. Exemplos de ECTs podem ser Cliente, Factura ou Colaborador.

Na definição de um ECT é especificada a estrutura e o

comportamento da entidade, ou seja:

• Os campos que constituem uma instância da entidade, e respectivo tipo de dados. Por exemplo: Nome, Morada ou País.

• O mapeamento destes campos para objectos utilizados pelas aplicações Office cliente. Por exemplo: o campo Nome da entidade corresponde ao campo FullName no Outlook.

• Os métodos que devem ser invocados pelos BCS para ler, criar, actualizar e apagar instâncias da entidade. Estes métodos pode corresponder, por exemplo, a stored procedures, queries SQL ou web services.

No contexto de um Modelo, podem ser definidas várias entidades, ou seja, vários External Content Types.

## Métodos

Os métodos são abstracções da API do sistema externo e permitem ao BDC Service saber que stored procedures ou web services devem ser chamados para manipular a informação. A definição de um método é feita no contexto de uma entidade e baseia-se sempre num dos estereótipos disponibilizados pelos BCS. Existem cerca de 20 estereótipos possíveis, como sejam, Updater (para actualizar um item), Finder (para listar itens), SpecificFinder (para obter um item específico), Deleter (para apagar um item) entre outros.

## Filtros

Os filtros descrevem os parâmetros que podem ser passados para os métodos na definição de cada entidade. Existem 18 tipos de filtros que podem ser utilizados nos métodos e que indicam ao BDC Service que informação deve ser passada para os mesmos. Exemplos de filtros são o UserName que permite passar o username do utilizador em contexto, ou o Limit que define o número máximo de itens a retornar numa chamada.

## Associação

Uma associação, designada por association, corresponde a uma relação entre duas entidades (External Content Types). No entanto, uma vez que não há garantia que a fonte de dados seja uma base de dados relacional, a associação requer a existência de métodos que permitam obter, a partir de uma entidade, elementos da entidade relacionada.

## Exemplo de Modelo

Felizmente, na grande maioria dos casos, não precisamos de editar o modelo manualmente já que as ferramentas fornecidas pela Microsoft permitem fazer grande parte da configuração de forma visual. No entanto, apenas a título de exemplo, um ficheiro de modelo tem o aspecto apresentado abaixo.

Figura 2 - Exemplo parcial de modelo BDC

## Apresentar Informação Externa

Mas toda esta complexidade tem um objectivo – o de permitir apresentar e manipular a informação armazenada em sistemas externos – por isso não faltam várias alternativas para o fazer.

## External List

A External List é um novo tipo de lista no SharePoint 2010 que, através da associação a um External Content Type, permite visualizar e manipular a informação desse ECT como se esta estivesse armazenada numa lista de SharePoint normal. Na realidade, a informação continua a

residir no sistema externo e é lida e manipulada em tempo real.

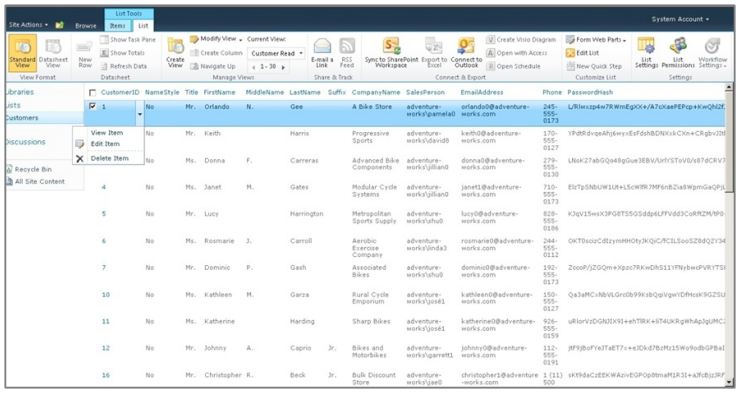


Figura 3 - External List

A grande vantagem das External Lists é que se parecem e comportam exactamente como listas normais e, adicionalmente, o object model do SharePoint trata-as como se assim fossem, permitindo aos developers ler e escrever itens como se estes estivessem armazenados no próprio SharePoint.

Por outro lado, nem tudo funciona exactamente como nas listas tradicionais. Em particular:

- Workflows
- Alertas
- Pastas (folders)
- Anexos (attachments)
- Feeds RSS
- Exportação para Excel

Para que um ECT possa ser utilizado numa External List este tem que definir, pelo menos, os métodos Finder (listar itens) e SpecificFinder (obter um item específico). Isto permitirá à External List apresentar a lista de itens e ver o detalhe de cada um. Adicionalmente, se o ECT possuir métodos Updater (actualizar um item), Deleter (eliminar um item) e Creator (criar um novo item), a External List disponibilizará as acções correspondentes.

## External Data Column

A External Data Column já existia no SharePoint 2007 e, embora tenha sido ligeiramente melhorada no SharePoint 2010, o seu objectivo é o mesmo – permitir utilizar informação externa numa coluna de uma lista. O funcionamento é semelhante ao de uma coluna lookup, permitindo ao utilizador seleccionar um dos itens retornados pelo ECT.

| Customer | Customer: FirstName | Customer: MiddleName | Customer: LastName |
|---|---|---|---|
| Bulk Discount Store | Christopher | R. | Beck |

Figura 4 - External Data Column

Uma das vantagens das External Data Columns é a possibilidade de serem utilizadas também pelo Word 2010, permitindo ao utilizador seleccionar um item exposto através de BCS e utilizando essa informação nos documentos.

Tal como para as External Lists, para utilizar um ECT numa External Data Column este tem que definir, pelo menos, os métodos Finder e SpecificFinder.

## Business Data Web Parts

As Business Data Web Parts são, como o nome indica, um conjunto de web parts que conseguem ligar-se a fontes de dados externas através de um ECT e apresentar essa informação no SharePoint. Estas web parts também já existiam no SharePoint 2007 mas foram melhoradas no SharePoint 2010, nomeadamente permitindo fazer cache da informação externa para melhorar o desempenho.

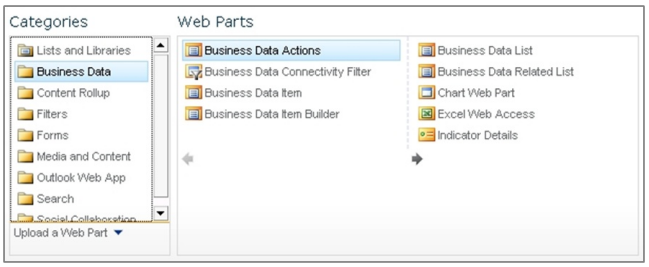



Figura 5 - Business Data Web Parts

As Business Data Web Parts utilizam XSLT para apresentar a informação, o que lhes dá uma enorme flexibilidade no que respeita ao seu aspecto gráfico bem como a possibilidade de edição através do SharePoint Designer 2010.

As web parts incluídas neste pacote são:

- Business Data List – permite listar instâncias (itens) de uma entidade (ECT).

• Business Data Item – permite apresentar o detalhe de uma instância (item) de uma entidade (ECT).

• Business Data Item Builder – permite utilizar parâmetros da query string para criar uma instância (item) de uma entidade (ECT) que pode depois ser utilizada para alimentar outras web parts, nomeadamente a Business Data Item web part.

• Business Data Related List – permite listar instâncias (itens) de uma entidade (ECT) relacionada. É especialmente útil para apresentar informação em cenários de Master/Detail.

• Business Data Connectivity Filter – permite filtrar a informação proveniente de um ECT antes que seja consumida por outra web part, como a Business Data List web part.

• Business Data Actions – apresenta as acções disponíveis para uma instância (item) de uma entidade (ECT).

## Pesquisa

Um dos maiores benefícios oferecidos pelos BCS é a possibilidade de indexar e realizar pesquisas sobre a informação externa exposta através dos ECTs como se se tratasse de informação armazenada em listas no SharePoint.

Para que um ECT seja indexável é necessário que defina, pelo menos, os métodos IDEnumerator e SpecificFinder. O primeiro permitirá ao SharePoint obter os IDs de todos os itens e o segundo obter o detalhe de cada um. Adicionalmente, o modelo tem que ter a propriedade ShowInSearchUI para que o SharePoint o consiga indexar.

Mas isto é para que a informação seja indexada. Para que, ao realizar uma pesquisa, o utilizador consiga clicar sobre um dos resultados e visualizar informação detalhada sobre o resultado que seleccionou, precisamos também de configurar a Profile Page de cada ECT indexado.

Uma Profile Page não é mais que uma página em SharePoint com algumas web parts que recebe o identificador de um item na query string e apresenta informação detalhada sobre esse item, incluíndo itens de ECT relacionados (através de Associations).

As Profile Pages são configuradas na Central Administration, acedendo à gestão do Business Data Connectivity Service. A única informação que precisamos de fornecer é o endereço URL do site onde estas páginas serão automaticamente criadas e o SharePoint fará o resto por nós.

Neste ponto, basta-nos apenas dizer ao Search Service que deve indexar uma nova Content Source do tipo Line of Business Data e efectuar um Full Crawl. Após a conclusão do crawl a informação externa passa a estar disponível para ser pesquisada e a informação de cada item será apresentada na respectiva Profile Page.

## User Profiles

Utilizando os BCS, o SharePoint 2010 consegue utilizar fontes de dados externas para complementar a informação dos User Profiles. Para isso basta que seja possível mapear User Profiles com itens de um ECT, utilizando um campo de cada lado.

Não é possível configurar um ECT como fonte principal para a sincronização de perfis, mas é possível que uma sincronização com Active Directory seja complementada com informação proveniente de um ECT.

## Integração com Office Client

A integração da informação externa nas aplicações do Office 2010 é outra das novidades do SharePoint 2010 relacionada com os Business Connectivity Services. Até agora, este tipo de funcionalidade só era possível com desenvolvimentos à medida de razoável complexidade.

Com os BCS é possível apresentar a informação externa nas aplicações Office, utilizá-la em cenários offline e, em determinados casos, actualizar a informação directamente na fonte de dados externa. Contudo, nem todas as aplicações incluídas no Office 2010 suportam esta integração nativamente. De momento apenas o Outlook 2010, o Word 2010, o Access 2010, o InfoPath 2010 e o SharePoint Workspace 2010 conseguem fazê-lo, sendo que cada uma das aplicações utiliza esta tecnologia de forma diferente.

## Outlook 2010

O Outlook 2010 é uma das aplicações que tira melhor partido das funcionalidades cliente dos BCS. Para que seja possível visualizar a informação exposta através de um ECT no Outlook 2010 são necessários dois passos na configuração desse ECT:

1. Definir qual o tipo de informação exposto pelo ECT, de entre os tipos de informação manipulados pelo Outlook: Contactos (Contacts), Tarefas (Tasks), Eventos (Appointments) ou Artigos (Posts). Esta configuração pode ser feita através do SharePoint Designer ou directamente no XML do Modelo.

2. Mapear os campos do ECT com os campos do Outlook para esse tipo de informação. Por exemplo, indicar quais os campos do ECT que correspondem aos campos Last Name, First Name, E-mail Address e outros, no Outlook.

Existindo uma External List que exponha a informação do


ECT, passa a ser possível utilizar o botão Connect to Outlook disponibilizado pela ribbon da lista. Ao pressionar o botão, o SharePoint vai analisar a especificação do ECT e vai incluí-lo num pacote de instalação Click Once que é imediatamente instalado no Outlook 2010 do utilizador como um Add-In.

Uma vez instalado o pacote, a lista aparece na interface do Outlook permitindo ao utilizador interagir com a informação externa como se fossem contactos, tarefas, eventos ou artigos normais. Caso o ECT defina os métodos necessários, é ainda possível utilizar o Outlook para actualizar a informação da fonte de dados externa. Todos os campos expostos pelo ECT que não estejam mapeados em campos do objecto Outlook, são mostrados numa secção própria do detalhe desse objecto e podem também ser actualizados.

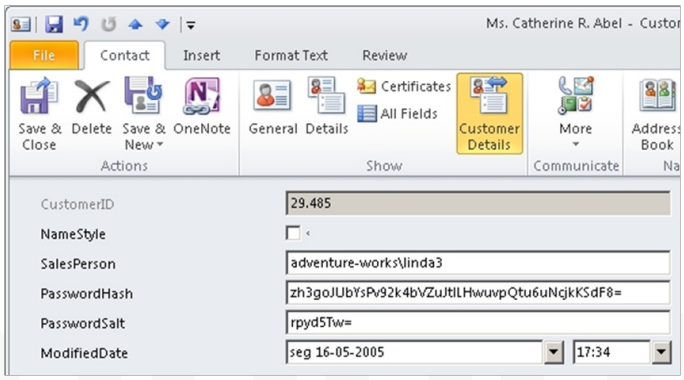



Figura 6 - Informação adicional (não mapeada) do ECT

Tal com as restantes aplicações Office, o Outlook tira partido de um mecanismo de cache e sincronização da informação permitindo ao utilizador trabalhar sobre esta em offline e sincronizando-a automaticamente assim que o acesso ao sistema externo fica disponível.

## Word 2010

O Word 2010 é outras das aplicações Office que tem suporte nativo para os BCS. No entanto, os cenários para aplicação desta tecnologia são diferentes dos disponíveis para Outlook. A utilização dos BCS em Word 2010 limita-se à inserção de informação proveniente de fontes de dados externas em documentos através de Quick Parts.

Para quem não conhece, as Quick Parts são uma funcionalidade do Word que permite criar campos para preenchimento dinâmico da informação no meio do texto de um documento. Estes campos podem depois ser preenchidos automaticamente com informação proveniente do content type do documento, no SharePoint. Isto inclui informação proveniente de uma External Data Column existente na Biblioteca de Documentos em que o documento está armazenado.

O funcionamento é simples:

1. Numa Biblioteca de Documentos, cria-se uma External Data Column configurando-a para expor a informação de um determinado ECT e definindo os campos do ECT que são expostos.

2. Cria-se um novo documento nessa biblioteca, utilizando o botão New da ribbon.

3. Já no Word, através da ribbon Insert, inserimos uma (ou mais) Quick Part, seleccionando a(s) Document Property(s) que corresponde(m) à informação externa que queremos incluir no documento.

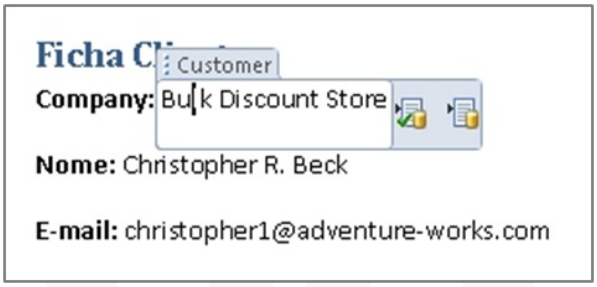


Figura 7 - Quick Parts com informação externa

4. O Word passa então a permitir que o utilizador seleccione um item do ECT, utilizando o Entity Data Picker, e popula automaticamente todos os campos relacionados.

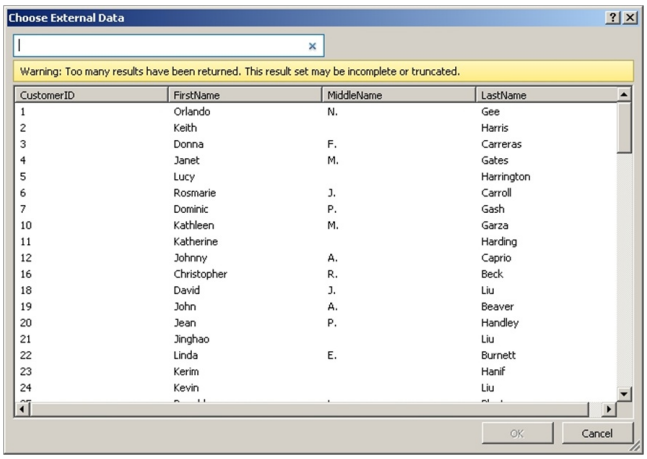



Figura 8 - External Data Picker

## Access 2010

O Access 2010 consegue importar um modelo Business Data Connectivity (BDC) e apresentar a informação externa sob a forma de tabelas. No entanto, as tabelas criadas são read-only, ou seja, não é possível escrever de volta para a fonte de dados externa.

## InfoPath 2010

Quando é criada uma External List, são também gerados forms para inserção, edição e consulta da informação externa. Por omissão, estes forms são gerados como páginas ASP.NET normais mas, utilizando o SharePoint Designer ou a ribbon da External List, é possível criar forms mais inteligentes utilizando InfoPath. Os forms são gerados automaticamente, mas podem depois ser modificados utilizando o InfoPath.

É ainda possível arrastar um External Data Picker para um formulário InfoPath e definir uma External List como fonte de informação, permitindo a leitura e escrita de informação proveniente de fontes de dados externas.

## SharePoint Workspace 2010

O SharePoint Workspace 2010 é a evolução do Groove 2007 e posiciona-se como a ferramenta de acesso offline à informação guardada em SharePoint 2010, incluindo External Lists. Tal como para os restantes tipos de listas, basta clicar no botão Sync to SharePoint Workspace para que o conteúdo das mesmas seja descarregado para a máquina do utilizador ficando disponível quando este está desligado do servidor.


No que respeita aos BCS, o que o SharePoint Workspace faz é descarregar a definição do ECT associado à External List e armazená-la localmente, bem como os forms de inserção, edição e consulta da informação que foram gerados para essa External List. Tal como as restantes aplicações Office descritas, o SharePoint Workspace utiliza a cache local para garantir a disponibilização da informação externa quando o sistema externo não está disponível.

# Soluções e Ferramentas

Uma das grandes queixas de quem utilizou o Business Data Catalog no SharePoint 2007, foi a falta de ferramentas que permitissem uma boa experiência na criação e manipulação da metainformação (modelos). A Microsoft ouviu essas queixas e brindou-nos com duas ferramentas fantásticas para utilizar com os Business Connectivity Services:

- SharePoint Designer 2010
- Visual Studio 2010

## SharePoint Designer 2010

O SharePoint Designer 2010 é uma ferramenta gratuita e obrigatória para qualquer utilizador avançado ou developer de SharePoint. Possui um enorme número de funcionalidades focando-se principalmente na criação de soluções sem código, ou seja, soluções de customização

do SharePoint sem necessidade de desenvolvimentos à medida.

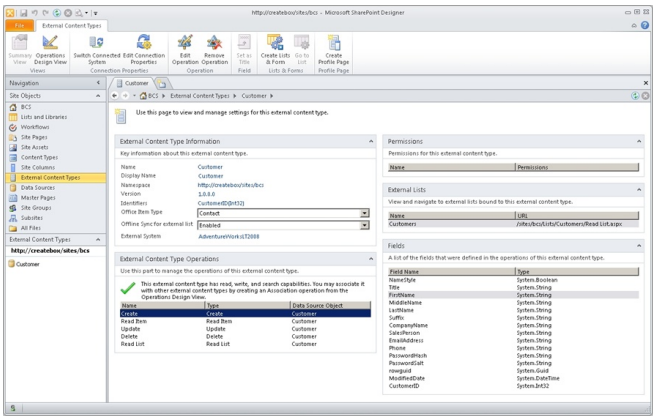


Figura 9 - Utilização do SharePoint Designer para gerir ECTs

No que respeita aos Business Connectivity Services, o SharePoint Designer 2010 permite:

• Criar e manipular External Content Types, incluindo alterar configurações, criar novos métodos e mapear ECTs com objectos Office. Na criação de ECTs apenas é possível efectuar ligações a bases de dados SQL Server, Web Services cujos schemas sejam suportados pelos BCS ou assemblies .Net existentes.

• Criar e configurar External Lists com base em ECTs já criados.

• Gerar e editar formulários InfoPath de suporte às External Lists.

• Utilizar informação externa em workflows.

• Criar web part pages e profile pages

Estas funcionalidades permitem a utilização dos BCS sem qualquer desenvolvimento à medida e adaptam-se às necessidades mais simples e comuns.

## Visual Studio 2010

Com o Visual Studio 2010 podemos criar soluções mais complexas para casos em que as funcionalidades do SharePoint Designer 2010 não são suficientes. Adicionalmente, com o Visual Studio 2010 podemos criar componentes reutilizáveis que depois poderão ser incorporados em soluções através do SharePoint Designer.

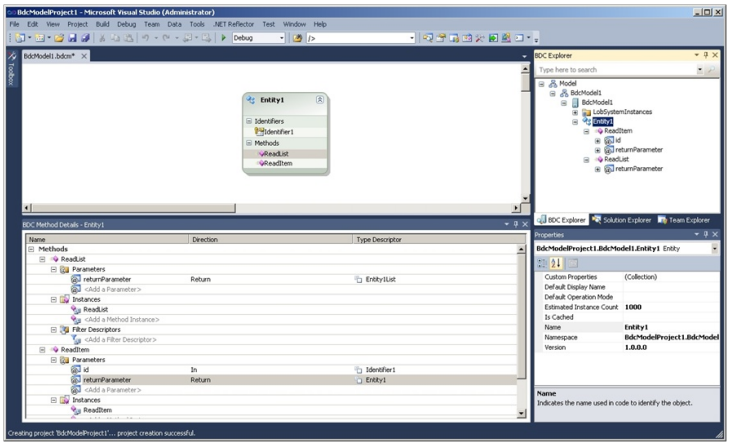


Figura 10 - Editor visual de Modelos BDC no Visual Studio 2010

Alguns dos casos de uso possibilitados pelo Visual Studio 2010 são:

• Criar e manipular External Content Types, utilizando o novo template de projecto Business Data Connectivity Model. Este template inclui um conjunto de designers que permite a edição visual do Modelo e respectivos ECTs, e possibilita ainda o desenvolvimento de soluções utilizando código .Net para acesso a virtualmente qualquer fonte de dados externa.

• Criar componentes reutilizáveis para os BCS utilizando os vários pontos de extensibilidade da API dos BCS, como sejam Code Actions que podem ser utilizadas dentro do Outlook, External Data Parts para utilização em task panes declarativas no Outlook, actividades para workflows e outros.

• Criar Add-Ins para aplicações Office com suporte para BCS, utilizando o object model dos BCS.

• Criar workflows à medida que tiram partido de informação em External Lists ou utilizam o object model dos BCS.

## Tipos de Solução por Ferramenta

A tabela na página seguinte ajuda a seleccionar a ferramenta ideal para cada necessidade.

| Capacidade | SharePoint Designer 2010 | Visual Studio 2010 |
|---|---|---|
| Ligar a... | WCF/WS, SQL Server e Assemblies .Net Existentes | Qualquer fonte de dados através de Assemblies .Net |
| Modelo de desenvolvimento | Descobrir e configurar | Criar e publicar |
| Serve para... | • Modelos simples com interfaces nativas (External Lists, Outlook, SharePoint Workspace, InfoPath, Search) e associações simples baseadas em chaves estrangeiras. | • Modelos complexos com lógica de conectividade à medida para agregação, transformação, segurança, etc.<br>• Interfaces à medida via customizações Office.<br>• Controlos reutilizáveis para servidor e cliente que se ligam a dados externos. |
| Limitações | • Fonte de dados tem que expor interface com formato suportado.<br>• Não suporta esterótipos avançados (operações em massa).<br>• Não suporta serviços genéricos ou polimórficos.<br>• Apenas associações por chave estrangeira. | • Visual designer apenas funciona para modelos baseados em objectos .Net.<br>• Desenvolvimento e *packaging* separados para componentes cliente e servidor. |

## Casos de Uso

Uma das perguntas mais frequentes relacionadas com a utilização de Business Connectivity Services é quais os casos de uso desta tecnologia ou, de outra maneira, quando devo utilizar os BCS.
Alguns dos casos de uso mais comuns para a utilização de BCS são:

• Necessidade de apresentar informação de uma base de dados SQL Server. Utilizando BCS é possível apresentar e, caso seja necessário, modificar a informação utilizando External Lists sem ser preciso desenvolver uma única linha de código à medida. É um back-office instantâneo.

• Necessidade de complementar os User Profiles dos utilizadores do domínio com informação proveniente do sistema de gestão de Recursos Humanos ou ERP. Tal como já foi falado, os BCS permitem responder a este requisito permitindo configurar um ECT como fonte de dados adicional para a sincronização de perfis do SharePoint.

• Necessidade de sincronizar contactos que estão armazenados num sistema de negócio ou ERP. Utilizando os BCS é possível definir um ECT que expõe esses contactos através de uma External List e ligá-la ao Outlook onde serão geridos como contactos normais. Este cenário permite ainda que os utilizadores tenham acesso aos contactos mesmo quando estão fora do escritório, em modo offline.

• Necessidade de apresentar informação proveniente de fontes de dados distintas. Utilizando o conector para assemblies .Net e desenvolvendo um ECT com o Visual Studio 2010, podemos construir cenários de acesso a múltiplas fontes de dados com agregação dos mesmos numa única entidade.

• Necessidade de indexar e pesquisar informação residente num sistema de negócio ou ERP. Os BCS permitem ao serviço de pesquisa do SharePoint indexar conteúdos expostos através de ECTs e pesquisá-los como se a informação estive armazenada no SharePoint.
Há muitos outros cenários onde os BCS podem ser úteis, por vezes apenas como um dos componentes da solução.

## Funcionalidades por Versão do SharePoint

A infraestrutura utilizada pelos Business Connectivity Services está disponível em todas as versões do

| Funcionalidade | SharePoint Foundation 2010 | SharePoint Server 2010 Standard | SharePoint Server 2010 Enterprise |
|---|---|---|---|
| BDC Service | ✓ | ✓ | ✓ |
| Connector Framework | ✓ | ✓ | ✓ |
| External List | ✓ | ✓ | ✓ |
| External Data Column | ✓ | ✓ | ✓ |
| Secure Store Service | | ✓ | ✓ |
| External Data Search | | ✓ | ✓ |
| Profile Pages | | ✓ | ✓ |
| Business Data Web Parts | | | ✓ |
| Integração com Office Cliente | | | ✓ |

SharePoint, incluíndo o SharePoint Foundation 2010. No entanto, nem tudo vem incluído na versão gratuita. A tabela abaixo ajuda a clarificar quais as funcionalidades que estão incluídas em cada uma das versões do SharePoint 2010.

O suporte para Business Connectivity Services em aplicações Office requer o Microsoft Office 2010 Professional Plus, ou superior.

## Links Úteis

Aqui ficam alguns links úteis para quem está agora a começar e quer saber mais sobre Business Connectivity Services.

Microsoft Business Connectivity Services Team Blog
O blog official da equipa que desenvolveu os BCS, com imensos artigos com vários níveis de complexidade. Obrigatório para todos os interessados nesta tecnologia.
http://blogs.msdn.com/b/bcs/

BCS Team Channel
O canal no YouTube onde alguns vídeos foram publicados pela equipa de produto.
http://www.youtube.com/user/MOSSBCSTeam

Connecting to a .NET Framework Source Using Business Connectivity Services in Office 2010
Visual How To sobre como desenvolver um ECT usando Visual Studio 2010 para obter dados de uma fonte externa.
http://msdn.microsoft.com/en-us/library/ff394331(office.14).aspx

Microsoft Business Connectivity Services
Secção do SDK do SharePoint 2010 dedicada aos Business Connectivity Services.
http://msdn.microsoft.com/en-us/library/ee556826.aspx

Business Connectivity Services Resource Center
Resource Center dedicado aos Business Connectivity Services, no TechNet.
http://technet.microsoft.com/en-us/sharepoint/ee518675.aspx

Business Connectivity Services: Technical Articles
Artigos técnicos da MSDN relacionados com Business Connectivity Services.
http://msdn.microsoft.com/en-us/library/gg481768.aspx

Link para o artigo: http://tinyurl.com/RPED28-04

## AUTOR

Escrito por **André Vala**
Licenciado e Mestre em Engenharia Informática e de Computadores pelo Instituto Superior Técnico, é actualmente consultor sénior na |create|it| e co-fundador da Comunidade Portuguesa de SharePoint .
Autor do blog http://blogit.create.pt/blogs/andrevala, trabalha com SharePoint desde 2006, altura em que surgiu a primeira versão beta do SharePoint 2007.
Tem participado em vários projectos nacionais e internacionais sobre SharePoint, e participa frequentemente como orador em eventos da Microsoft relacionados com o mesmo tema.

# A PROGRAMAR

Lua – Linguagem de Programação (Parte 8)
jQuery 1.5 e Ajax
Datagrid em Windows Presentation Foundation
Planos de Execução em ORACLE
Smarty PHP Template Engine

# Lua – Linguagem de Programação (Parte 8)

Este artigo trata o uso de operações de aleatoriedade e a manipulação de cadeias (operações de detecção de tamanho de cadeias – revisão, repetição de caracteres, separação de cadeias, busca e substituição de caracteres, conversão em modo ASCII).

## ERRATA

Por falha pessoal no artigo anterior (Parte 7) ficou indicado após a conclusão que esta parte trataria do tema: arquivo. No entanto, este assunto fora apresentado na sexta parte desta série de artigos.

## ALEATORIEDADE

É sabido que aleatoriedade é a característica do que é indeterminado ou incerto. Uma das possibilidades operativas de uma linguagem de programação é a capacidade de "gerar" valores numéricos aleatórios. O termo: gerar é grafado entre aspas devido a característica que os computadores possuem de fazer este trabalho de uma forma considerada não real, ou seja, por meio de uma acção considerada pseudo-aleatória.

Para esta acção em linguagem Lua há as funções de geração de números aleatórios: math.randomseed() e math.rendom().

Os valores gerados por estas funções são valores pseudo-aleatórios, e necessitam ser usados com alguma cautela, tanto que há no manual de referência da linguagem Lua a advertência: Nenhuma garantia pode ser dada para suas propriedades estatísticas.

A função math.randomseed(n) faz uso do valor "n" como parâmetro de semente para a geração de valores aleatórios.

A função math.random([n[,m]]) pode ser usada de três formas diferentes: usada sem parâmetros o que fará a geração de valores entre 0 e 1, pode ser usada apenas com o parâmetro "n" para gerar valores inteiros entre 1 e o valor estabelecido junto a parâmetro "n" e pode ainda ser usada com os parâmetros "n" e "m" para gerar valores inteiros entre "n" e "m".

O programa seguinte efectua a acção de geração de valores aleatórios da forma mais simples possível.

```
-- inicio do programa ALEAT01

  math.randomseed(0)

  local function SORTEIO()
    N = math.random()
    return N
  end

  for I = 1, 5 do
    X = SORTEIO()
    print(X)
  end

-- fim do programa ALEAT01
```

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome aleat01.lua e execute-o com a linha de comando lua 5.1 aleat01.lua.

A função math.randomseed() necessita ser usada a frente da função math.random() para que math.random() consiga gerar os valores aleatórios.

Ao executar o programa várias vezes os valores apresentados como saída sempre serão:

0.0011597033600879
0.23557237464522
0.64815210425123
0.074373607593005
0.27024140140996

Os valores se repetem pelo fato de estar sendo utilizado o valor de semente "0" (zero). Se o valor de semente for mudado para 1, 2, 3 ou outro valor qualquer serão

conseguidos valores diferentes. No entanto para um mesmo valor de semente para mais de uma execução ter-se-á a apresentação dos mesmos valores. Uma forma de mudar um pouco este comportamento é fazer uso da função os.time() como valor de semente. A função os.time() retorna o valor do tempo corrente. Assim sendo, observe o código de programa seguinte:

```
-- inicio do programa ALEAT02

  math.randomseed(os.time())

  local function SORTEIO()
    N = math.random()
    return N
  end

  for I = 1, 5 do
    X = SORTEIO()
    print(X)
  end

-- fim do programa ALEAT02
```

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome aleat02.lua e execute-o com a linha de comando lua 5.1 aleat02.lua.

Ao se executar o programa várias vezes notar-se-á que os resultados apresentados são levemente diferentes. Por exemplo, a seguir apresenta-se os valores de saída de duas execuções sequenciais do programa:

0.85509811700797
0.76201666310617
0.21799371318705
0.17670216986602
0.24967192602313

0.85558641315958
0.40208136234626
0.94387646107364
0.84850611896115
0.82357249671926

Observe que uso da função os.time() como valor semente permite um comportamento de aleatoriedade mais convincente.

Agora imagine que se queira sortear valores numéricos entre 1 e 5. Assim sendo, observe o código seguinte:

```
-- inicio do programa ALEAT03

  math.randomseed(os.time())

  local function SORTEIO()
    N = math.random(1,5)
    return N
  end

  for I = 1, 5 do
    X = SORTEIO()
    print(X)
  end

-- fim do programa ALEAT03
```

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome aleat03.lua e execute-o com a linha de comando lua 5.1 aleat03.lua.

Execute o programa algumas vezes. Note que o primeiro valor é sempre o mesmo em toda a execução, somente os demais valores são apresentados diferentemente. Este é um comportamento operativo da linguagem Lua que gera dúvidas nos iniciantes no uso desta linguagem. Não se preocupe em seguida será mostrado como contornar este tipo de ocorrência.

Observe que para gerar valores entre 1 e 5 fora usado: N = math.random(1,5).

O próximo programa mostra como contornar o problema de repetição do primeiro valor da sequência sorteada.

```
-- inicio do programa ALEAT04

  math.randomseed(os.time())
  math.random()

  local function SORTEIO()
```

```
    N = math.random(1,5)
    return N
  end

  for I = 1, 5 do
    X = SORTEIO()
    print(X)
  end

-- fim do programa ALEAT04
```

Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome aleat04.lua e execute-o com a linha de comando lua 5.1 aleat04.lua.
Execute o programa algumas vezes e note a diferença nesta versão. Observe o uso da função math.random() logo após o uso da função math.randomseed(os.time()). Este pequeno ajuste faz o acerto desejado.

## MAIS MANIPULAÇÃO DE CADEIAS

No sexto artigo desta série fora apresentada uma forma de detecção da quantidade de caracteres de uma cadeia com o uso do operador # por meio da instrução de código print(#"Linguagem Lua") que mostra como resultado o valor 13.
Esta mesma acção pode ser efectuada por meio da função string.len(texto) como apresentado no quinto artigo desta série, onde texto é a indicação da cadeia que terá contada a quantidade de caracteres.
Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome cadeia01.lua e execute-o com a linha de comando lua 5.1 cadeia01.lua.

```
-- inicio do programa CADEIA01

   X = "Linguagem Lua"
   print(string.len(X))

-- fim do programa CADEIA01
```

Nas operações de manipulação de cadeias há ainda a possibilidade de separar partes de uma cadeia. Para este efeito faz-se uso da função string.sub(texto, início, [fim]), onde texto é a indicação da cadeia que será separada, início é a indicação da posição de separação inicial que pode ser positiva ou negativa e fim - indicação opcional da posição final de separação.
Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome cadeia02.lua e execute-o com a linha de comando lua 5.1 cadeia02.lua.

```
-- inicio do programa CADEIA02

   X = "COMPUTADOR"

   print(string.sub(X))
   print(string.sub(X, 1))
   print(string.sub(X, 1, 3))
   print(string.sub(X, 4, 5))
   print(string.sub(X, 6, 7))
   print(string.sub(X, 8))
   print(string.sub(X, -5))

-- fim do programa CADEIA02
```

Após a execução serão apresentados os textos COMPUTADOR, COM, PU, TA, DOR e TADOR.
Outro factor de manipulação de cadeias de caracteres é a realização de operações de substituição de caracteres de uma cadeia. Para tanto, use a função de substituição string.gsub(texto, busca, troca, vezes), onde texto é a cadeia de texto definida, busca é o carácter a ser localizado, troca é o carácter que será substituído e vezes indica o número máximo de substituições a serem efectuadas, sendo este último argumento opcional.
Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome cadeia03.lua e execute-o com a linha de comando lua 5.1 cadeia03.lua.

```
-- inicio do programa CADEIA03
   X = "A BOLA AZUL APARECEU"
   print(string.gsub(X,"A","4"))
   print(string.gsub(X,"A","X",2))
-- fim do programa CADEIA03
```

Observe que após a execução, o programa mostra além da troca realizada o número de trocas realizadas.
Outra acção para manipulação de cadeias é a função string.rep(texto, vezes), onde texto é a cadeia a ser repetida e vezes é a definição do número de repetições.
Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome cadeia04.lua e execute-o com a linha de comando lua 5.1 cadeia04.lua.

```
-- inicio do programa CADEIA04

   X = "OBA "
   print(string.rep(X,2))

-- fim do programa CADEIA04
```

Outro efeito com cadeias é a obtenção do código ASCII dos caracteres que formam a cadeia com a função string.byte(texto, início, [fim]), onde texto é a indicação da cadeia que terá seus caracteres convertidos em formato ASCII, início é a indicação da posição de separação inicial que pode ser positiva ou negativa e fim - indicação opcional da posição final de separação.
Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome cadeia05.lua e execute-o com a linha de comando lua 5.1 cadeia05.lua.

```
-- inicio do programa CADEIA05

   X = "ABCDEF"
   print(string.byte(X, 1))
   print(string.byte(X, 1, 3))
   print(string.byte(X, 4, 5))
   print(string.byte(X, 1, 6))
   print(string.byte(X, -5))

-- fim do programa CADEIA05
```

O programa apresenta os valores:
- 65
- 65 66 67
- 68 69
- 65 66 67 68 69 70
- 66

A acção inversa é conseguida com o uso da função string.char(código1, código2, …, códigoN), onde cada argumento usado é um valor ASCII.
Em seguida escreva o código de programa em um editor de texto, gravando-o com o nome cadeia06.lua e execute-o com a linha de comando lua 5.1 cadeia06.lua.

```
-- inicio do programa CADEIA06

   print(string.char(65))
   print(string.char(65, 66, 67))

-- fim do programa CADEIA06
```

## CONCLUSÃO

Neste artigo o enfoque foi o uso dos recursos de geração de valores aleatórios e alguns detalhes sobre a manipulação de cadeias.
No próximo artigo a ênfase será dada a criação e uso de módulos em linguagem Lua.

Link para o artigo: http://tinyurl.com/RPED28-06

## AUTOR

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

# jQuery 1.5 e AJAX

O objectivo deste artigo é expor a funcionalidade de AJAX que o jQuery inclui, ao detalhe, e é também falar sobre as novas funcionalidades introduzidas pela versão 1.5 da framework, neste caso, os Deferreds.
Nesta nova versão toda a funcionalidade de AJAX foi redesenhada, pelo que iremos entrar no tema das novas funcionalidades através da sua utilização no próprio AJAX e depois expandindo a outras alterações também com relevância.

Começando pelo princípio, o método mais simples de efectuar um pedido AJAX em jQuery é utilizando a função jQuery.get:

```
var ajaxObj = $.get('ajax/mypage.aspx', function(data) {
  $('#ajaxDiv').html(data);
  alert('callback called!');
});
```

Este é o método mais simplificado, especificamos unicamente que url irá retornar os dados, e a função de callback retorna-nos os dados e aí poderemos adicionar qualquer lógica necessária.
Os dados que retornam do nosso pedido podem ser texto, JSON, XML ou JavaScript, e a função infere o tipo, pois neste caso não o estamos a especificar. Além da variável data, poderíamos especificar outras duas variáveis na função de callback, a segunda seria o textStatus do XHR (XMLHttpRequest) e a terceira seria o mesmo que o ajaxObj irá conter, um jqXHR (que passou a ser um jqXHR a partir da versão 1.5, anteriormente era um XHR nativo).
Neste exemplo caso retornássemos HTML seria adicionado ao DOM como innerHTML do objecto com o id ajaxDiv e mostraria um alert, depois do pedido retornar com sucesso.
O objecto jqXHR implementa o interface de Promises (que iremos descortinar mais à frente na funcionalidade Deferreds do jQuery 1.5) e inclui toda a sua funcionalidade, pelo que inclui os métodos error() success() e complete() para acordar com os callbacks da função $.ajax (que também iremos rever mais à frente) que aceitam uma função como parâmetro que é chamada quando o pedido terminar a sua execução, ou mesmo que estes callbacks sejam assignados após o pedido AJAX ter sido executado, estas são chamadas de qualquer modo, esta é uma das novidades deste interface de Promises, permite assignar callbacks à posteriori, o que não era possível nas versões anteriores às 1.5.
Podemos ver aqui o exemplo de como assignar estes callbacks, e verificar que mesmo após o pedido ser completamente executado, assignando novos callbacks, estes executam de qualquer modo:

```
/*Assignar handlers imediatamente após executar o pedido e
guardar numa var o objecto jqXHR*/
   var xhrObj = $.get("ajax.aspx", function()
{
    alert("sucesso!");
  })
     .success(function() { alert("novamente sucesso"); })
  .error(function() { alert("erro"); })
     .complete(function() { alert("pedido completo"); });

//alguma lógica adicional (...)

/*adicionar outro callback de complexão aqui, e verificar que é executado mesmo que o pedido já tenha sido completamente efectuado anteriormente, devido às funcionalidades das Promises*/
   xhrObj.complete(function(){ alert("completo novamente"); });
```

Em versões anteriores do jQuery, no caso de utilizarmos esta função get(), se existisse um erro não conseguiríamos assignar um callback a não ser através da função global

ajaxError(), ou seja, não conseguiríamos ter um error handling local e objectivo, a não utilizando uma função mais genérica com a ajax().
Uma ressalva, os pedido efectuados com a função get() que não sejam pedidos JSONP ou Script, não permitem cross-domain, como é usual.
Se quisermos efectuar outro tipo de pedidos com a função get() :

```
//fazer apenas o request e ignorar resultado
$.get("ajax.aspx");
//passar parâmetros simples e ignorar resultados
$.get("ajax.aspx", { tipo: "noticias", quantas: "10" } );
//passar arrays e ignorar resultados
$.get("ajax.aspx", { 'valores[]': ["10", "20"]} );
//combinar parâmetros com callback
$.get("ajax.aspx", { param1: "teste" }, function(data) {
        alert("callback executado!");
});
//receber um JSON já parsed
$.get("ajax.aspx", { param1: "teste" }, function(data) {
     alert(data.prop1); // valor da variável data: { "prop1": "valor1" }
});
```

Outra das funcões para efectuar pedidos AJAX é a load():

```
//carrega o resultado no/s objecto do DOM especificado/s pelo selector
$('#ajaxDiv').load('ajax.aspx', function() {
  alert('HTML carregado');
});
//carrega o resultado no/s objecto do DOM especificado/s pelo selector, mas apenas o que faz match com o selector passado ao lado do url
$('#ajaxDiv').load('ajax.aspx #mainContent');
```

Também existe a possibilidade de enviar parâmetros, como o segundo parâmetro,à semelhança do get().
Existe também a função post() que funciona do mesmo exacto modo que a get() mas ao invés de enviar os dados por HTTP GET, envia precisamente por HTTP POST.

Caso o nosso objectivo seja exclusivamente obter JSON, existe uma função específica para tal, a getJSON(), que tem algumas especificidades, tais como no caso de especificarmos adicionar ao url o texto callback=? O pedido passa a ser tratado com um pedido JSONP, e não JSON, o que permite pedidos cross domain sem qualquer problema. O segundo parâmetro pode ser utilizado para enviar parâmetros, como nas outras funções.

```
$.getJSON('outputjson.json', function(data) {
  $('.result').html('<p>' + data.foo + '</p>'
    + '<p>' + data.baz[1] + '</p>');
});
//estrutura de JSON esperada:
{
   "foo": "The quick brown fox jumps over the lazy dog.",
  "bar": "ABCDEFG",
  "baz": [52, 97]
}
```

Passando à função mais completa e talvez a mais utilizada, a função ajax(), podemos definir o url, e imensos settings, vou passar aqui pelos mais importantes:

**async:** permite definir se o pedido é ou não executado assíncronamente;
beforeSend(jqXHR, settings): este callback é executado imediatamente antes do pedido ser executado, e caso retornemos false, o pedido não é executado;
**complete(jqXHR, textStatus):** este callback é executado quando o pedido foi completamente executado, a partir da versão 1.5 podemos passar aqui um array de funções que serão todas executadas;
**data:** permite passar parâmetros no formato querystring (valor1=X&valor2=y...);
**dataType:** permite definir exactamente que tipo de dados iremos receber, json, script, text, html, jsonp, xml (podemos

passar múltiplos valores, por exemplo “jsonp xml”, para efectuar um pedido jsonp e converter para XML);
**error(jqXHR, textStatus, error):** callback em caso de erro;
**statusCode:** definir um callback conforme o HTTP error code:

```
$.ajax({
  statusCode: {404: function() {
    alert('page not found');
  }
});
```

**success(data, textStatus, jqXHR):** callback executado quando o pedido é retornado com sucesso;
**type:** tipo de pedido “GET” ou “POST”;
**url:** URL do pedido.

Podemos utilizar a função ajaxSetup() para definir estes settings globalmente na nossa aplicação, sendo que depois podemos fazer override em cada caso aos settings que se alteram, centralizando tudo o que são settings transversais. Exemplos:

```
//obter um script via AJAX (GET)
$.ajax({
   type: "GET",
   url: "my.js",
   dataType: "script"
});

//fazer o pedido por POST, enviando parámetros e com callback
$.ajax({
   type: "POST",
   url: "ajax.aspx",
   data: "nome=Ricardo&location=Lisboa",
   success: function(msg){
     alert("Dados enviados: " + msg);
   }
});

/*pedir a última versão de uma página, especificando que não queremos que o browser persista qualquer cache*/
$.ajax({
  url: "teste.html",
  cache: false,
  success: function(html){
    $("#resultado").append(html);
  }
});

/*efectuar um pedido que ao estando o seu resultado a ser utilizado de imediato para assignar à  variável html, devemos especifica que não pode ser assíncrono, pois caso contrário poderíamos tentar usar a variável html e esta não iria ter o valor esperado.*/
var html = $.ajax({
  url: "page.aspx",
  async: false
 }).responseText;

/*o mesmo caso que o anterior, mas aqui enviamos parâmetros e temos um callback de sucesso, e o o dataType é especificado.
Ao utilizar o global a false, estamos a dizer explicitamente que os eventos globais de ajax não vão ser disparados, logo os seus callbacks não vão executar, isto caso estejam definidos via ajaxStart() e ajaxStop()*/
var bodyContent = $.ajax({
      url: "script.aspx",
      global: false,
      type: "POST",
      data: ({id : this.getAttribute('id')}),
      dataType: "html",
      async:false,
      success: function(msg){
         alert(msg);
      }
   }
).responseText;
```

## jQuery 1.5 e AJAX

Com esta especificação extensa do AJAX, vamos passar às novas funcionalidades do jQuery 1.5, começando pelos já mencionados Deferreds (Promises interface).
Esta funcionalidade tem como objectivo fazer com que uma tarefa e a lógica executada após esta estar completa sejam desacoplados, quer isto dizer que podemos assignar múltiplos callbacks para o resultado de uma tarefa e mesmo após esta estar completa podemos continuar a adicioná-los e estes são executados do mesmo modo. Esta tarefa pode ser assíncrona ou não, nada obriga que o seja.
Visto que o AJAX do jQuery 1.5 foi redesenhado para incluir os Deferreds, podemos usufruir deles directamente:

```
// este pedido é assíncrono por omissão
var req = $.get('foo.htm')
   .success(function(response) {
      //em caso de sucesso
   })
   .error(function() {
      //em caso de erro
   });

//isto até pode ser executado antes do get acima
algumaFuncao();

/*definir algo mais a ser executado em caso de sucesso, que pode ou não já ter ocorrido, mas com os deferreds realmente não interessa, é executado de qualquer forma*/
req.success(function(response) {
    /*tomar alguma acção com a resposta isto vai ser executado quando o sucesso ocorrer, ou caso este já tenha ocorrido, é disparado de imediato, caso os outros callbacks de sucesso já tenham sido executados
   */
})
```

Deste modo podemos ver que já não estamos limitados a definir apenas um callback para error, sucesso e complexão, podemos definir quantos quisermos, e mais importante, quando quisermos!

Como podemos ver, deste modo podemos organizar o código de maneira diferente, até podemos criar uma abstracção à função de ajax no contexto da nossa aplicação e ter funções para atribuição de callbacks, que são executados numa metodologia FIFO (First in first out).
Não temos de definir callbacks de complexidade extrema pelo facto de apenas podermos definir um e até podemos começar a usar esta funcionalidade de um modo inteligente, para por exemplo, executar determinado código caso algumas funções ajax tenham sido executadas com sucesso, isto de uma forma extremamente simples, utilizando a função $.then():

```
function doAjax() {
    return $.get('ajax.aspx');
}

function doMoreAjax() {
    return $.get('ajax2.aspx');
}

$.when( doAjax(), doMoreAjax() )
    .then(function(){
          console.log('Executado quando ambos os pedidos estão completos!');
    })
    .fail(function(){
          console.log('Executado quando um ou mais pedidos falharam!');
    });
```

Este código funciona porque o AJAX agora retorna uma promise() que é utilizada para monitorizar o pedido assíncrono, esta promise() é um objecto apenas de leitura que existe no resultado da tarefa. Os deferreds verificam a existência da função promise() para determinar se um objecto é observable ou não, que é o que lhe permite funcionar como deferred. A função when() aguarda pela execução das funções AJAX passadas por parâmetro e quando estas são executadas os métodos then() e fail() são executados, conforme o estado da tarefa. Importante referir novamente que os callbacks são executados pela ordem cujo são assignados a cada método.

Uma nota importante: os deferreds aceitam ou funções ou arrays de funções, que nos permite definir conjuntos de comportamentos na nossa aplicação e passá-los genericamente, ao invés de passarmos apenas uma função isolada.
Podemos verificar o estado de um deferred através das suas funções isRejected() e isResolved().
No caso do AJAX o que obtemos é um acesso a uma parte do deferred, visto que se tivessemos acesso completo poderíamos controlar quando os callbacks são executados através da função resolve() e poderíamos invocá-los antes dos pedidos realmente serem executados, o que iria quebrar a lógica, logo temos apenas acesso a uma parte do deferred, à promise(), que é apenas de leitura, como já foi referido.

Em termos de métodos, os que utilizámos até agora foram o then(), success() e fail(), também falámos do complete() no caso de AJAX, mas existem mais métodos que podemos utilizar, especialmente no caso de estarmos a lidar com AJAX. O método escolhido depende exclusivamente do estado ao qual queremos fazer bind.
Para todos os deferreds existem os seguintes métodos:
- then(doneCallbacks, failedCallbacks);
- done(doneCallbacks);
- fail(failCallbacks);

Os deferreds de AJAX têm 3 métodos adicionais que se podem especificar, 2 dos quais invocam um dos acima especificados. Estes métodos específicos existem exclusivamente para não quebrar a compatibilidade com os nomes dos callbacks para AJAX que existiam nas versões anteriores de jQuery:
- success(doneCallbacks); -> maps to done()
- error(failCallbacks); -> maps to fail()

Existe também o método complete() que é invocado após a função AJAX ser executada, retorne ou não erro. Ao contrário do success e do error o complete é um alias para o done, que é resolvido assim que o pedido AJAX termina, independentemente do seu resultado.
- complete(completeCallbacks);

Um exemplo de utilização de deferreds num bloco de código "típico":
Podemos ver aqui a utilização dos deferreds num bloco simples, e a sua explicação é muito simples:

```
function getData(){
   return $.get('/echo/html/');
}

function showDiv() {
    var dfd = $.Deferred();

    $('#foo').fadeIn( 1000, dfd.resolve );

    return dfd.promise();
}

$.when( getData(), showDiv() )
    .then(function(result) {
          console.log('A animação e o pedido
AJAX foram executados');
    });
```

Na função showDiv estamos a criar um objecto deferred novo, e retornamos a promise(). Este deferred como o código o mostro é resolvido assim que o fadeIn terminar, pois o dfd.resolve foi definido como callback deste fadeIn. O getData, retorna um objecto compatível com deferred (não exactamente igual, visto que é um AJAX e como já foi referido o AJAX não é um deferred "simples"), e como o objecto retornado pelo getData, tem o método promise, é tratado com deferred e o when() aguarda que ambos estejam no estado resolved, após estarem, executa o callback passado no método then() e escreve na consola.

Neste artigo podemos observar todo o potencial do AJAX, a sua evolução nesta nova versão 1.5 e também a grande nova funcionalidade que são os deferreds.
O jQuery está em constante evolução, esta é uma das novas features da versão 1.5, como foi demonstrado, tem um potencial enorme e uma abrangência e influência grandes, visto que até afectou áreas core da framework.
Stay tuned!


Link para o artigo: http://tinyurl.com/RPED28-08

## AUTOR

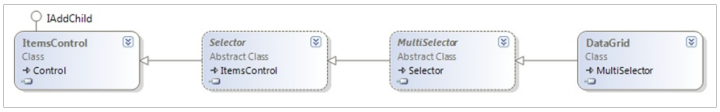
Escrito por **Ricardo Rodrigues**
É técnico Nível III em Informática/Gestão pela Fundação Escola Profissional de Setúbal, tendo ingressado após na FCT da Universidade Nova de Lisboa.
Posteriormente frequentou vários cursos da Microsoft em diversas áreas como Windows Forms,ASP.NET, Securing .NET Applications, WCF, WWF, Web Services e COM+ tendo obtido as certificações MCP .NET 2.0, MCAD .NET 1.1, MCSD .NET 1.1, MCPD Windows, Web e Distributed Applications e MCPD - Enterprise Applications Developer. (MCP Profile)
Contribui activamente em comunidades como StackOverflow e também possui um blog/twitter como temática relacionada: Blog / @ricmrodrigues

# Datagrid em Windows Presentation Foundation

Neste artigo pretendo apresentar a Datagrid em Windows Presentation Foundation (WPF) na .Net Framework 4.0. Vou começar por uma breve apresentação teórica e em seguida irei apresentar vários exemplos. De salientar que não terei em conta Design Patterns.

A DataGrid é um controlo que permite apresentar dados, representando cada linha um item de uma lista de objectos do mesmo tipo e as colunas representam as várias características do objecto. Ou seja, se na instância da datagrid apresento uma lista de empregados, cada linha representa um empregado e cada coluna representa uma propriedade do empregado.

A classe DataGrid está incluída no namespace System.Windows.Controls e é um selector que permite seleccionar mais do que um item ao mesmo tempo e tem por base a classe ItemsControl, que é um Controlo e que implementa a interface IAddChild. A seguinte imagem mostra-nos esta hierarquia de classes.




Vejamos agora algumas propriedades, métodos e eventos relevantes da DataGrid.

As propriedades mais relevantes são:

• Name: permite definir o nome.

• Foreground: permite definir a cor da letra.

• Background: permite definir a cor de fundo.

• AlternatingRowBackground; AlternationIndex: permitem definir a cor de fundo de cada linha, de forma alternada.

• Width; Height; MaxHeight; MinWidth; MaxWidth; MinHeight: permitem definir a largura, altura e seus valores mínimos e máximo. ActualHeight; ActualWidth permitem obter qual é o valor actual da altura e da largura.

• ColumnWidth; MaxColumnWidth; MinColumnWidth; ColumnHeaderHeight; permitem definir a largura, largura máxima, largura mínima e a altura do cabeçalho da coluna.

• RowHeight; MinRowHeight: permitem definir a altura e altura mínima da linha.

• GridLinesVisibility: permite definir a visibilidade das linhas que delimitam as linhas e colunas. Caso sejam visíveis as propriedades HorizontalGridLinesBrush e VerticalGridLinesBrush permitem definir aparência das linhas.

• SelectionMode; SelectionUnit permitem definir o modo de selecção dos itens, ou seja, se é possível seleccionar mais de que um item e se é permitido seleccionar linha(s) completa(s) ou célula(s).

• AutoGenerateColumns: permite gerar automáticas as colunas da datagrid. Para a geração automática é considerada todas as propriedades do tipo de objecto em causa, ou no caso de se estar a usar um dataTable a geração é baseada nas colunas da dataTable.

• Columns: permite obter a colecção de colunas.

• CellStyle: permite definir o estilo da célula.

• ColumnHeaderStyle: permite definir o estilo do cabeçalho da coluna.

• CanUserAddRows: permite que o utilizado adicione novos itens.

• CanUserDeleteRows: permite que o utilizado apague itens.

• CanUserReorderColumns permite que o utilizador organize as colunas.

• CanUserResizeColumns: permite que o utilizador redimensione as colunas.

• CanUserResizeRows: permite que o utilizador redimensione as linhas.

• CanUserSortColumns: permite que o utilizador ordene os itens ao fazer duplo clique no cabeçalho da coluna.

• CurrentColumn: permite obter a coluna actual.

• CurrentCell: permite obter a célula actual.

• CurrentItem: permite saber o item actual.

• SelectedCells: permite saber quais as células

que estão seleccionadas.

• SelectedIndex: permite saber qual o índex do item seleccionado.

• SelectedItem: permite obter o item que está seleccionado.

• SelectedItems: permitem saber os itens que estão seleccionados.

• HasItems: permite saber se a datagrid contém itens.

• Items: permite obter a lista de itens.

• ItemsSource: permite obter a lista de itens.

• ItemTemplate; ItemTemplateSelector: permitem definir o template aplicar aos itens.

Nota: Existem diferenças entre a propriedade Items e a propriedade ItemsSource e apenas se pode usar uma delas. Quando se usa a propriedade ItemsSource, não é possível adicionar ou remover itens através desta propriedade ou através da propriedade Items. No entanto, se atribuirmos uma ObservableCollection ao ItemsSource, essa questão é ultrapassada.

Os métodos mais relevantes são:

• BeginEdit: permite invocar o comando BeginEditCommand, que será responsável por alterar para modo de edição a linha ou célula seleccionada.

• CommitEdit: permite invocar o comando CommitEditCommand para célula ou linha que está em modo de edição.

• CancelEdit:permite invocar o comando CancelEditCommand para célula ou linha que está em modo de edição.

Os eventos mais relevantes são:

• BeginningEdit: ocorre antes da célula ou linha entrar em modo de edição.

• PreparingCellForEdit: ocorre quando a célula entra em modo de edição.

• CellEditEnding: ocorre antes de haver um "commit" da célula ou antes de ser cancelada a edição.

• RowEditEnding: ocorre antes de haver um "commit" da linha ou antes de ser cancelada a edição.

• SelectionChanged: ocorre quando a selecção é alterada.

• CurrentCellChanged: ocorre quando o valor da propriedade CurrentCell é alterado.

• SelectedCellsChanged: ocorre quando o valor da propriedade SelectedCells é alterado.

Nota: O leitor que pretenda efectuar um estudo mais pormenorizado sobre as propriedades, métodos e eventos da DataGrid, deve consultar a página DataGrid Class no MSDN: http://bit.ly/ewIdaP

Antes de passarmos aos exemplos práticos, vejamos os vários tipos de colunas da DataGrid.

Em WPF, podemos definir os seguintes tipos de colunas:

• DataGridTextColumn – permite apresentar e editar texto;

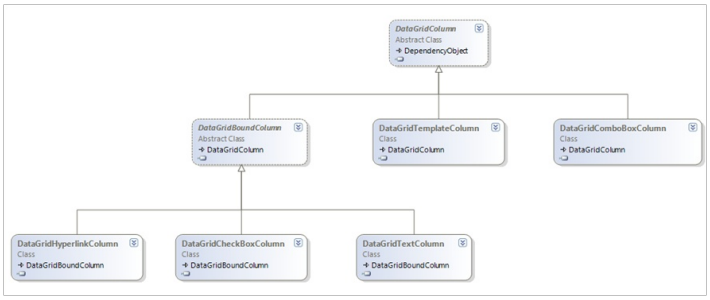
• DataGridCheckBoxColumn – permite atribuir um valor lógico (verdadeiro / falso)

• DataGridComboBoxColumn – permite seleccionar uma opção de um conjunto de opções;

• DataGridHyperlinkColumn – permite adicionar uma hiperligação;

• DataGridTemplateColumn – permite definir o template da coluna, isto é, podemos customizar a coluna.

O seguinte diagrama, permite-nos perceber melhor qual a classe de base de cada tipo de coluna.

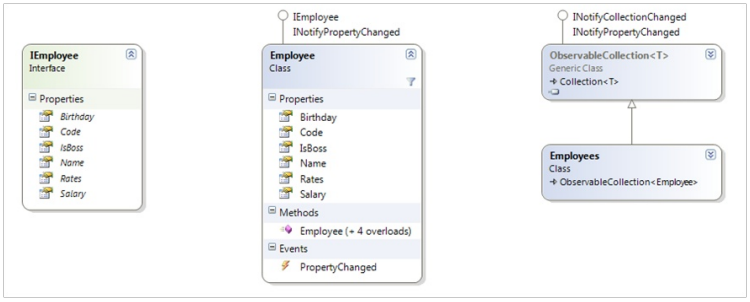



Passemos agora aos exemplos práticos.

Nota: De apoio a este artigo criei uma demo, o leitor poderá obter em: http://bit.ly/efSP1W

Suponhamos que temos um objecto empregado (Employee) e um conjunto de empregados (Employees). O empregado, tem como principais características: código,

nome, se é o patrão, salário, taxas e aniversário.




Notas:

• A classe Employee implementa a interface INotifyPropertyChanged para que sejam feitas notificações quando alguma propriedade é alterada. Isto é em parte responsável pela actualização da informação na Datagrid.

• A classe Employees é uma ObservableCollection de Employee, porque pretendo que sejam feitas notificações quando um ou mais empregrados são adicionados/removidos da lista de empregados.

Caso tivesse optado por List<Employee>, Collection<Employee> ou IEnumerable<Employee> essas notificações não aconteceriam. Na imagem anterior é possível analisar o que é uma ObservableCollection, sendo a implementação das interfaces INotifyCollectionChanged e INotifyPropertyChanged reponsável por informar sobre as notificações/actualizações.

1º Exemplo:

• Objectivo: Apresentar numa janela uma Datagrid com uma lista de empregados, as colunas da Datagrid devem ser geradas automaticamente:

• XAML

```
<DataGrid  Name="dataGridEmployees"
        Grid.Column="1" Grid.ColumnSpan="3"
        Grid.Row="2"
        AutoGenerateColumns="True"/>
```

• Code Behind:

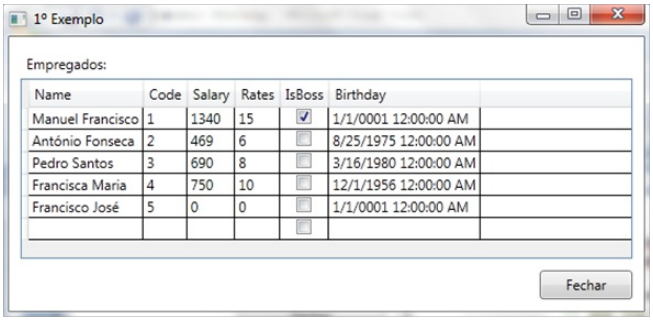
Usando a propriedade ItemsSource para atribuir a lista de empregados.

```
private void WindowLoaded(object sender,
RoutedEventArgs e){
        var employees=new Employees();
        // add the employee list
        //using ItemsSource
        dataGridEmployees.ItemsSource =
employees;
}
```

Usando a propriedade Items para atribuir a lista de empregados.

```
Usando a propriedade Items para atribuir a
lista de empregados.
private void WindowLoaded(object sender,
RoutedEventArgs e){
        var employees=new Employees();
        // add the employee list
        //using Items
        foreach (var employee in employees)

dataGridEmployees.Items.Add(employee);
}
```

• Resultado:




2º Exemplo:

• Objectivo: Apresentar numa janela uma Datagrid com uma lista de empregados, as colunas da Datagrid não devem ser geradas automaticamente e apenas se pretende visualizar as colunas relativas ao nome, se é patrão, salário e taxas.

## Datagrid em Windows Presentation Foundation

• XAML:

```
<DataGrid  Name="dataGridEmployees"
        Grid.Column="1" Grid.ColumnSpan="3"
        Grid.Row="2"
        AutoGenerateColumns="False">
        <DataGrid.Columns>
                <DataGridTextColumn
Header="Nome" MinWidth="200"
Binding="{Binding Name}" />
                <DataGridTextColumn
Header="SalÃ¡rio" Width="100" MaxWidth="150"
Binding="{Binding Salary}"/>
                <DataGridTextColumn
Header="Taxa" Width="50" Binding="{Binding
Rates}"/>
                <DataGridCheckBoxColumn
Header="É patrão" Binding="{Binding
IsBoss}"/>
   </DataGrid.Columns>
</DataGrid>
```

Notas:

• Através do objecto Binding da coluna é definido qual a propriedade do objecto que se vai apresentar.

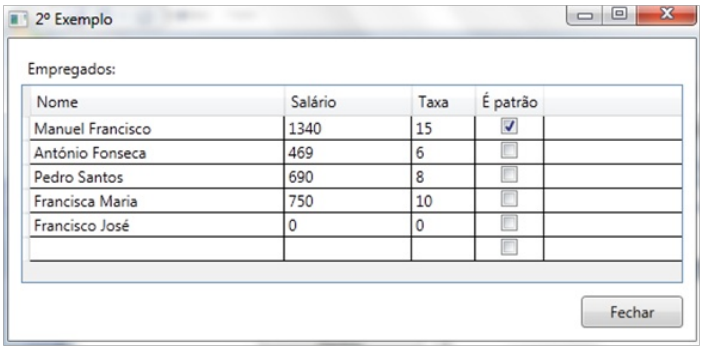
• Foram definidas as medidas para a largura, largura mínima e largura máxima das várias colunas.

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e) {
        var employees = new Employees();
        // add the employee list
        dataGridEmployees.ItemsSource =
employees;
}
```

• Resultado:

(A imagem com o resultado pode ser visto na coluna do lado direito em cima de todo)




3º Exemplo:

• Objectivo: Formatar a apresentação do salário e das taxas, ou seja, um salário com valor 0 (zero) deve apresentar o valor “Não está atribuído” e os salários que estejam definidos devem apresentar o símbolo da moeda € (euro). As Taxas devem apresentar o valor da taxa com o símbolo de percentagem.

Neste exemplo, temos necessidade de criar duas classes, cada classe representa o conversor que será usado no objecto de Binding da coluna. O conversor será responsável por transformar o valor original no valor que é pretendido apresentar. Ambas as classes implementam a interface IValueConverter.

Portanto, o conversor para o salário será definido da seguinte forma:

```
[ValueConversion(typeof(double),
typeof(string))]
public class
SalaryValueConverter:IValueConverter{
        private const string
NotDefinedValue="Não está definido";
        private const string EuroSymbol = "€";

        public object Convert(object value,
Type targetType, object parameter,
System.Globalization.CultureInfo culture) {
                if (value == null ||
(double.Parse(value.ToString()) == 0))
                        return NotDefinedValue;

                return string.Format("{0} {1}",
```

```
value, EuroSymbol);
        }

        public object ConvertBack(object
value, Type targetType, object parameter,
System.Globalization.CultureInfo culture) {
                if (value is string &&
value.ToString().Equals(NotDefinedValue))
                        return 0;

                return value is string &&
value.ToString().Contains(EuroSymbol)
                                        ?
double.Parse(value.ToString().Replace(EuroSym
bol, string.Empty))
                                        : value;
        }
}
```

O conversor para a taxa será definido da seguinte forma:

```
[ValueConversion(typeof(int),typeof(string))]
public class
RatesValueConverter:IValueConverter{
    private const string PercentageSymbol =
"%";
        public object Convert(object value,
Type targetType, object parameter,
System.Globalization.CultureInfo culture){
                if (value is int &&
int.Parse(value.ToString()) == 0)
                return string.Empty;
                return string.Format("{0} {1}",
value, PercentageSymbol);
    }
        public object ConvertBack(object
value, Type targetType, object parameter,
System.Globalization.CultureInfo culture){
         throw new NotImplementedException();
    }
}
```

• XAML:

É necessário adicionar o namespace dos resources

```
xmlns:Converters="clr-
namespace:WPFDatagridImplementation.Converter
s"
```

Nos resources da janela, inicializamos os dois conversores

```
<Window.Resources>
    <Converters:RatesValueConverter
x:Key="ratesValueConverter"/>
    <Converters:SalaryValueConverter
x:Key="salaryValueConverter"/>
</Window.Resources>
```

É necessário alterar o Binding das colunas, é neste objecto que definimos o conversor aplicar.

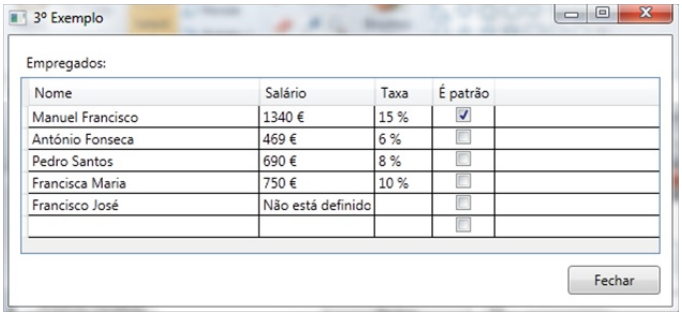
```
<DataGrid  Name="dataGridEmployees"
        Grid.Column="1" Grid.ColumnSpan="3"
        Grid.Row="2"
        AutoGenerateColumns="False">
        <DataGrid.Columns>
                <DataGridTextColumn
Header="Nome" MinWidth="200"
Binding="{Binding Name}" />
                <DataGridTextColumn
Header="Salário" Width="100" MaxWidth="150"
Binding="{Binding
Path=Salary,Converter={StaticResource
salaryValueConverter}}"/>
                <DataGridTextColumn
Header="Taxa" Width="50" Binding="{Binding
Path=Rates,Converter={StaticResource
ratesValueConverter}}"/>
                <DataGridCheckBoxColumn
Header="É patrão" Binding="{Binding
IsBoss}"/>
        </DataGrid.Columns>
</DataGrid>
```

## Datagrid em Windows Presentation Foundation

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e) {
        var employees = new Employees();
        // add the employee list
        dataGridEmployees.ItemsSource =
employees;
}
```

• Resultado:




4º Exemplo:

• Objectivo: Apresentar uma coluna extra, que será o resultado do produto do salário com a taxa associada.

É necessário criar um conversor, que receba o salário e a taxa, e devolva o resultado final, formado. Neste caso, como é preciso receber dois valores a classe terá que implementar a interface IMultiValueConverter.

```
public class
FinalySalaryConverter:IMultiValueConverter {
        public object Convert(object[] values,
Type targetType, object parameter,
System.Globalization.CultureInfo culture) {
                double salary = 0.0;
                int rates = 0;

                if (values[0] is double)
                        salary =
double.Parse(values[0].ToString());

                if (values[1] is int)
                        rates =
int.Parse(values[1].ToString());
                if (salary == 0 && rates == 0)
                        return string.Empty;

                return string.Format("{0} €",
salary - salary * rates / 100);
        }

        public object[] ConvertBack(object
value, Type[] targetTypes, object parameter,
System.Globalization.CultureInfo culture) {
                throw new
NotImplementedException();
        }
}
```

Nota: O objecto values que está definido no argumento do método Convert, recebe os valores de acordo com a ordem que for definido no multibinding que será definido na coluna.

• XAML:

Adicionamos o conversor nos resources da janela.

```
<Window.Resources>
        <Converters:RatesValueConverter
x:Key="ratesValueConverter"/>
        <Converters:SalaryValueConverter
x:Key="salaryValueConverter"/>
        <Converters:FinalySalaryConverter
x:Key="finalySalaryConverter"/>
</Window.Resources>
```

Adicionamos a nova coluna com o nome “Salário Final”, repare-se que usou-se um Multibinding, para permitir enviar o valor do salário e o valor da taxa.

```
<DataGrid.Columns>
        <DataGridTextColumn Header="Nome"
MinWidth="200" Binding="{Binding Name}" />
        <DataGridTextColumn Header="Salário"
Width="100" MaxWidth="150" Binding="{Binding
```

```
Path=Salary,Converter={StaticResource
salaryValueConverter}}"/>
        <DataGridTextColumn Header="Taxa"
Width="50" Binding="{Binding
Path=Rates,Converter={StaticResource
ratesValueConverter}}"/>
        <DataGridTextColumn Header="Salário
Final">
          <DataGridTextColumn.Binding>
           <MultiBinding
Converter="{StaticResource
finalySalaryConverter}">
                        <Binding
Path="Salary"/>
                        <Binding Path="Rates"
/>
               </MultiBinding>
          </DataGridTextColumn.Binding>

        </DataGridTextColumn>
        <DataGridCheckBoxColumn Header="É
patrão" Binding="{Binding IsBoss}"/>
</DataGrid.Columns>
</DataGrid>
```

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e) {
        var employees = new Employees();
        // add the employee list
        dataGridEmployees.ItemsSource =
employees;
}
```

• Resultado:


5ºExemplo:

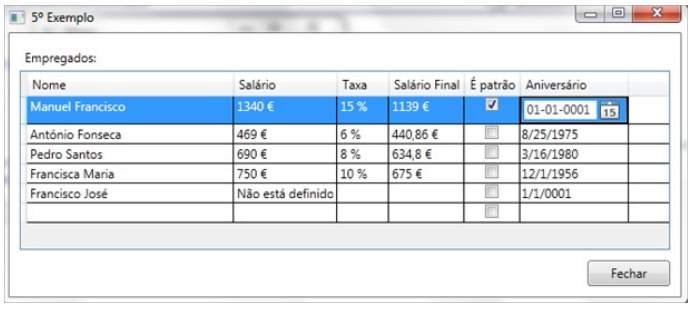
• Objectivo: Apresentar a coluna de Aniversário, recorrendo ao controlo DataPicker.

• XAML:

```
<DataGridTemplateColumn Header="AniversÃ¡rio"
MinWidth="100">

<DataGridTemplateColumn.CellEditingTemplate>
                <DataTemplate>
                        <DatePicker
SelectedDate="{Binding Birthday}"
SelectedDateFormat="Short" />
                </DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>
        <DataGridTemplateColumn.CellTemplate>
                <DataTemplate>
                        <TextBlock
Text="{Binding Birthday, StringFormat=d}" />
                </DataTemplate>
        </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

Nota: Apenas apresento o XAML da coluna criada. Foram definidos dois templates: um para o caso em que se está em modo de edição, em que usou aplicar um DatePicker. O outro template apenas apresenta a data.

• Resultado:



## Datagrid em Windows Presentation Foundation

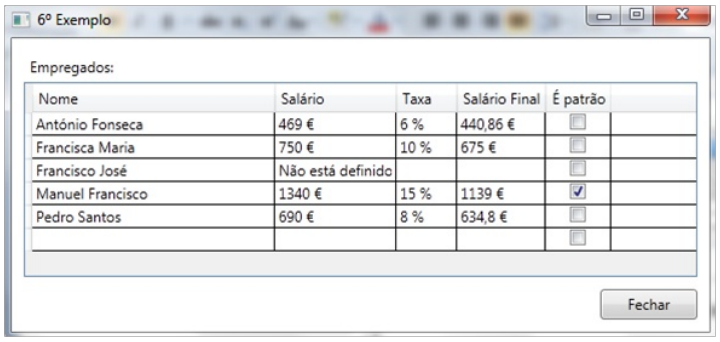
6º Exemplo:

• Objectivo: Apresentar os empregados ordenados por Nome.

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e) {
        var employees = new Employees();
        dataGridEmployees.ItemsSource =
employees;

        ICollectionView view =
CollectionViewSource.GetDefaultView(dataGridE
mployees.ItemsSource);

        view.SortDescriptions.Add(new
SortDescription("Name",
ListSortDirection.Ascending));
}
```

• Resultado:



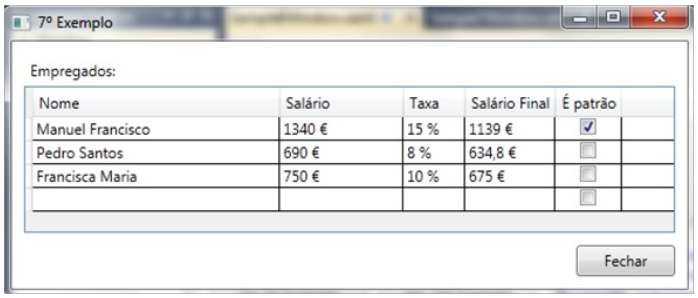

7º Exemplo:

• Objectivo: Filtrar os empregados com salário superior a 500.00€.

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e) {
        var employees = new Employees();
        dataGridEmployees.ItemsSource =
employees;

        ICollectionView view =
CollectionViewSource.GetDefaultView(dataGridE
mployees.ItemsSource);
        view.Filter = new
Predicate<object>(EmployeesWithSalaryGreaterT
han500);
}

private bool
EmployeesWithSalaryGreaterThan500(object
param) {
        var employee = (Employee)param;

        if (employee.Salary > 500)
                return true;

        return false;
}
```

• Resultado:




8º Exemplo:

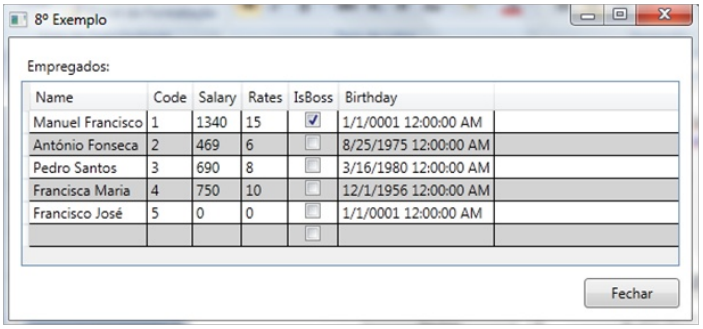
• Objectivo: As linhas da datagrid devem apresentar uma cor alternada.

• XAML:

```
<DataGrid  Name="dataGridEmployees"

        Grid.Column="1" Grid.ColumnSpan="3"
        Grid.Row="2"
        AutoGenerateColumns="True"
```

```
        AlternatingRowBackground="LightGray"
AlternationCount="2"/>


private void WindowLoaded(object sender,
RoutedEventArgs e) {
        var employees = new Employees();
        dataGridEmployees.ItemsSource =
employees;
}
```

• Resultado:




9º Exemplo:

• Objectivo: Alterar a altura do cabeçalho das colunas, alterar o tipo de fonte da letra e tamanho e cor. Alterar a cor das linhas horizontais e verticais da datagrid.

• XAML:

Nos resources da Window, definimos o estilo que iremos aplicar às células.

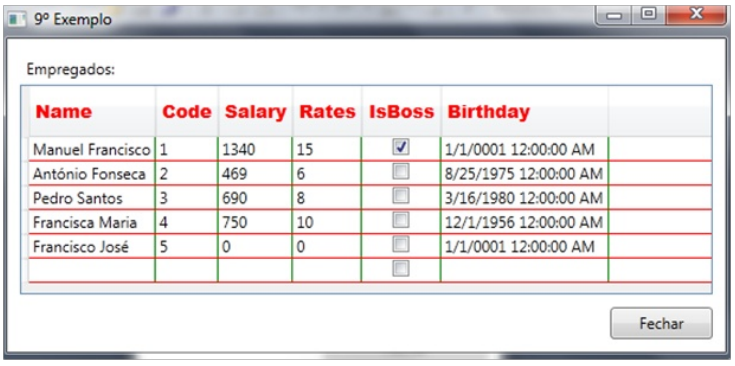
```
<Window.Resources>
        <Style x:Key="columnStyle"
TargetType="DataGridColumnHeader">
                <Setter Property="FontFamily"
Value="Arial Black" />
                <Setter Property="FontSize"
Value="14"/>
                <Setter Property="Foreground"
Value="Red"/>
        </Style>
</Window.Resources>
```

```
<DataGrid Name="dataGridEmployees"

        Grid.Column="1" Grid.ColumnSpan="3"
        Grid.Row="2"
        AutoGenerateColumns="True"
        ColumnHeaderStyle="{DynamicResource
columnStyle}"
        ColumnHeaderHeight="40"
        GridLinesVisibility="All"
        HorizontalGridLinesBrush="Red"
        VerticalGridLinesBrush="Green"/>
```

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e){
        var employees = new Employees();
        dataGridEmployees.ItemsSource =
employees;
}
```

• Resultado:




10º Exemplo:

• Objectivo: Ao seleccionar uma célula e mostrar uma mensagem de aviso de que foi detectada uma alteração, apresentando o nome do empregado seleccionado e o nome da coluna seleccionada.

## Datagrid em Windows Presentation Foundation

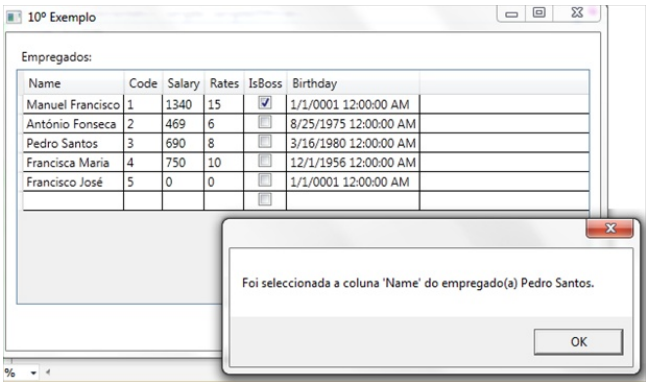
• XAML:

```
<DataGrid  Name="dataGridEmployees"

        Grid.Column="1" Grid.ColumnSpan="3"
        Grid.Row="2"
        AutoGenerateColumns="True"
        SelectionMode="Single"
SelectionUnit="Cell"

CurrentCellChanged="DataGridEmployeesCurrentC
ellChanged"/>
```

• Code Behind:

```
private void WindowLoaded(object sender,
RoutedEventArgs e)
        {
            var employees = new Employees();
            dataGridEmployees.ItemsSource =
employees;
        }
private void
DataGridEmployeesCurrentCellChanged(object
sender, System.EventArgs e) {
        Employee selectedEmployee =
(Employee)dataGridEmployees.CurrentItem;

        string message = String.Format("Foi
seleccionada a coluna '{0}' do empregado(a)
{1}.",
dataGridEmployees.CurrentCell.Column.Header,
selectedEmployee.Name);
        MessageBox.Show(message);

}
```

• Resultado:




Demo do artigo: http://bit.ly/efSP1W

Contém:

- Diagrama de Classes da hierarquia das colunas
- Diagrama de classes da hierarquia da Datagrid
- Diagrama de classes do modelo usado no demo
- Interfaces
- Classes de dados
- Conversores
- Janela principal com menu para os vários exemplos
- Janelas dos vários exemplos

Referências:


- MSDN: DataGrid Class - .Net Framework 4.0: http://bit.ly/ewIdaP
- WindowsClient.Net: WPF Toolkit: DataGrid Feature Walkthrough: http://bit.ly/hksWsL


Link para o artigo: http://tinyurl.com/RPED28-05

## AUTOR


Escrito por **Sara Silva**

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

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

# Planos de Execução em ORACLE

## Introdução

Nas últimas duas edições, abordámos o tema da optimização de SQL com o recurso a técnicas de melhoria do código SQL. Recorremos, nomeadamente, à utilização de bind variables e à correcta implementação de índices. Nesta edição, vamos apresentar uma das ferramentas de diagnóstico que permitem prever e verificar a optimização: os planos de execução.

## O que é

Antes de efectuar um SELECT na base de dados, o optimizador traça um plano de execução, de modo a avaliar que dados recolher, onde e como os deve obter. O plano de execução é influenciado por múltiplos factores e varia ao longo do tempo, isto é, varia consoante o estado presente da base de dados.

Os principais factores que definem o plano de execução são a dimensão das tabelas acedidas, a diversidade de dados das tabelas (selectividade) e a existência ou não de índices. Mas, como é que o optimizador obtém, rapidamente, esta informação? O Oracle produz estatísticas da base de dados. As estatísticas registam os principais factores que definem o plano de execução. Deste modo, o optimizador, em vez de percorrer todos os dados afectados por um SELECT, sabe à partida o que vai encontrar. No entanto, para que as estatísticas produzam bons resultados nas queries, é fundamental que estejam actualizadas. Por exemplo, se a estatística nos diz que uma tabela tem dez registos mas, entretanto, carregámos um milhão de novos registos, a estatística produzirá valores errados e induzirá o optimizador por um caminho que não é o óptimo.

Podemos fazer uma analogia com uma deslocação de automóvel desde casa até ao local de trabalho. Se soubermos, antecipadamente, o estado do trânsito, sabemos qual a via que, mesmo não parecendo ser a mais óbvia, deverá ser a mais rápida. Se tivermos ouvido o estado do trânsito há uma hora atrás, a situação deverá já estar totalmente diferente, quando sairmos de casa, e a rota mais rápida poderá agora estar entupida. Se, por último, desconhecermos totalmente o estado do trânsito, seguiremos pelo caminho que consideramos mais directo mas que, eventualmente, poderá não ser o mais rápido naquele momento. O Oracle procede do mesmo modo, pelo que, manter as estatísticas actualizadas é uma mais-valia para que o optimizador possa tomar a melhor decisão e, no caso do plano de execução, apresentar o plano mais idêntico à realidade.

## Criar a tabela PLAN_TABLE

Os planos de execução são registados numa tabela, usualmente definida como PLAN_TABLE, sob a forma de linhas com relações hierárquicas entre si. Por isso, é possível registar vários planos de execução na PLAN_TABLE, de modo a compararmos diferentes queries.

A PLAN_TABLE poderá não estar disponível no ambiente Oracle. O Oracle fornece um script que permite criar essa tabela facilmente. O script pode ser encontrado na directoria %ORACLE_HOME%/rdbms/admin com o nome UTLXPLAN.SQL. Consoante a versão de Oracle que estamos a utilizar, o script pode conter algumas variações a nível das colunas da tabela. Este script não é mais do que um CREATE TABLE, mas não será aqui exposto, devido à dimensão do mesmo e às variantes que existem para diferentes versões Oracle.

Tendo a garantia de que a tabela existe, devemos sempre limpar o seu conteúdo, antes de iniciarmos o “estudo” das queries, recorrendo à instrução TRUNCATE TABLE PLAN_TABLE.

## Determinar o plano de execução

Estamos, agora, em condições de começar a obter planos de execução das nossas queries. Necessitamos, primeiramente, de registar o plano de execução na PLAN_TABLE, recorrendo à instrução EXPLAIN PLAN SET STATEMENT_ID = 'XPTO' FOR, seguida da query que queremos analisar. Por exemplo:

```
EXPLAIN PLAN SET STATEMENT_ID = 'XPTO' FOR
SELECT C.COD_ASSOC FROM ASSOCIADOS A
INNER JOIN CARTOES C ON C.COD_ASSOC =
A.CODIGO
WHERE A.CODIGO BETWEEN 21000 AND 21500;
```

O STATEMENT_ID atribuído é o identificar do plano do tipo varchar2. O plano está criado e podemos continuar a determinar os planos de execução de outras queries ou, por exemplo, verificar os planos antes e após o cálculo das estatísticas. Os planos ficam registados na tabela e só serão eliminados manualmente, pelo que se poderão guardar os registos para consulta futura.

## Analisar o plano de execução

Naturalmente que, se a PLAN_TABLE é uma tabela, a visualização do plano de execução é feita com um SELECT à mesma tabela. Alguns editores SQL de Oracle já possuem essa funcionalidade à distância de um clique. O que fazem é executar um dos muitos SELECT's possíveis à tabela. O Oracle também já facilita a consulta dos planos de execução através da seguinte instrução, adaptada aqui ao nosso exemplo, com o respectivo statement_id:

```
SELECT * FROM
TABLE(dbms_xplan.display('plan_table','xpto',
'all'));
```

O resultado desta query poderia ser o seguinte:

```
-----------------------------------------------------------------------------------
| Id  | Operation            | Name               | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                    |   394 |  3940 |   135  (15)|
|   1 |  NESTED LOOPS        |                    |   394 |  3940 |   135  (15)|
|*  2 |   TABLE ACCESS FULL  | CARTOES            |   514 |  2570 |   132  (13)|
|*  3 |   INDEX RANGE SCAN   | ASSOCIADOS_PK      |     1 |     5 |            |
-----------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("C"."COD_ASSOC"<=21500 AND "C"."COD_ASSOC">=21000)
   3 - access("C"."COD_ASSOC"="A"."CODIGO")
       filter("A"."CODIGO"<=21500 AND "A"."CODIGO">=21000)
```

A forma correcta de ler o mapa do plano é começar pela instrução mais à direita (maior nível). Quando duas instruções estão ao mesmo nível, começa-se pela que tem um ID menor. Neste caso, vemos que a primeira instrução a ser executada será a n.º 2.

Vejamos, primeiro, as colunas da PLAN_TABLE apresentadas nesta query:

• Id: identificador da linha da instrução do presente plano. Atenção, não é o ID da linha na PLAN_TABLE, mas sim o ID da amostra de dados retirada dessa mesma tabela. Lembremo-nos que a PLAN_TABLE pode conter vários planos de execução.

• Operation: O tipo de instrução a ser executada.

• Name: Tabela ou índice a que se refere a Operation.

• Rows: Número de linhas afectadas ou acedidas.

• Bytes: Total de bytes que serão movimentados para ler os dados da instrução.

• Cost: O custo de CPU para a instrução. Este campo não tem qualquer unidade, pelo que o mesmo deverá ser utilizado como meio de comparação. Por exemplo, comparando o Cost CPU de uma query leve com uma query mais pesada. Este valor é parametrizado num ficheiro de configuração do Oracle e a sua compreensão mais profunda seria alvo de um tema de administração de bases de dados.

Note-se que os campos apresentados pela função do Oracle não correspondem aos campos da PLAN_TABLE em bruto. A função não faz mais do que criar uma VIEW com a selecção de dados da tabela do plano de execução.
Analisemos, agora, a informação apresentada para este plano. Os ID's 2 e 3 estão sob a operação de junção NESTED LOOPS. Este é um dos tipos de junção de tabelas

que consiste em:

(2) Escolher a tabela sem índice na condição, neste caso a tabela CARTOES que não possui um índice no campo de junção COD_ASSOC. É feito um FULL TABLE SCAN filtrando as linhas com a condição:

```
filter("C"."COD_ASSOC"<=21500 AND
"C"."COD_ASSOC">=21000)
```

Repare-se no * antes do 2, referente à nota em baixo (Predicate Information), que indica o filtro no SCAN com os números de código do SELECT:

```
WHERE A.CODIGO BETWEEN 21000 AND 21500;
```

(3) Para cada uma das 514 linhas da tabela anterior, já filtrada, é feito um acesso ao índice da tabela ASSOCIADOS, ASSOCIADOS_PK, obtendo apenas os dados com:

```
filter("A"."CODIGO"<=21500 AND
"A"."CODIGO">=21000)
```

Tendo em conta que cada linhas das duas tabelas são cruzadas obedecendo ao match:

```
access("C"."COD_ASSOC"="A"."CODIGO")
```

(1) O Oracle pode agora juntar as tabelas, fazendo o INNER JOIN através do algoritmo de NESTED LOOPS. Obtêm-se 394 linhas, no final, e terão sido acedidos 3940 bytes.

## Conclusão

O plano de execução é uma ferramenta bastante poderosa para fazer uma avaliação prévia do impacto da nossa query na base de dados. Seja para avaliar o impacto de uma query num ambiente de produção em alturas críticas ou simplesmente para optimizar e procurar queries com menor consumo de CPU ou IO, o plano de execução é um bom começo.

Conhecendo, detalhadamente, o tipo de operações que o Oracle executa, o plano de execução pode ser, na grande

**Lista de algumas expressões utilizadas pelo optimizador nos planos de execução**

| | | | |
|---|---|---|---|
| Tabelas | TABLE ACCESS | FULL | Usualmente, a leitura mais lenta à tabela. Percorre todas as linhas da tabela. |
| | | HASH | Acesso à tabela através de uma *hash key* (uma espécie de índice construído no momento da pesquisa em situações de JOIN). |
| | | BY ROWID | Acesso a uma única linha da tabela através do seu respectivo ROWID. É o método mais rápido e requer um índice (a não ser que a pesquisa seja feita directamente com um SELECT (...) WHERE ROWID...). |
| Índices | INDEX | UNIQUE SCAN | Consulta a um índice que retorna o ROWID de uma única linha. |
| | | RANGE SCAN | Consulta a um índice que retorna várias linhas, por exemplo através da pesquisa com operadores WHERE >, <, >=, <=. |
| Junção | MERGE JOIN | | Junção através do método MERGE JOIN. |
| | | OUTER | Junção externa através do método MERGE JOIN. |
| | NESTED LOOPS | | Junção através do método NESTED LOOPS. |
| | | OUTER | Junção externa através do método NESTED LOOPS. |
| Agregação | COUNT | | Contagem das linhas do resultado para satisfazer uma função COUNT() |
| | SORT | JOIN | Ordenação das linhas para a seguir efectuar uma junção. |
| | | UNIQUE | Ordenação para a seguir remover as linhas duplicadas. |
| | | GROUP BY | |
| | | ORDER BY | |

maioria dos casos, mais do que suficiente para prever o resultado da query na BD. No entanto, nem sempre o plano de execução é eficaz. Há casos específicos que necessitam de especial atenção e de uma avaliação pós-processamento através de outra ferramenta, o SQL Trace. O SQL Trace vai registar as operações realizadas pelo Oracle, permitindo-nos analisar detalhadamente o que ocorreu no SELECT. Enquanto o plano de execução permite fazer uma avaliação a priori, o SQL Trace faz uma avaliação a posteriori.

Convém, ainda, frisar que, para que o plano de execução seja o mais fiel possível, as estatística deverão estar actualizadas, o que nem sempre é possível ou viável se a dimensão das bases de dados for muito grande.

**Referências:**

Estrutura da tabela PLAN_TABLE e lista de operações produzidas pelo plano de execução:
http://download.oracle.com/docs/cd/A58617_01/server.804/a58246/explan.htm#891

Outras formas de representar o plano de execução:
http://download.oracle.com/docs/cd/A58617_01/server.804/a58246/explan.htm#1088

Link para o artigo: http://tinyurl.com/RPED28-07

## AUTOR

Escrito por **Ricardo Trindade**
É actualmente o responsável pela área informática dos Serviços Sociais da Câmara Municipal de Lisboa onde efectua, desde administração de sistemas, até programação em .NET. Está também ligado ao projecto N-Ideias na área da consultoria, webdesign e software, trabalhando essencialmente com BD's Oracle, PostgreSQL e SQL Server.

# Smarty PHP Template Engine

A maioria das pessoas que aprendem PHP começam, normalmente, a construir as suas aplicações Web de forma bastante rudimentar. Isto é normal, pois ao longo do tempo com que trabalham com PHP, os programadores vão-se habituando, aprendendo novas técnicas e desenvolvendo o seu próprio estilo de programar.

Este estilo desenvolvido reflecte-se maioritariamente na estrutura da aplicação desenvolvida. Isto porque cada aplicação tem as suas funcionalidades, e estas reflectem-se na necessidade de uma estrutura sólida capaz de suportar aplicações mais robustas, extensíveis e seguras.

No entanto, há estruturas padrões e versáteis que são dadas como ideais para a construção de aplicações, e criam-se então frameworks capazes de tornarem estas estruturas disponíveis para qualquer programador construir qualquer aplicação sem se ter de preocupar muito com a estrutura base da aplicação.

Em PHP usam-se muitas aplicações que seguem a lógica do padrão de arquitectura de software MVC, que pretende separar a lógica do padrão da lógica de apresentação. Models e Views, tendo pelo meio os Controllers. Mas não é necessariamente obrigatório usar frameworks MVC nem sequer aplicar o padrão MVC para criar aplicações que separem a lógica da aplicação da lógica da apresentação.

Cada programador pode criar a sua aplicação de modo a separar estas duas camadas, mas isso nem sempre é um trabalho fácil quando bem feito, e quando mal feito pode resultar no aparecimento de graves falhas na aplicação e de grandes quebras de performance. Assim sendo, porque não usar uma framework poderosa, robusta e simples de usar que já exista e nos simplifique o trabalho? É aqui que o Smarty entra.


## Introdução ao Smarty

Neste pequeno artigo vamos usar a versão 3.0.7 do Smarty que pode ser descarregada aqui[1]. Vamos também usar a versão 5.3.0 do PHP.

O Smarty permite-nos chamar determinados ficheiros e imprimi-los no ecrã, substituindo certos valores pelos obtidos e/ou gerados pela nossa aplicação, mostrando ao utilizador final, dados dinâmicos.

A diferença entre o método de templates em PHP ou Samrty é a mais fácil adaptação do visual da aplicação, sem que o designer tenha de saber a linguagem de programação (PHP) e tornando o template mais fácil de entender. Este método torna também pequenos pedaços de texto (possivelmente HTML) sejam reutilizados mais facilmente, evitando partes de código duplicadas na aplicação.

O Smarty não necessita de nenhuma configuração do sistema, tornando-o assim extremamente simples de instalar em qualquer Sistema Operativo, dado que os passos de instalação baseiam-se apenas na criação e edição de pastas e ficheiros.

## Instalar o Smarty

Depois de termos feito o download da última versão do Smarty, descomprimimo-la e copiamos os ficheiros para uma pasta no nosso projecto (na raiz do projecto ou possivelmente numa pasta chamada smarty). Neste artigo usaremos uma pasta dentro do nosso projecto chamada smarty. Portanto, apenas temos de copiar os conteúdos da pasta libs para dentro da pasta smarty. Depois, no decorrer do desenvolvimento do nosso projecto apenas vamos ter de especificar a localização de certas pastas necessárias ao funcionamento do Smarty, tais como a pasta de cache e dos templates.

## Primeiro Projecto em Smarty

Portanto, agora que temos os ficheiros que necessitamos, vamos começar a trabalhar com a biblioteca. Tudo gira à volta de uma classe, chamada Smarty, mas para isso necessitamos de incluir o ficheiro.

```
require('C:\caminhoAbsolutoParaOProjecto\smarty\Smarty.class.php');
```

Depois necessitamos de instanciar a classe para uma variável, algo simples.

```
$smarty = new Smarty();
```

Agora só temos de definir alguns directórios, necessários para o funcionamento da biblioteca.

Primeiro, vamos criar quatro pastas dentro da pasta smarty: cache, configs, templates, templates_c. Depois adicionamos o seguinte código ao ficheiro.

```
//Define as pastas. O seu caminho deve ser absoluto
$smarty->setTemplateDir('C:\caminhoAbsolutoParaOProjecto\smarty\templates');
$smarty->setCompileDir('C:\caminhoAbsolutoParaOProjecto\smarty\templates_c');
$smarty->setCacheDir('C:\caminhoAbsolutoParaOProjecto\smarty\cache');
$smarty->setConfigDir('C:\caminhoAbsolutoParaOProjecto\smarty\configs');
```

Feito isto, podemos começar a usar o Smarty à vontade.
Apesar de o Smarty ser muito bom, ele não adivinha os valores que a página vai ter, pelo que temos de ser nós a dar-lhos. Isto é bastante simples, apenas temos de chamar uma função e especificar o nome da variável que queremos substituir no template e o seu respectivo valor. Adicionemos então o seguinte código.

```
$smarty->assign("value", "Artigo de introdução ao Smarty");
```

Isto vai fazer com que as variáveis definidas no template com o nome value sejam substituídas pela respectiva string. Mas também é possível passar arrays. Para isso usamos a mesma função, a diferença estará no template.

```
$array = new Array( 1 => "Introdução ao Smarty", 2 => "Instalar o Smarty", 3 => "Primeiro Projecto em Smarty");
$smarty->assign("chapters", $array);
```

Depois apenas temos de mostrar o template formatado.

```
$smarty->display("index.tpl");
```

O ficheiro index.tpl apenas terá de conter o seguinte texto:

```
<html>
  <head>
      <title>Smarty</title>
  </head>
  <body>
   <h1>{$value}</h1>
    <h2>Capítulos:</h2>
    <ul>
    {foreach $chapters as $chapter}
    <li>{$chapter}</li>
    {/foreach}
    </ul>
  </body>
</html>
```

## Sintaxe do Templates

A sintaxe usada na construção dos templates em Smarty é bastante simples e flexível, completamente abstraída da linguagem (PHP). Os ficheiros são HTML com pequenas partes específicas do Smarty.
Para aumentar a performance da aplicação e evitar gastos de tempo desnecessários ao fazer parse dos ficheiros de

template de cada vez que uma página é vista, o Smarty "compila" os templates para ficheiros PHP, aumentando assim drasticamente a performance. Quando um template é editado, o Smarty detecta-o e recompila o template, não sendo por isso necessária preocupação ao modificar os templates.

A Sintaxe do Smarty é de tal maneira poderosa que é possível incluir lógica dentro dos templates, no entanto esta lógica deverá ser direccionada para a apresentação. Com isto pretendo dizer que a sintaxe do Smarty suporta excepções, ciclos, variáveis, includes e um sem-número de outras coisas fascinantes que não seriam possíveis de abordar apenas num artigo. Por isso, aqui vamos apenas abordar os aspectos mais importantes e mais comuns.

Basicamente, com o Smarty, temos o HTML, no qual podemos adicionar as partes do código que serão interpretadas pelo motor. Estas partes do código estão, geralmente, entre chavetas (**{** e **}**), no entanto isto pode ser mudado. Para isso, basta-nos usar as seguintes linhas de código.

```
$smarty->left_delimiter = "({";
$smarty->right_delimiter = "})";
```

Podemos personalizar os delimitadores que quisermos.

Os delimitadores servem para delimitar cada tag Smarty, pelo que cada tag Smarty imprime o valor de uma variável ou chama uma determinada função. Essa função pode ser interna, ou seja, definida pelo próprio Smarty, ou externa, definida por nós e criada através de plugins. Aqui vamos ver algumas das funções internas do Smarty, mas também algumas funções externas que vêm já definidas com o motor.

### Variáveis

As variáveis são, provavelmente, a coisa mais simples, básica e necessária para o Smarty, pois elas são o método convencional e mais usado para introduzir dados no template, e consequentemente, mostra-los. As variáveis para o Smarty, no PHP, podem ser definidas usando a seguinte função.

```
$smarty->("nome", $conteudo);
```

A parte do nome é o nome que será usado no template para se referenciar à variável, enquanto que o conteúdo é aquilo que a variável vai conter. As variáveis podem ser simples, como strings e integers ou arrays e objectos.

Quando a variável é uma simples string, para a mostrarmos no HTML, basta:

```
{$nome}
```

No entanto podemos querer usar arrays, e para isso teremos de usar funções como Smarty.

Para imprimirmos objectos, é simples e bastante similar ao PHP.

```
{$name>someproperty}
{$name->anothervar}
```

### For

Muitas vezes podemos querer mostrar uma lista, ou algo cujo número de itens não é estático/previamente definido. Isso significa que temos de usar ciclos no próprio template para percorrer um array fornecido pelo script PHP. Para isso usamos o ciclo for ou o foreach.

A sua sintaxe é a seguinte:

```
{for $index=1 to 20 max=3}
 - {$index}<b />
{forelse}
Sem items.
{/for}
```

Isto irá imprimir:

```
 - 1<br />
 - 2<br />
 - 3<br />
```

O parâmetro max é opcional, e o bloco do forelse também. Este é mais importante quando temos valores dinâmicos no inicio e no fim do ciclo, pois estes podem ser nulos ou inválidos (o inicio ser maior que o fim).

### Foreach

O foreach é mais indicado quando o próprio PHP nos retorna um array. Este pode ou não ser associativo. Por exemplo, para imprimir um array associativo, definimos no PHP:

```
$utilizador = array('Nome Real' => 'Pedro Silva', 'Nickname' => 'Scorch', 'E-mail' => 'xpto@email.net');
$smarty->assign('utilizador', $utilizador);
```

Depois escrevemos o seguinte código no template:

```
{foreach $utilizador as $var}
<b>{$var@key}:</b> {$var}<br />
{/foreach}
```

Isto irá imprimir:

```
<b>Nome Real:</b> Pedro Silva<br />
<b>Nickname:</b>Scorch<br />
<b>E-mail:</b>xpto@email.net<br />
```

A lógica é muito similar à do PHP.

### If, Elseif e Else

Como não podia deixar de ser, o Smarty suporta estruturas condicionais. Uma vez que os templates são compilados para scripts PHP, estas estruturas são extremamente poderosas e muito similares às nativas do PHP. Este é um exemplo simples:

```
{if $var == '1'}
Condição if verdadeira.
{elseif $name == '2'}
Condição elseif verdadeira.
{else}
É o que sobra.
{/if}
```

Para uma lista completa de exemplos e das propriedades da estrutura condicional, podem visitar aqui[2].

### Cycle

Esta é uma função muito útil que permite obter valores de forma cíclica. Por exemplo, temos um ciclo for com 10 iterações, e temos uma função cycle com 3 valores, por exemplo: Um, Dois e Três, ou seja, ele vai obter os valores pela seguinte ordem: Um Dois Três Um Dois Três Um Dois Três Um. Mas vamos ver um exemplo.

```
{for $index=1 to 20 max=3}
 - {cycle name="ciclo_1" values="Um,Dois,Três"}
<b />
{/for}
```

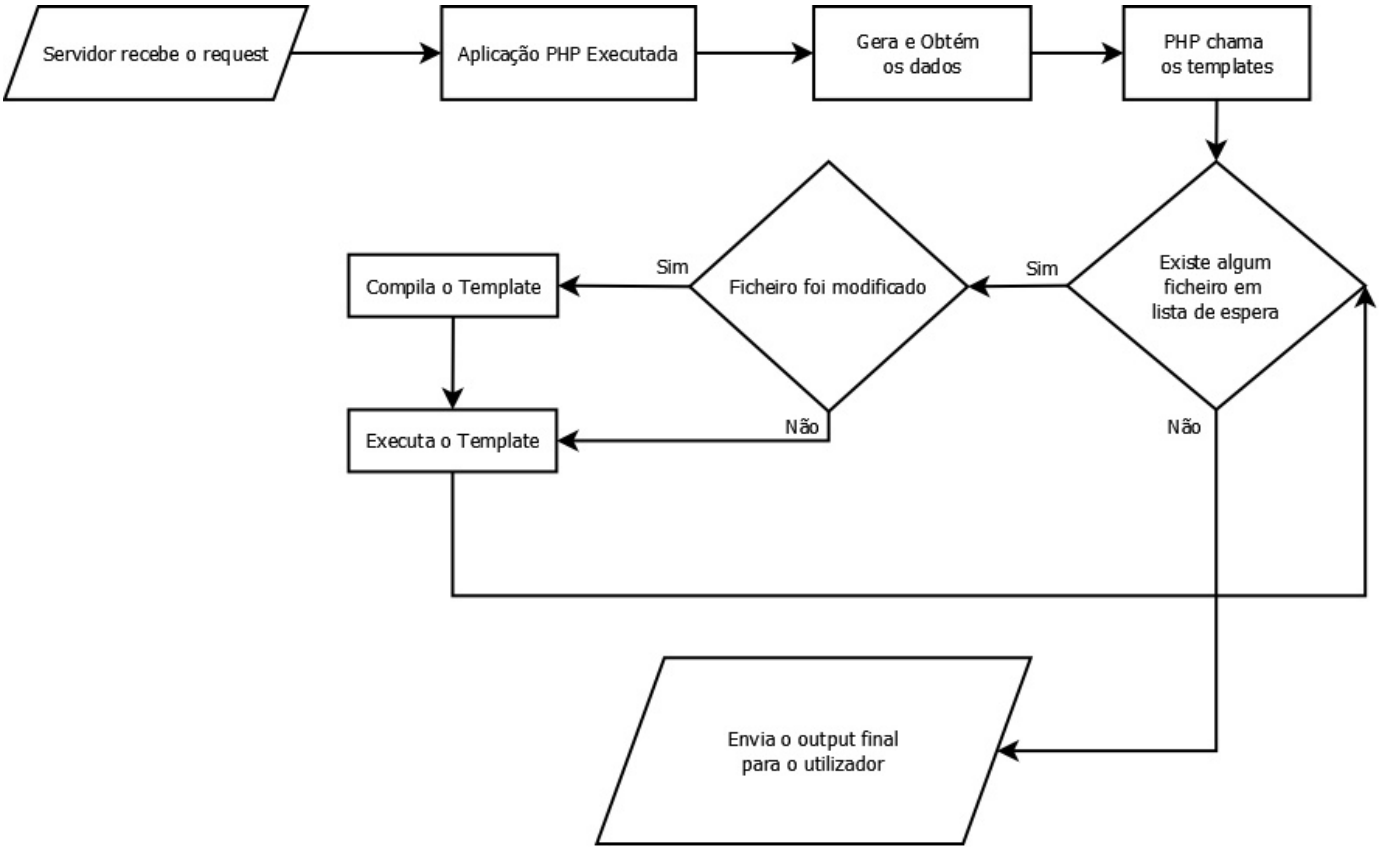
O parâmetro name é opcional, e permite-nos ter mais que um ciclo ao mesmo tempo. O parâmetro values deve conter os valores a percorrer, separados por vírgulas e sem espaços. Apesar da vírgula ser o valor por defeito, pode-se sempre definir um separador através do parâmetro delimiter. O conjunto de todos os parâmetros pode ser visto aqui[3].

Esta função é particularmente interessante quando imprimimos os dados numa tabela ou noutra forma qualquer de listagem, e queremos que a cor de fundo, por exemplo, mude ciclicamente em cada item. Por exemplo, o primeiro ficará branco, o segundo cinzento, o terceiro branco e o quarto cinzento, até ao fim da lista.

### Conclusão

Smarty é um sistema de templates em PHP, que permite uma variedade enorme de funcionalidades e tem uma grande performance, uma vez que os templates são convertidos para código PHP.

A lista completa de funções e documentação pode ser encontrada aqui[4], em Inglês.




Exemplo do funcionamento de uma aplicação a usar Smarty

1) http://smarty.net/download
2) http://smarty.net/docs/en/language.function.if.tpl
3) http://smarty.net/docs/en/language.function.cycle.tpl
4) http://smarty.net/docs/en/index.tpl

Link para o artigo: http://tinyurl.com/RPED28-10

## AUTOR

Escrito por **Pedro Silva**
Estudante, é um apaixonado por computadores e tecnologia. Gosta de Web Development, sendo a área onde se inicou na programação. Também trabalha com várias tecnologia .NET.
Gosta também de multimédia, desporto e fotografia. É membro do staff da comunidade Portugal-a-Programar, e autor de um blog: Blog / @Scorchpt

portugal-a-programar

facebook.com/portugal.programar

twitter.com/pt_programar

# COLUNAS

CORE DUMP - O X No Quadrado Certo

VISUAL (NOT) BASIC - Entity Framework 4:Model/Code-First

# O X No Quadrado Certo

O tema da escolha da universidade é um tópico sazonal e recorrente no P@P, dando aso a vários tópicos e, como é natural, a opiniões opostas.

Sendo um participante assíduo dessas discussões, e estando na altura de pensar de forma clara e séria qual a universidade a escolher, vou partilhar aqui a minha opinião como profissional e empregador na área das Tecnologias de Informação (TI).

Mas antes, comecemos pelo senso comum. Parece-me pacífico que um candidato escolha determinada instituição por vários parâmetros, sendo um deles o da preparação para o mercado de trabalho. Diz-nos o bom senso que optando por uma instituição que é reconhecida como boa pelo mercado de trabalho, a presença de um curso superior no Curriculum Vitae (CV) de um candidato é um bom cartão de visita para ser chamado para uma entrevista.

Quem não tem experiência profissional foca de forma quase exclusiva todos os créditos que tem para apresentar a um empregador no seu percurso académico. Neste ponto, **o nome da instituição que consta no CV do candidato é um bom cartão de visita** e funciona como primeiro filtro. Este primeiro filtro é normalmente usado para fazer duas pilhas de CV, separando os candidatos cujo CV vamos ler com atenção dos candidatos que ficam eliminados logo na primeira ronda. Para os que estão "chocados" ou "revoltados" com esta abordagem crua e até, de certa forma, cruel, tenho apenas a dizer que da mesma forma que eu uso a instituição como primeiro filtro, também as empresas de recursos humanos (RH) o fazem. E não somos os únicos. Este tipo de abordagem é normal para as empresas que operam nesta área.

Há alguns dias atrás (re)confirmei esta situação numa reunião com uma empresa de RH no âmbito do recrutamento de um consultor. Nesta altura muitos de vós estão a pensar que isto não é verdade porque têm conhecimento de pessoas que foram a entrevistas e cuja

instituição onde se licenciaram não é um dos pesos fortes no mercado. Acontece que essa situação não é um contra-exemplo do que se passa. As empresas de RH vivem da colocação de pessoas em empresas, pelo que é do seu interesse entrevistas todas as pessoas e tentar colocá-las numa empresa, esse é o seu negócio. Há empresas, e actividades, onde a formação de base não requer uma preparação tão exigente, fazendo com que esses candidatos sejam boas escolhas, ficando contentes as três partes envolvidas.

Uma vez passado o choque, voltemos ao essencial: **quando enviam um CV em que pilha é que querem que o mesmo seja colocado**? A resposta a esta pergunta começa muitos anos antes da larga maioria das pessoas escrever o seu primeiro CV, começa quando colocam o X no quadrado certo, no quadrado que vos vai permitir ser reconhecidos pelo mercado como estando, à partida, mais bem preparados. A verdade é que o mercado sabe muito bem quem são as instituições que melhor preparam as pessoas para a vida profissional, e tipicamente isso é valorizado. Para os que se lembram do tempo da bolha, nessa altura quase qualquer um podia ingressar e fazer uma carreira em TI, tal era a necessidade de recursos que as empresas, em particular as consultoras, tinham. Mas esses tempos já lá vão, e num mercado cada vez mais

difícil e num país com uma taxa de desemprego tão elevada, todos os argumentos são preciosos, e um carimbo de uma instituição reconhecida pelo mercado tem bastante peso.

Com a introdução do regime de Bolonha, algumas empresas viram essa situação como um retrocesso educativo na preparação dos estudantes para o mercado de trabalho. Para garantir o mesmo nível de preparação, essas mesmas empresas passaram a exigir aos seus candidatos não uma licenciatura mas sim um mestrado. O mestrado é uma boa forma de aumentar os conhecimentos mas é também uma forma das empresas darem menos peso à instituição onde um candidato se licenciou. No entanto, as regras são as mesmas da licenciatura, o valor do carimbo das instituições que ministram mestrados têm pesos diferentes no mercado de trabalho. O mestrado proporciona também uma especialização e uma maneira de preparar o direccionamento da carreira em TI. Estas duas vantagens são muito interessantes tanto para para as empresas, que valorizam os seus recursos e investem na investigação e desenvolvimento, como para os candidatos, que podem querer evoluir por uma vertente mais estimulante para si.

Nesta altura alguns já se estão a questionar sobre os que não têm curso superior. Para todos os que não têm curso superior e querem ingressar nesta vida, as dificuldades são acrescidas. No entanto há outras formas de começar. No que toca à parte técnica e tecnológica é fácil encontrar informação e estudar por si mesmo ou tirar cursos ou mesmo certificações. No entanto não se devem ter ilusões, mesmo assim será mais difícil iniciar uma carreira em TI. É que ao contrário do que muitos jovens pensam, **numa licenciatura de informática não se ensina a programar**. Os conhecimentos adquiridos ao longo do curso são muitos e variados, alguns até são de utilidade duvidosa, mas são valiosos. Essencialmente uma licenciatura ensina a pensar, a analisar, ensina a descobrir soluções e alarga os horizontes dos estudantes, fazendo com que estes tenham contacto com imensas coisas novas e diferentes. Numa comparação directa, não é de estranhar que quem saia de uma universidade reconhecida pelo mercado tenha mais facilidade de encontrar um bom emprego e de fazer carreira em TI.

Se após toda a minha argumentação não estão convencidos, então guiem-se pelo vosso bom senso. Esse, aposto que vos diz para colocarem os X nos quadrados certos quando preenchem o formulário de candidatura ao ensino superior.

Link para o artigo: http://tinyurl.com/RPED28-09

## 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.

# Entity Framework 4.0: Model-First e Code-First

Existem três tipos de abordagens quando estamos a utilizar Entity Framework 4.0: *database-first* onde são criadas as nossas entidades (classes) usando uma base de dados já existente; *model-first* onde é criado o nosso modelo conceptual e, com base nele, é gerado um script para a criação da base de dados; e *code-first* onde é utilizado POCO (Plain Old Code CRL) para criação manual de toda a lógica de entidades e ligações, não perdendo no entanto, todas as vantagens da utilização do Entity Framework.

Na edição da Revista PROGRAMAR nº 26, de Dezembro de 2010, abordei a utilização do modelo *database-first*, mostrando como criar as entidades e como efectuar algumas operações CRUD (acrónico para Create, Read, Updade e Delete).

Neste artigo irei abordar de uma forma geral como utilizar as restantes abordagens: *model-first* e *code-first*.

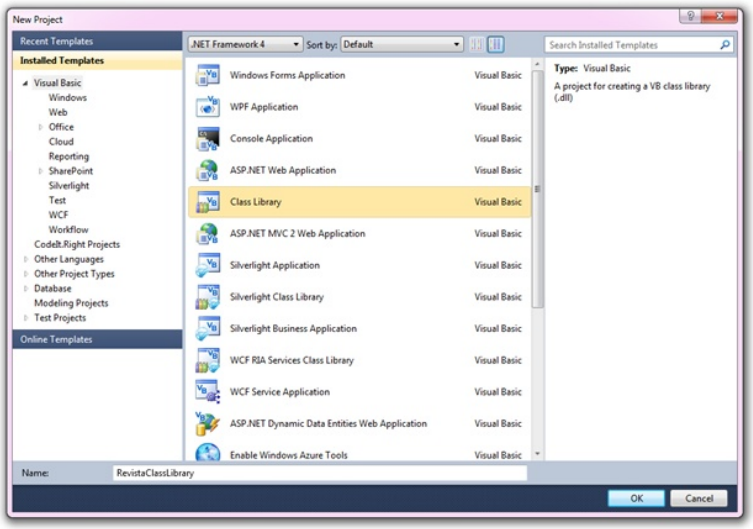
**Abordagem Model-First**

Esta abordagem permite-nos criar o nosso modelo conceptual, usando o Visual Studio, e depois, com base neste, criar a base de dados. Após a criação do modelo é criada a DDL (Data Definition Language), que será guardado num ficheiro *.sql e que nos permite então criar a base de dados.

Existem algumas vantagens desta abordagem pois é uma forma de trabalhar onde temos o nosso modelo e não nos precisamos de preocupar com a base de dados ou como esta irá ser construída. Não necessitamos também de ter conhecimentos muito específicos de bases de dados (como criar tabelas, relações, etc.), sendo tudo feito no Visual Studio de uma forma simplificada. Além disso, ficamos com o DDL que nos permite criar a base de dados em qualquer altura (por exemplo após a instalação da aplicação).

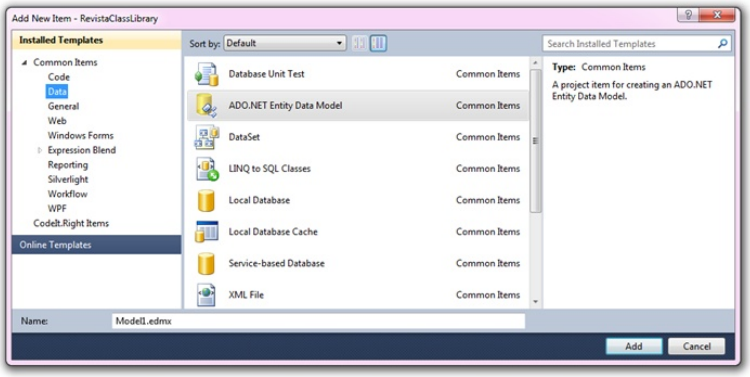
Neste exemplo será criada uma Class Library, permitindo assim isolar a lógica de acesso a dados e, caso seja necessário, reutilizar em diferentes projectos.

Para a criação do nosso modelo, numa abordagem *model-first*, criamos um novo projecto no Visual Studio 2010 definindo como Framework a 4.0, e com o nome "RevistaClassLibrary".



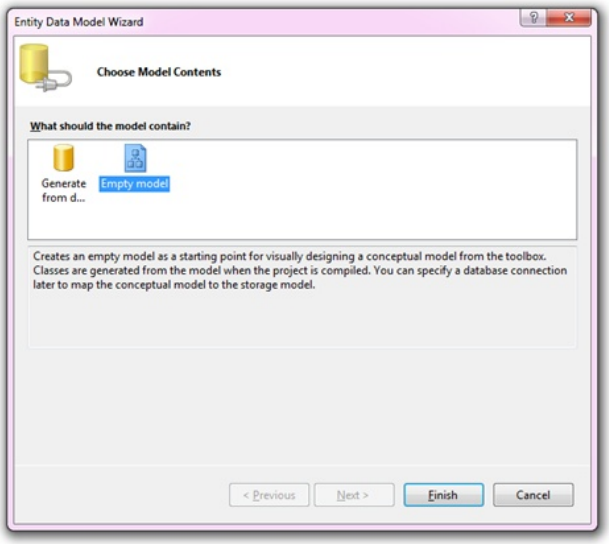

NOTA: Após o projecto criado podemos apagar a classe que aparece por defeito (Class1.vb) pois não será necessária.

Adicionamos um novo item, recorrendo aos templates no separador Data - ADO.NET Entity Data Model. Esta opção, que iremos definir com o nome RevistaModel.edmx, irá criar um ficheiro *.edmx que irá representar o nosso modelo.

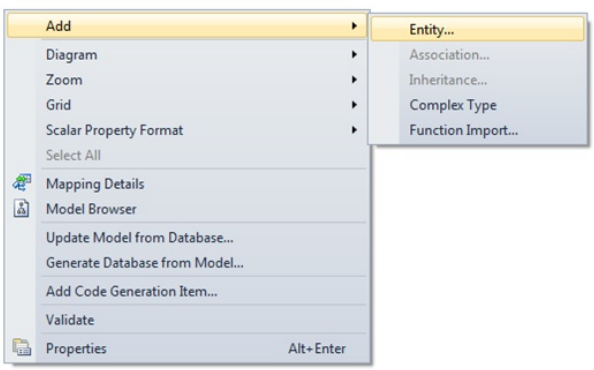


O *.edmx é um ficheiro XML que define o modelo conceptual, o modelo de dados e as relações entre as diferentes entidades.

Na seguinte opção, podemos escolher se queremos gerar um modelo de uma base de dados, como foi abordado no artigo anterior na Revista PROGRAMAR edição 26, ou criar um modelo vazio. Iremos escolher a segunda opção – Empty model.

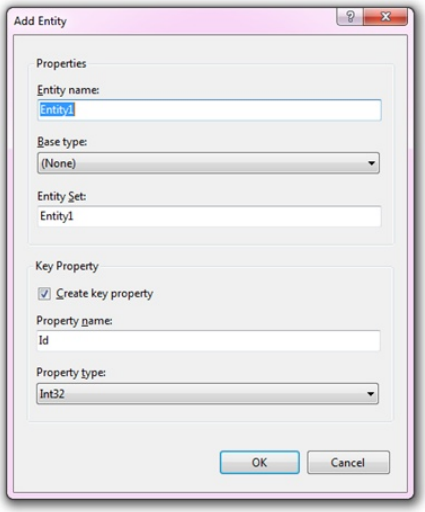



Com o nosso documento criado, ainda vazio, existem duas ferramentas importantes para o desenvolvimento do modelo: a Toolbox, que tem agora objectos específicos para o Entity Framework e o Model Browser que permite explorar o nosso modelo. Existem duas formas de criar o modelo: através da Toolbox, já referida anteriormente, ou clicando com o botão direito do rato sobre a janela aberta. Esta segunda opção é mais simples de utilizar.




Ainda antes de iniciarmos a construção do nosso modelo conceptual, existe uma opção interessante, se olharmos para a janela das propriedades: Pluralize New Objects. Esta opção, caso esteja definida como verdadeira (True), irá automaticamente tentar pluralizar os nomes das entidades. Infelizmente não funciona na versão Portuguesa, mas se utilizarmos designações em Inglês, é muito interessante e prática.

Para este exemplo vamos então criar um modelo muito simplificado que permita representar as edições da revista PROGRAMAR. Vamos criar duas entidades que vão representar os artigos e os seus autores.

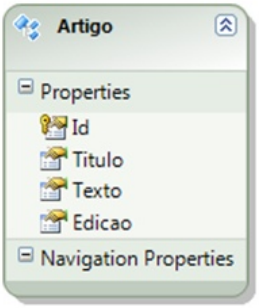


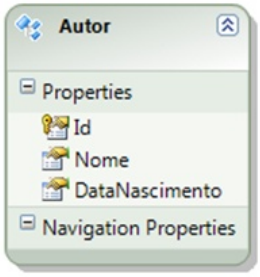

Por defeito, quando criamos uma nova entidade, ele inclui a criação de um identificador (chave primária), que podemos, caso não seja necessário, retirar. Não define também nenhuma herança, podendo esta ser indicada no “Base Type”, ou seja, podemos definir heranças entre entidades, seleccionado a entidade correspondente.

Adicionando algumas propriedades, e definindo os tipos de dados correctos (ex. DataNascimento = DateTime ou Edicao = Int32), rapidamente construímos os modelos para representar as edições da revista. De notar que por defeito, quando é adicionada uma nova propriedade à entidade, o tipo de dados está definido como String, com um tamanho de armazenamento máximo para este tipo de dados (2^3-1) - nvarchar(max). Podemos e devemos alterar, caso seja necessário, assim como explorar e ajustar as restantes propriedades, como por exemplo, o tamanho máximo para

uma String (Max Lenght), qual o valor por defeito (Default Value), se permite valores nulos (Nullable), etc.

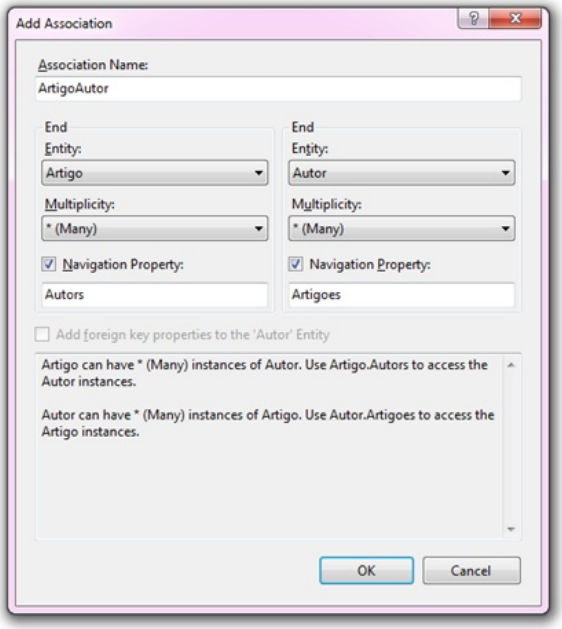
E já estão então criadas as duas entidades e neste



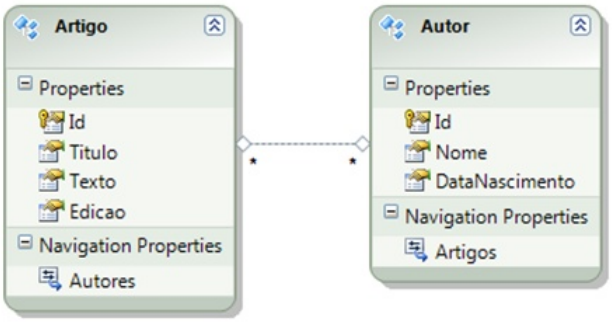

momento só nos falta definir a relação entre ambas. Como um artigo pode ter vários autores e um autor pode ter vários artigos, necessitamos de criar uma relação muitos para muitos (many-to-many).

Ao adicionarmos a relação entre as entidades, usando a opção Association, é criado um novo campo no final,

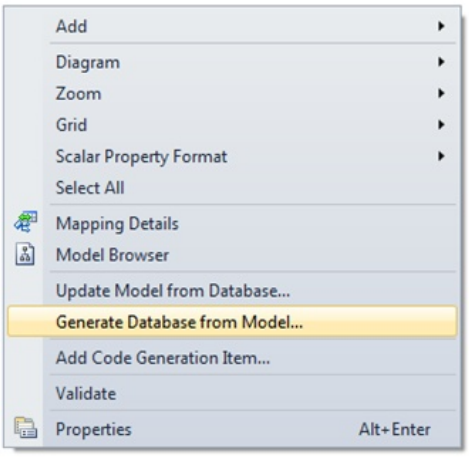



designado por Navigation Property, que irá permitir a navegação entre as entidades. Podemos alterar o nome da propriedade de navegação (caso seja necessário).

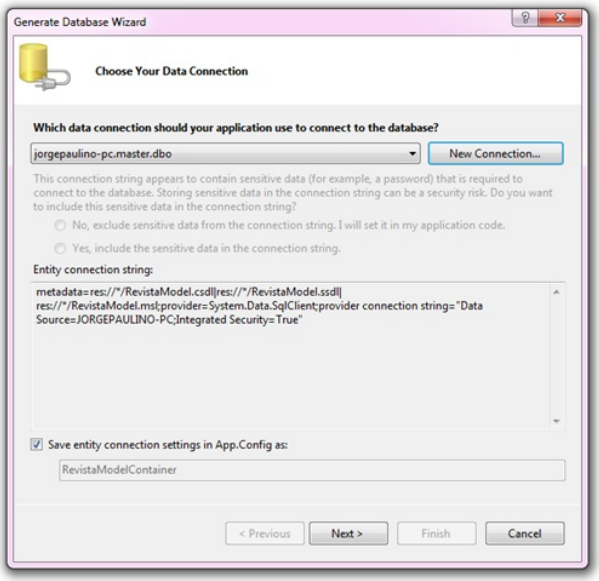
E já está! Este foi o último passo para criar o modelo conceptual e é agora altura de gerar a base de dados. Para



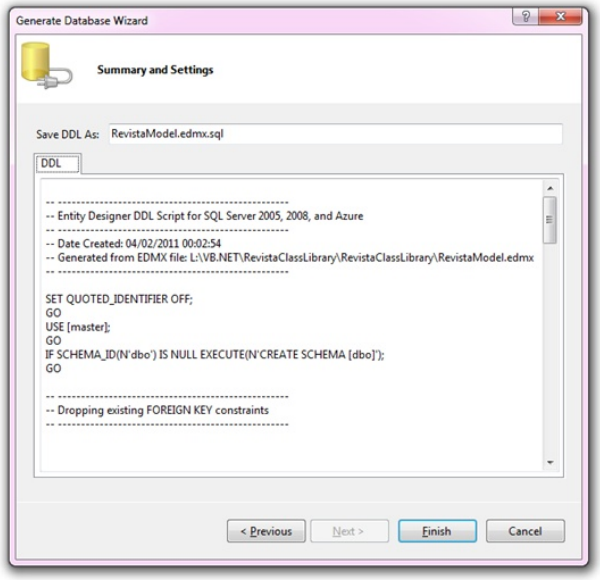

o fazer é só necessário clicar com o botão direito do rato sobre o editor e seleccionar a opção “Generate Database from Model”.
Irá então aparecer um *wizard* que nos permite seleccionar uma ligação já existente ou criar uma nova.



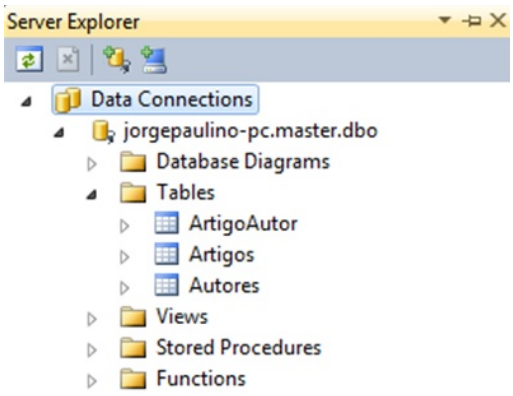

Seleccionado a ligação que pretendemos, irá ser gerado um script DDL que permitirá, após execução, criar uma



base de dados em SQL Server 2005, 2008 e Azure.




Reparem que apenas criamos duas entidades (autores e artigos), mas como definimos uma relação muitos para muitos (many-to-many), é necessário utilizar uma tabela auxiliar. O Entity Framework fez isso por nós e no SQL ficamos então com três tabelas, como mostra a seguinte imagem:




Se no Solution Explorer escolhermos a opção “Show All Files”, podemos ver que o nosso modelo (*.edmx) tem um ficheiro com a extensão *.vb (neste caso RevistaModel.Designer.vb) . Neste ficheiro podemos ver e o código que está por detrás do nosso modelo conceptual, e efectuar eventualmente, algumas alterações.

Ao adicionarmos um novo projecto a esta solução (*.sln), que irá representar a camada de apresentação (Presentation Tier), ou usando externamente, é necessário adicionar referências à classe que criamos (separador Project) e a System.Data.Entity (separador .NET). É necessário também copiar a Connection String, que se encontra no ficheiro de configuração (app.config), uma vez que esta foi definida na Class Library, no momento de ligação à base de dados.

```
<connectionStrings>
<add name="RevistaModelContainer"
connectionString="metadata=
res://*/RevistaModel.csdl|
res://*/RevistaModel.ssdl|
res://*/RevistaModel.msl;
provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=.;Initial
Catalog=master;Integrated Security=True;
MultipleActiveResultSets=True&quot;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
```

A Connection String tem um conjunto de metadata que indica a localização do ficheiro de CSDL (Conceptual Schema Definition Language), do SSDL (Store Schema Definition Language) e do MSL (Mapping Schema Language). O asterisco (*) indica que estes ficaram embebidos no ficheiro binário, podendo isto ser alterado, indicando nas propriedades do modelo conceptual, que o Metadata Artifact Processing não esta Embed in Output Assembly mas sim Copy to Output Directory. Indica depois a ligação propriamente dita à base de dados.

**Abordagem Code-First**

Esta é outra abordagem que podemos utilizar em Entity Framework, além das já referidas neste artigo (*model-first*) e no artigo da edição 26 da revista PROGRAMAR (*database-first*).

Uma das vantagens do *code-first* (ou como é também designado *code-only*), utilizando POCO (Plain Old CLR objects ), é que não ficamos “presos” ao Entity Framework, pois todas as classes geradas automaticamente herdam da classe EntityObject. Além disso utilizamos as classes que queremos, com muito menos código (um exemplo mesmo pequeno como este tem mais de 300 linhas de código), o torna a manutenção da aplicação muito mais simples.

Para usar esta abordagem, onde somos nós que desenhamos as classes que vão representar as entidades, podemos seleccionar o nosso modelo conceptual (*.edmx) e na janela de propriedades, na opção “Code Generation Strategy”, que está definida para Default, seleccionamos None. Isto fará com que o código gerado automaticamente seja apagado, possibilitando desta forma o desenvolvimento personalizado.

Para representar o modelo conceptual mostrando anteriormente, e tendo em conta que este é um exemplo muito simples (para efeitos de demonstração apenas), necessitamos apenas de três classes: Artigo, Autor e RevistaModelContainer.

As duas primeiras classes ( Artigo e Autor), vão representar as duas entidades e definem as propriedades da classe e a associação entre ambas (usando uma ICollection). Podemos ter mais propriedades e mais métodos nas classes, mas para que isto funcione, temos de ter pelo menos as que estão definidas no modelo.

```
Imports System.Collections
Imports System.Data.Objects

Public Class Artigo

  Public Property Id As Integer
  Public Property Titulo As String
  Public Property Texto As String
  Public Property Edicao As Integer

   ' Overridable para permitir o LazyLoading
   Public Overridable Property Autores() As
ICollection(Of Autor)

  Sub New()
    Autores = New List(Of Autor)
  End Sub

End Class

Public Class Autor
  Public Property Id As Integer
  Public Property Nome As String
  Public Property DataNascimento As DateTime

   ' Overridable para permitir o LazyLoading
   Public Overridable Property Artigos() As
ICollection(Of Artigo)

  Sub New()
    Artigos = New List(Of Artigo)
  End Sub
End Class
```

**IMPORTANTE**: A declaração de variáveis em Visual Basic não é, como sabem, *case sensitive*. No entanto, na declaração das propriedades das entidades (classes) é obrigatório que se declarem de acordo com o modelo (*.edmx), ou seja, há distinção entre maiúsculas e minúsculas.

A terceira classe é onde se define o nosso Container, que herdas do ObjectContext, ou seja, é onde criamos a ligação entre o Entity Framework e as nossas classes anteriores. O ObjectSet é novo na versão 4.0 do Entity Framework e permite-nos trabalhar os dados como colecções, permitindo também executar queries.

```
Public Class RevistaModelContainer
  Inherits ObjectContext

  Sub New()

    ' Indica a ConnectionString (ver em
    ' app.config) e nome do Container
     MyBase.New("name=RevistaModelContainer",
"RevistaModelContainer")

    ' Cria as instancias dos ObjectSets
    _Artigos = MyBase.CreateObjectSet(Of
            Artigo)("Artigos")
    _Autores = MyBase.CreateObjectSet(Of
            Autor)("Autores")
```

```
        ' Define que o LazyLoading está
        ' activo (por defeito não está)
        MyBase.ContextOptions.LazyLoadingEnabled
= True

    End Sub

    Private _Artigos As ObjectSet(Of Artigo)
    Public ReadOnly Property Artigos()
                        As ObjectSet(Of Artigo)
    Get
        Return _Artigos
      End Get
    End Property

    Private _Autores As ObjectSet(Of Autor)
    Public ReadOnly Property Autores()
                         As ObjectSet(Of Autor)
      Get
        Return _Autores
      End Get
    End Property

End Class
```

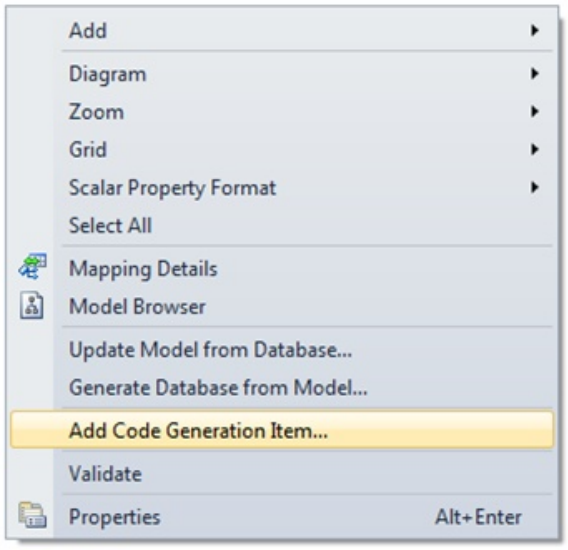
Como é possível ver neste codigo, habilitamos a opção LazyLoadingEnable do ObjectContext, colocando-a a True. É necessário também definir as propriedades como Overridable para que o Entity Framework saiba que propriedades usar. O Lazy Loading permite que as entidades relacionadas sejam automaticamente carregadas da fonte de dados quando acedemos às propriedades de navegação (neste caso Autores e Artigos).

E é tudo! Com poucas linhas de código criamos as nossas classes, que definem as nossas entidades, e criamos o nosso Container que permite interligar as classes com o Entity Framework.

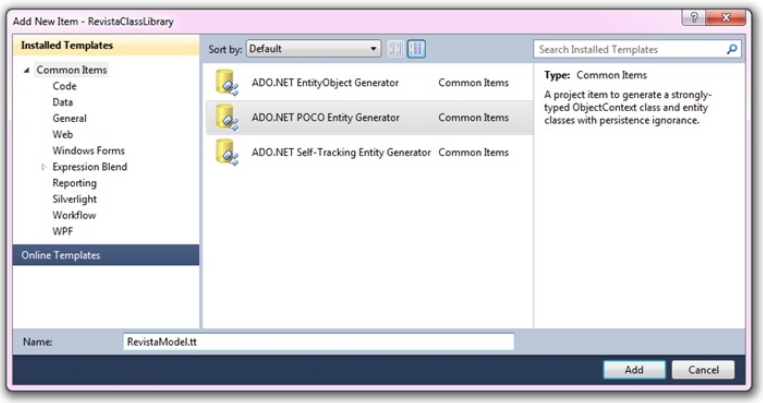
Mas este é apenas um exemplo bastante simples e caso usássemos um modelo complexo, com inúmeras entidades e associações, era muito trabalhoso criar todas estas classes POCO.

Por isso o Entity Framework permite a utilização/criação de templates que geram o código por nós. São designados por T4 (Text Template Transformation Toolkit) Text Templates e são ficheiros de texto com uma extensão *.tt .

Existe um template para a criação de classes POCO. Não está disponível nos templates do Visual Studio 2010, mas se abrirmos o modelo conceptual e seleccionarmos “Add Code Generation Item”, podemos pesquisar em "Online Templates" e instalar.

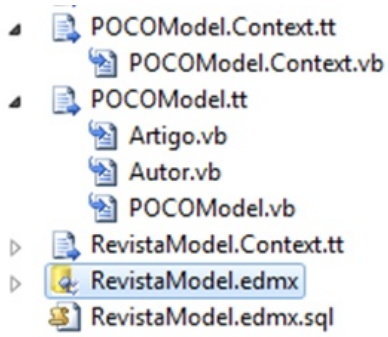



Após a rápida instalação, podemos então seleccionar ADO.NET POCO Entity Generator.




Isto irá adicionar dois ficheiros *.tt ao projecto: POCOModel.tt e POCOModel.Context.tt. O POCOModel tem as classes que representam as entidades e o

POCOModel.Context tem as classe de contexto.




*NOTA: Ao adicionarmos os templates ao projecto, todo o código de Entity Framework será apagado.*

Desta forma, e muito rapidamente, criamos as nossas classes POCO com a ajuda de templates.

Como foi possível ver ao londo deste artigo, e do artigo da edição nº 26 da revista PROGRAMAR, existem diferentes abordagens para a utilização do Entity Framework 4.0, permitindo criar um modelo relacional de uma base de dados, criar um modelo e com base neste criar a base de dados ou criando as nossas classes (entidades) usando POCO, ficando desta forma independentes e com a possibilidade de uma maior personalização de toda a lógica. A existencia e utilizaçao de templates permite simplificar o processo de criação de classes POCO e podemos inclusivé criar os nossos próprios templates.

No momento da criação deste artigo está já disponível o Entity Framework 4.1 Release Candidate que tem como principal novidade a DbContext API, que é uma abstracão simples do ObjectContex e que pode ser utilizada em qualquer abordagem (*Database-First*, *Model-First* e *Code-First*). Existe também algumas alterações na abordagem Code-First.

**Alguns endereços interessantes**:

ADO.NET team blog
http://blogs.msdn.com/b/adonet/

Beginner's Guide to the ADO.NET Entity Framework
http://msdn.microsoft.com/en-us/data/ee712907

EF 4.1 Release Candidate Available
http://blogs.msdn.com/b/adonet/archive/2011/03/15/ef-4-1-release-candidate-available.aspx

EF 4.1 Model & Database First Walkthrough
http://blogs.msdn.com/b/adonet/archive/2011/03/15/ef-4-1-model-amp-database-first-walkthrough.aspx

EF 4.1 Code First Walkthrough
http://blogs.msdn.com/b/adonet/archive/2011/03/15/ef-4-1-code-first-walkthrough.aspx

Link para o artigo: http://tinyurl.com/RPED28-01

## AUTOR

Escrito por **Jorge Paulino**
Exerce funções de analista-programador numa multinacional sediada em Portugal. É formador e ministra cursos de formação em tecnologias Microsoft .NET e VBA. É Microsoft Most Valuable Professional (MVP), em Visual Basic, pela sua participação nas comunidades técnicas . É administrador da Comunidade Portugal-a-Programar e membro de várias comunidades (PontoNetPT, NetPonto, MSDN, Experts-Exchange, CodeProject, etc). É autor do blog http://vbtuga.blogspot.com - twitter @vbtuga

# COMUNIDADES

AndroidIPC - Inter Process-Communication

Automatização de deployments em Windows Azure

# AndroidIPC - Inter Process-Communication

As aplicações de Android correm sobre uma máquina virtual de Java, de nome Dalvik. Uma das particularidades desta máquina virtual, é que as aplicações estão limitadas a uma Sandbox de execução, ou seja, o seu espaço de memória é privado. Este sistema é muito semelhante ao comportamento as aplicações nativas do sistema operativo subjacente do Android, o GNU/Linux.

Uma das consequências disto, é que a troca de informação entre aplicações excluí logo à partida a partilha de informação através de memória partilhada. O que resta, é a troca de mensagens entre aplicações.
O Android fornece vários mecanismos para envio de mensagens entre aplicações, alguns são mais apropriados que outros, dependendo do cenário de utilização. Temos por exemplo os BroadcastReceivers, que se tratam de classes que recebem informação que é enviada para o sistema operativo, e este trata de retransmitir a informação para as várias aplicações que se registaram como "Receivers" para esse tipo de informação.
Este método não é o ideal em muitos casos, veja-se por exemplo que a informação fica passível a ser enviada para várias aplicações e não só para uma, e que a comunicação é só numa direcção, ou seja, a mensagem é enviada, recebida, mas não há sequer confirmação de que a mesma foi recebida, nem é possível retornar qualquer tipo de informação. Existe ainda o senão de que num ambiente em que existe mais que um BroadcastReceiver, o sistema operativo envia a informação para a 1ª aplicação, e se esta assim decidir pode consumir a mensagem e não a passar de volta para as seguintes.

Para resolver, este e outros cenários, a SDK fornece um mecanismo de troca de mensagens entre aplicações, aquilo que usualmente se chama de IPC (InterProcess-Communication).

Este mecanismo é apoiado por uma linguagem de ligação das duas aplicações que queremos que comuniquem entre si, chamada de AIDL[1] (Android Interface Definition Language). Esta linguagem tem uma sintaxe bastante simples, e consiste apenas na declaração da assinatura dos métodos responsáveis pela comunicação.

A título de exemplo, vamos produzir 2 aplicações muito simples, de nomes appA e appB, que comunicarão entre si. A appA enviará uma mensagem à appB com o pedido para calcular o produto de 2 números inteiros. A appB receberá estes parâmetros na mensagens, e devolverá o resultado do produto dos mesmos. Apesar da simplicidade do exemplo, servirá para demonstrar os problemas envolvidos na produção da solução, tais como, o que fazer quando a aplicação com que estamos a comunicar não existe.

O primeiro passo será definir o ficheiro AIDL. Este ficheiro contém a definição da função que a appA irá chamar.

```
package ipc;

interface ServicoIPC {
int multiplicar(in int num1, in int num2);
}
```

Repare que ambos os argumentos têm o prefixo "in". Isto faz parte da especificação AIDL, neste caso é "in" porque estamos a lidar com tipos de dados simples de Java, e esses apenas podem ser do tipo "in".

O ficheiro que contem o código em cima especificado deverá estar presente tanto na appA como na appB, na pasta src ou em dentro de qualquer package dentro da mesma. O ficheiro terá obrigatoriamente de ter a extensão .aidl , uma vez que em tempo de compilação estes ficheiros são tratados de forma especial atendendo à sua extensão.

Do lado da appB, que é aplicação que irá receber a mensagem, e fazer o cálculo, temos de declarar um Service de Android para lidar com a recepção da mensagem. O código em baixo apresentado ilustra esta situação.

```
package com.exemplo.appb;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;
import android.util.Log;

public class ServicoRemoto extends Service {

 @Override
 public void onCreate() {
   super.onCreate();
 }

 @Override
 public IBinder onBind(Intent intent) {

   return new ServicoIPC.Stub() {

     /**
      * Método que calcula o produtor de num1
com num2
      */
       public int multiplicar(int num1, int
num2) throws RemoteException {
            Log.d("appB","A appB recebeu um
mensagem para calcular o produto de "+num1+"
com "+num2);
       return num1 * num2;
     }

   };
 }
}
```

Após isto, temos obviamente de adicionar o Service criado em cima ao Manifest da aplicação, neste caso da appB. É também aqui que vamos expor o Service ao exterior. Fazemos isso adicionando as linhas coloridas ao ficheiro AndroidManifest.xml :

```
...
<application â€¦>
        <service android:name="ServicoRemoto"
android:process="com.exemplo.appb.SetPrefsSer
vice">
              <intent-filter>
                    <action
android:name="com.exemplo.appb.IPC" />
              </intent-filter>
              </service>
</application>
...
```

Do lado da appA, temos agora de efectuar a ligação com a appB. Primeiro temos de garantir que temos o ficheiro .aidl na árvore de fonte de código da mesma, como já tinha sido referido. É importante também garantir que o ficheiro AIDL esteja na mesma package de Java em ambas as aplicações.

Por motivos de escalabilidade convém separar a lógica da ligação e do envio da mensagem, do resto do código, assim fica mais simples usar e manter o uso da comunicação inter-procedual conforme a aplicação vai crescendo.

Para tal, vamos fazer uma classe que implementa o ServiceConnection. O seguinte código exemplifica uma classe deste género. Note-se que seria nesta classe que o programador pode gerir os pedidos, a altura em que são enviados, possíveis filas de prioridade de pedidos, etc.

```
package com.exemplo.appa;


import android.content.ComponentName;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.os.RemoteException;
import android.util.Log;

public   class   ConServicoRemoto   implements
ServiceConnection{

       ServicoIPC service = null;
```

```
        private boolean estaVivo= false;


        /**
         * Constructor.
         */
        public ConServicoRemoto() {
                // construtor vazio
        }


        public int adicionarRemoto(int val1,
int val2){
                if (service==null)
                {
                        Log.d("appA","O Serviço
não está ligado. Causa possível: appB não
está instalada");
                        return 0;
                }
                try {
                        return
service.multiplicar(val1, val2);
                } catch (RemoteException e) {

Log.d("appA","RemoteException ao tentar fazer
pedido de multiplicar à  appB");
                        e.printStackTrace();
                        return 0;
                }


        }


        @Override
        public                              void
onServiceConnected(ComponentName          name,
IBinder boundService) {
                service                         =
ServicoIPC.Stub.asInterface((IBinder)
boundService);
                Log.d("appA","Ligou-se    à
appB");
                        estaVivo=true;
        }

        @Override
        public                              void
onServiceDisconnected(ComponentName name) {
                service=null;
                estaVivo=false;
                Log.d("appA","Desconectou    da
appB");
        }

        /**
         *
          * @return True se a ligação à
aplicação appB estiver activa. False caso
contrário.
         */
        public boolean ligacaoEstaActiva(){
                return estaVivo;
        }

}
```

Agora falta apenas fazer uma Activity de Android, ainda na appA, a exemplificar o uso da classe acima declarada.

Para a nossa Activity, declaramos na pasta “layout” o seguinte ficheiro main.xml que será a Interface Gráfica da mesma.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk
/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText android:id="@+id/valor1"
```

```
    android:layout_width="60dip"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dip" />

        <EditText android:id="@+id/valor2"
    android:layout_width="60dip"
        android:layout_height="wrap_content"
        android:layout_below="@id/valor1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dip" />

        <Button android:id="@+id/btn_calcular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/valor2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dip"
        android:text="Calcular"/>

        <TextView android:id="@+id/resultado"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:layout_below="@id/btn_calcular"
        android:layout_marginTop="40dip"
        android:layout_centerHorizontal="true"
        android:text="O   resultado   aparecerá
aqui"
        />
</RelativeLayout>
```

E por fim temos o código da classe de nome Principal que fará uso desta interface gráfica acima declarada, e da classe ConServicoRemoto.

## Conclusão

Repare-se que o código é simplista, não existem verificações se o utilizador de facto escreve números inteiros, e não existe tratamento de erros na classe ConServicoRemoto. Isso fica ao critério do leitor, uma vez que este exemplo está focado unicamente no tema abordado.

## Bibliografia

1. AIDL na documentação oficial da SDK de Android: http://developer.android.com/guide/developing/tools/aidl.html

Link para o artigo: http://tinyurl.com/RPED28-03

## AUTOR

Escrito por **Pedro Veloso**
Licenciado em Ciências da Computação na Universidade do Minho, trabalha actualmente na EmergeIT. Desenvolve profissionalmente para Android, e tem interesse especial na área de Unix, em particular Linux, e administração de sistemas deste tipo. Faz parte integrante das comunidades androidPT e GTUG Portugal.

# Automatização de deployments em Windows Azure

Primeiro que tudo, e para que todos nos possamos encontrar com a mesma base de conhecimento, parece-nos importante contextualizar todos os leitores quanto ao tema em questão.

**O que é então Cloud computing?**

Cloud computing é, numa definição abrangente, uma abordagem à computação assente sobre os conceitos de escalabilidade e alta disponibilidade de processamento e armazenamento online, disponível a um inúmero número de dispositivos e endpoints.

Apresentando-se este como um mercado emergente e potencialmente bastante rentável, principalmente porque o "factor custo" passa a ter um papel fulcral e preponderante nas decisões ao longo de todo o planeamento e desenvolvimento aplicacional, algumas empresas de referência na área das TI não poderiam ficar de fora da corrida.

Entre as várias ofertas que existem hoje no mercado, parece-nos importar salientar o caso do Google App Engine da Google, o AWS e o EC2 da Amazon, as inúmeras ofertas da Rackspace ou mesmo da Salesforce, e finalmente o Windows Azure Platform da Microsoft. Será sobre esta última que focaremos as nossas atenções, sem qualquer desprimor para as restantes ofertas mencionadas.

A plataforma oferecida pela Microsoft é composta então por três grandes blocos: Windows Azure, SQL Azure e Windows Azure AppFabric.

O Windows Azure disponibiliza capacidades de computação e armazenamento escalável, elástico e altamente disponível, bem como uma gestão completamente automatizada dos serviços usados, possibilitando realizar todas estas tarefas com recurso a ferramentas, tecnologias e linguagens de programação já conhecidas pelos programadores. Além de tudo isto, este


bloco possibilita também o acesso através de ligações remotas para as máquinas que estivermos a usar e a possibilidade de fazer uso de VPNs. De ressalvar que estas redes virtuais não têm de ser apenas entre máquinas que se encontram na cloud, mas podem também ser entre máquinas que se encontram na cloud e máquinas que se encontram on-premises.

Relativamente ao SQL Azure, este bloco possui a oferta da Microsoft ao nível dos sistemas relacionais de base de dados. Esta versão é em tudo parecida com o SQL Server 2008 R2, apresentando no entanto algumas restrições. Neste momento encontram-se disponíveis dois tipos (edições) de bases de dados que podemos criar num servidor SQL Azure: Web Edition e Business Edition.

As diferenças fundamentais entre estas duas hipóteses pretendem-se com o tamanho máximo de uma base de dados e com o custo associado às mesmas. As versões Web são bases de dados que serão normalmente utilizadas para ambientes de baixos requisitos, pois possuem limites de 1GB e de 5GB. Já as bases de dados associadas a edições Business possuem limites máximos de 10GB, 20GB, 30GB, 40GB e 50GB e serão normalmente utilizadas para suporte a aplicações de negócio complexas (Line of Business Applications). Neste bloco encontram-se ainda incluídos os sistemas de reporting e de sincronização de dados.

Para por término a esta contextualização relativamente aos blocos disponíveis, falta pois falar do Windows Azure AppFabric. Este bloco inclui então acesso a um Service Bus, a mecanismos de federação e controlo de acessos, bem como a mecanismos de caching. Mais uma vez, todos estes mecanismos são geridos e mantidos de forma automatizada e não imputável a nós.

Sendo que o objectivo deste artigo é falar sobre automatização de deployments em Windows Azure, é imperativo regressarmos a este bloco para falarmos de forma mais aprofundada sobre o mesmo e sobre os serviços que o compõem. O poder computacional disponível encontra-se seccionado em cinco segmentos distintos: XS (Extra Small), S (Small), M (Medium), L (Large) e XL (Extra Large). Estes segmentos são progressivos, isto é, a capacidade computacional e de armazenamento local em cada uma das máquinas incluídas vai aumentando de forma gradual.

Uma instância XS possui processamento partilhado, 768MB de memória RAM e 20GB de armazenamento local, apresentado um custo de $0.05 por cada período horário, sendo que uma instância S possui já 1 CPU Core dedicado, 1.7GB de RAM e 250GB de armazenamento local, apresentando um custo horário de $0.12. Se precisarmos de uma máquina com 2 CPU Cores dedicados com 3.5GB de RAM e 500GB de armazenamento, então o segmento M será a resposta por um custo horário de $0.24. Se precisarmos de capacidades de processamento e armazenamento superiores, poderemos então abordar as ofertas L ou mesmo XL. Uma máquina com o perfil L possui já 4 CPU Cores, 7GB de RAM e 1TB de armazenamento local, apresentando um custo horário de $0.48, sendo que uma máquina com o perfil XL possui 8 CPU Cores, 15GB de RAM e 2TB de armazenamento, apresentando um custo horário de $0.96.

Como podemos ver pelos dados fornecidos, a capacidade computacional e de armazenamento local é efectivamente progressiva ao longo de todos os segmentos, bem como o custo associado a cada um deles. Se precisamos de mais processamento, o custo será necessariamente maior. No entanto, sendo que a plataforma é totalmente elástica, isso permite-nos fazer uso de todo esse poder de processamento apenas quando tal é necessário, revertendo para máquinas com processamento mais limitado quando tal não se justifique. Tal capacidade permite ter alta disponibilidade quando sabemos que vamos precisar dela e apenas pagar por a mesma durante o período de carência, ao contrário do que aconteceria num data center que tivéssemos de ser nós a dimensionar e a gerir.

Quando fazemos uso de uma máquina no Windows Azure, podemos fazer uso de três perfis de execução: Worker Role, Web Role e VM Role. Uma Worker Role é essencialmente um executável que executa no servidor com as características que nós definirmos, permitindo desta forma implementar os mais variados cenários, como seja a disponibilização de servidores aplicacionais próprios, Apache Tomcat, servidores de base de dados, entre muitas outras coisas.

Uma Web Role goza de todas as potencialidades e capacidades de uma Worker Role, mas encontra-se suportada no/pelo IIS (Internet Information Services), podendo correr em modo Full Trust ou Partial Trust. A VM Role é o perfil adicionado mais recentemente à oferta que nos permite colocar uma imagem de um sistema operativo a executar no Windows Azure. Esta Role, ao contrário das restantes que são completamente geridas pela plataforma, deixa ao nosso cuidado tudo o que tem a ver com a gestão da máquina, desde actualizações ao sistema operativo a garantias de segurança da própria máquina e das instalações/actualizações nela efectuadas, pois somos nós quem controla tudo.

O Windows Azure dispõe de uma componente denominada Fabric que é responsável por toda a gestão de máquinas, falhas e recuperações dos sistemas existentes na plataforma.

Tendo isto em linha de conta, é importante pois perceber qual é então o ciclo de vida das Roles que são geridas por este Fabric, relembrando que nos referimos às Worker Roles e às Web Roles. Para melhor ajudar a compreender todo este ciclo, apresenta-se então graficamente o mesmo:

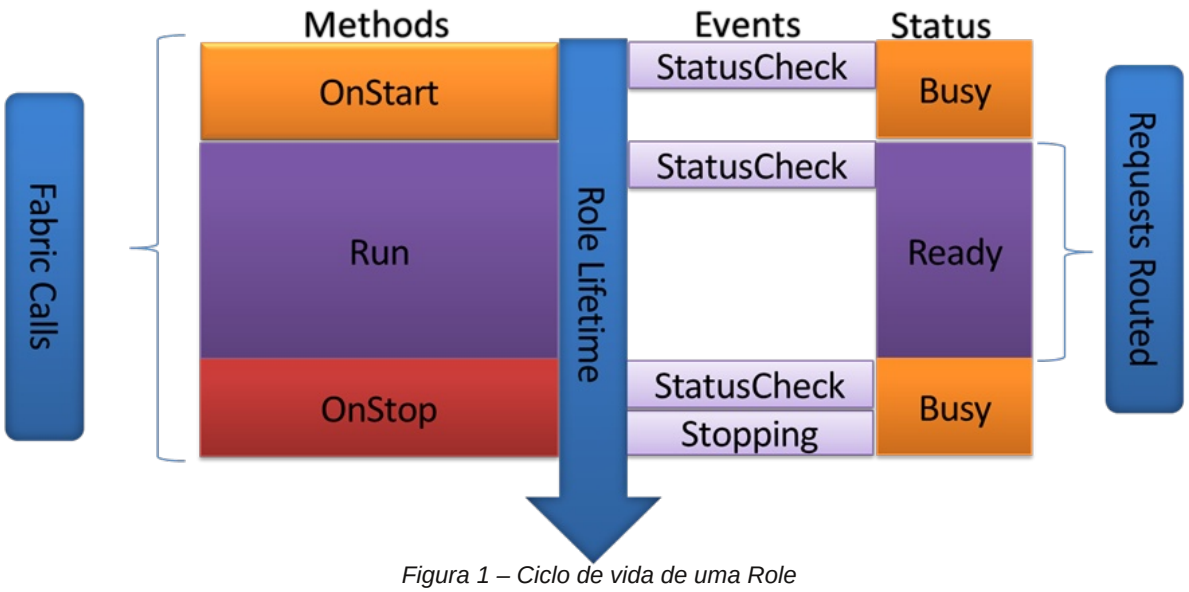



*Figura 1 – Ciclo de vida de uma Role*

Uma Role tipicamente estende o RoleEntryPoint, passando desta forma a poder ser envolvida em todo o ciclo de vida controlado pelo Fabric. Quando o processo WaWorkerHost é iniciado (processo que representa a Role), dá-se o carregamento do assembly associado a essa instância, começando o Fabric por efectuar uma chamada ao método OnStart().

Esta chamada faz com que o estado da instância seja colocado como Busy, o que significa que o Load Balancer do Azure não vai ter essa instância em conta quando estiver a processar pedidos para serem respondidos aplicacionalmente. É neste método que devem ser efectuados todos os processos de automatização do deployment, de forma a garantir que quando a instância for colocada como disponível, todos os sistemas se encontram disponíveis e funcionais.

O método seguinte a ser chamado pelo Fabric é o método Run(). Esta chamada faz com que o estado da instância passe agora para Ready, significando isto que a instância se encontra totalmente operacional e disponível para atender a pedidos, passando desta forma a ter tida em conta pelo Load Balancer quando este estiver a distribuir carga pelas várias instâncias disponíveis. Se o método Run() terminar, o Fabric invoca o método OnStop() da Role. Esta chamada faz com que o estado da instância seja modificado novamente para Busy, sendo removida também da lista de instância disponíveis para o Load Balancer poder utilizar para direccionar os pedidos. É neste método que devem ser efectuados todos os processos de término "gracioso" das aplicações que se encontrem a correr na Role, bem como a salvaguarda de todos os dados que possam ser relevantes armazenar num armazenamento persistente. Este armazenamento persistente pode ser uma base de dados ou, tirando partido do Azure Storage que se encontra incluído no Windows Azure, como Blobs.

Para melhor compreender como efectuar um deployment no Windows Azure, teremos de analisar o que esse processo envolve. Um serviço é então composto por dois artefactos primordiais: um ficheiro de definição do serviço (*.csdef) e um ficheiro de configuração do serviço (*.cscfg). Quando efectuamos a compilação de um projecto para Windows Azure, o Visual Studio pega no código e ficheiros associados e no ficheiro de definição do serviço, embalando tudo num só pacote (*.cspkg), sendo que este ficheiro se encontra encriptado de forma a garantir a segurança da informação nele contida.

De uma forma resumida, podemos ilustrar este processo como:

Encrypted( Zipped( Code + *.csdef ) ) == *.cspkg

Com este pacote disponível (*.cspkg) e com o ficheiro de configuração (*.cscfg), podemos então dar início à criação e disponibilização de um novo serviço no Windows Azure.

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="WebDeploy" xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WebRole name="WebUX">
    <Startup>
      <Task commandLine="..\Startup\EnableWebAdmin.cmd" executionContext="elevated" taskType="simple" />
    </Startup>
    <Imports>
      <Import moduleName="RemoteAccess" />
      <Import moduleName="RemoteForwarder" />
    </Imports>
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpIn" endpointName="HttpIn" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
      <InputEndpoint name="mgmtsvc" protocol="tcp" port="8172" localPort="8172" />
    </Endpoints>
    <ConfigurationSettings>
      <Setting name="DiagnosticsConnectionString" />
    </ConfigurationSettings>
  </WebRole>
</ServiceDefinition>
```

*Figura 2 – Exemplo de um ficheiro de definição de serviço (*.csdef)*

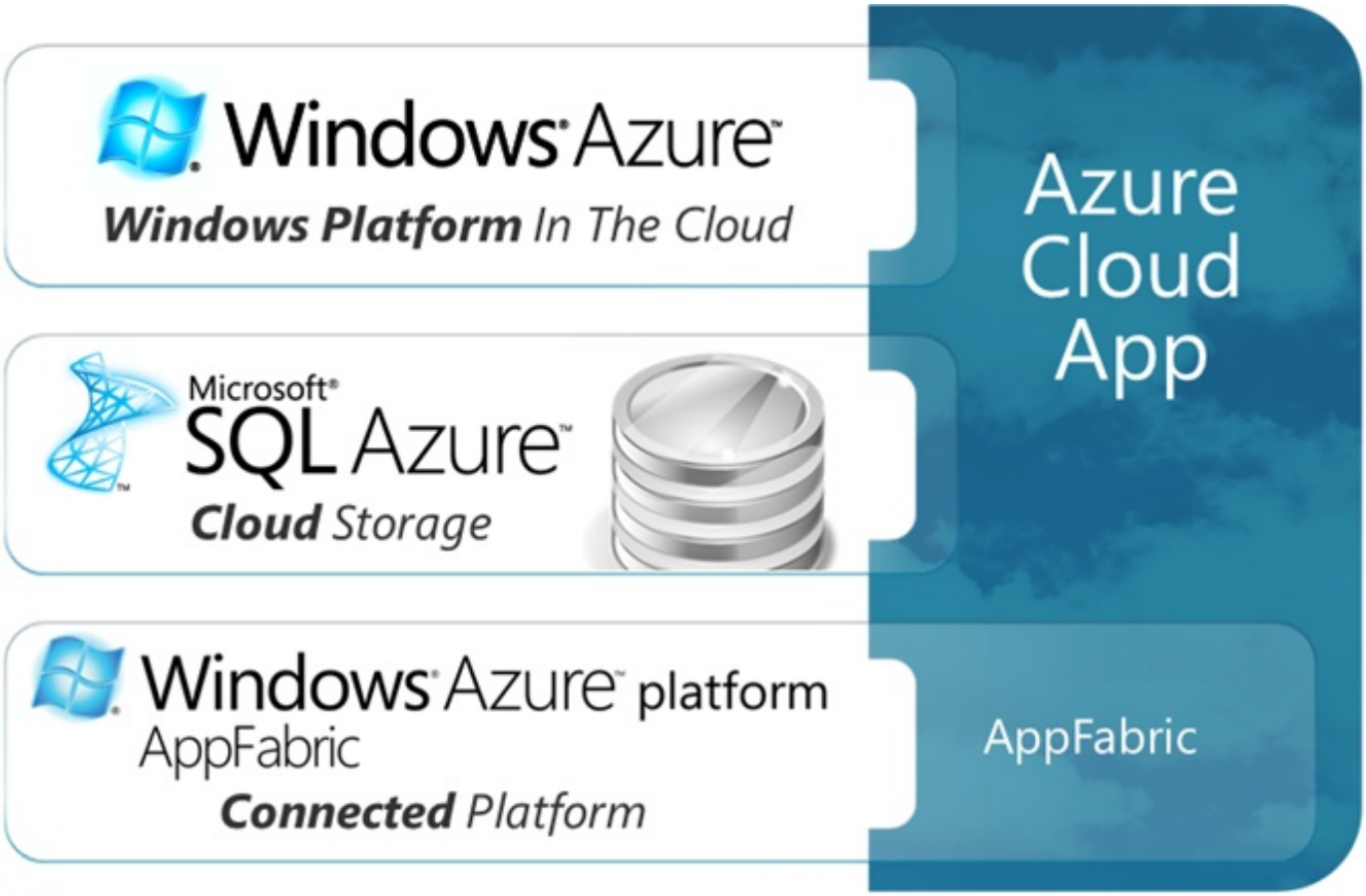
```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="WebDeploy" xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration">
  <Role name="WebUX">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="DiagnosticsConnectionString" value="UseDevelopmentStorage=true" />
      <Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.Enabled" value="True" />
      <Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountUsername" value="dunnry" />
      <Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountEncryptedPassword" value="MIIBrAYJKoZIhvcNAQcDoIIBnTCCAZ
      <Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountExpiration" value="2010-12-23T23:59:59.0000000-07:00" />
      <Setting name="Microsoft.WindowsAzure.Plugins.RemoteForwarder.Enabled" value="True" />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name="Microsoft.WindowsAzure.Plugins.RemoteAccess.PasswordEncryption" thumbprint="D6BE55AC439FAC6CBEBAFF432BD
    </Certificates>
  </Role>
</ServiceConfiguration>
```

*Figura 3 – Exemplo de um ficheiro de configuração de serviço (*.cscfg)*

Tal como mencionado anteriormente, um deployment dito "normal", conterá em si todos os ficheiros necessários à disponibilização de uma dada aplicação/site. Apesar de esta ser a solução mais simples de todas, o seu custo de manutenção pode ser bastante elevado e penoso. Porquê?

Imaginemos que estamos a disponibilizar uma aplicação/sistema que possui inúmeros ficheiros. Cada vez que necessitarmos de efectuar uma modificação, por mais diminuta que ela seja e por menos ficheiros que envolve, teremos sempre de fazer build de um novo pacote de deployment. Esta situação pode não ser particularmente gravosa caso estejamos a falar de deployments de pequena dimensão, mas caso estejamos a falar de pacotes de vários megabytes, a gravidade aumenta consideravelmente, podendo mesmo tornar-se num factor crítico.

Para aumentar a complexidade do caso em questão, imaginemos agora que o sistema recorre a um ficheiro de




configurações internas, ficheiro este que é específico para cada serviço que disponibilizarmos. Neste caso, além de termos de lidar com o problema já mencionado anteriormente, teremos ainda de lidar com a necessidade de ter pacotes de deployments em tudo em tudo semelhantes, onde apenas esse ficheiro de configuração é diferente. Além do crescimento exponencial que se pode verificar ao nível da quantidade de pacotes necessários, há também de ter em conta a dificuldade de manutenção acrescida destes ficheiros.

**Poderemos fazer alguma coisa para minimizar/ultrapassar estas situações? Há alguma solução "mágica"?**

A resposta à primeira pergunta é claramente afirmativa, sendo que não existem silver bullets para resolver este tipo de problemas, apenas abordagens mais correctas ou menos correctas, sendo que devem ser sempre analisadas caso a caso.

Uma das soluções possíveis pode passar pela disponibilização, em Azure Storage, de um arquivo comprimido com todos os ficheiros comuns necessários à aplicação/sistema, bem como de um ficheiro de configuração por cada serviço existente. Desta forma, a Role, durante a chamada do Fabric ao seu método OnStart(), poderia contactar o armazenamento persistente e descarregar e descomprimir os ficheiros comuns, bem como o ficheiro de configuração da aplicação/sistema. Desta forma, apenas seria necessário gerir um ficheiro comprimido partilhado por todos os serviços, minimizando assim a margem de erro e facilitando a sua manutenção.

A solução anterior apresenta ainda um enorme problema no caso de o arquivo partilhado ser consideravelmente grande, pois quando maior for, mais tempo vai demorar a disponibilizar no armazenamento persistente. Esta dificuldade pode ser ultrapassada de uma forma mais ou menos simples através da existência de um mecanismo de actualizações. Ou seja, além do arquivo inicial de que a Role faria uso, seriam também disponibilizados outros

arquivos que permitiriam ir actualizando o sistema de forma incremental e com recurso a arquivos de tamanho consideravelmente inferior. Não só diminuiríamos o tempo necessário à disponibilização de uma actualização, como aumentaríamos a facilidade de manutenção de todo o sistema, pois disponibilizaríamos de um histórico fidedigno.

Há ainda um outro factor que nos deve incomodar e preocupar: a necessidade da existência de um ficheiro de configuração por cada serviço existente. A solução para este problema passaria pela introdução neste ficheiro de wildcards, como seja por exemplo o caso de [database_name]. Esta wildcard permitiria à Role proceder à configuração deste ficheiro, por defeito genérico, com dados específicos de cada serviço. Estes dados poderiam estar configurados no ficheiro de configuração do serviço (*.cscfg) já mencionado anteriormente, ou poderíamos fazer uso de Tables, disponíveis em Azure Storage.

Recorrendo a algumas das técnicas mencionadas, existe um enorme aumento da capacidade de manutenção dos sistemas, bem como da facilidade com que se pode criar e disponibilizar um novo serviço. No limite, para disponibilizar um novo serviço poderá ser apenas necessário um ficheiro de configuração do serviço (*.cscfg) específico e/ou configurar um conjunto de dados existentes numa dada Table existente em Azure Storage.

É também importante relembrar a diminuição da margem de erro e de risco existentes, factores extremamente importantes quando se trata de disponibilizar novos serviços. Finalmente, há ainda a mencionar o facto de se proceder a uma diminuição significativa dos custos envolvidos em todo o processo, não só pela diminuição da largura de banda necessária para disponibilizar um novo serviço, mas também pela redução do espaço necessário em Azure Storage para armazenar toda a informação.

**Páginas úteis:**

Microsoft Windows Azure Homepage
http://www.microsoft.com/windowsazure/

Windows Azure SDK Downloads
http://www.microsoft.com/windowsazure/windowsazuresdk+tools/

Microsoft Windows Azure Team Blog
http://blogs.msdn.com/b/windowsazure/

Artigos Interessantes
http://blogs.msdn.com/b/jnak/archive/2010/02/11/windows-azure-roleentrypoint-method-call-order.aspx
http://msdn.microsoft.com/en-us/library/gg433030.aspx

Azure Storage Explorer + SQL Azure Migration Wizard
http://azurestorageexplorer.codeplex.com/
http://sqlazuremw.codeplex.com/

Link para o artigo: http://tinyurl.com/RPED28-02

## AUTOR

Escrito por **Virgílio Esteves**
É Research Leader do grupo de R&D de uma multi-nacional portuguesa. Nutre um enorme gosto por arquitectura de software, desenvolvimento em WPF, Silverlight e XNA, e cloud computing. Participa activamente na comunidade NetPonto e é autor do blog http://pontonetpt.org/blogs/raposo - Twitter: @vraposo

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar.org