PROGRAMAR

A REVISTA PORTUGUESA DE PROGRAMAÇÃO

Revista nº9 - Julho de 2007

www.portugal-a-programar.org

AJAX & PHP

Saiba como aplicar todo o potencial da tecnologia AJAX ao seu projecto em PHP.

RSS Feed em PHP + MySQL

Saiba como criar o seu próprio agregador de conteúdos

Sistema básico de

Templates em PHP

Como separar o código do conteúdo de maneira simples e eficaz.

e ainda...

A WEB 2.0 EM ANÁLISE

Linux

Compilar e detectar erros em C

índice

equipa PROGRAMAR

contacto
revistaprogramar
@portugal-a-programar.org

website
www.revista-programar.info

A vida é feita de mudança...e surpresas

Foi há pouco mais de um ano que este projecto nasceu, fruto da ideia mas, principalmente, do trabalho de bastantes utilizadores dos quais, embora não sendo sempre os mesmos, ainda hoje dependemos para que a cada dois meses possamos lançar aquela que é a única revista de programação (de carácter generalista) em países de expressão portuguesa.

E porquê em países de expressão portuguesa? Porque é com muito agrado que temos recebido comentários bastante positivos e, inclusive, artigos do Brasil, que nos têm deixado bastante contentes e orgulhosos, eles são verdadeiramente a surpresa anunciada no título. Na edição deste mês figuram dois artigos produzidos por utilizadores brasileiros, um por um utilizador regular do nosso fórum, e outro enviado para o nosso endereço de email. Com isto quero incentivar todos aqueles que nos lêem a que façam o mesmo, pois a revista vive de vós.

Foi com algum custo que esta edição veio à luz do dia. Apesar de todo o bom feedback que temos tido por parte dos utilizadores, ao fim de algumas edições estes esquecem-se de como este projecto deles precisa para vingar. O valor desta revista está mais que provado, mas utilizadores com vontade de escrever e transmitir conhecimentos gratuitamente serão sempre, sempre necessários e bem-vindos.

E qual o porquê da mudança dita no título? É com muita pena minha que o coordenador da revista durante 7 edições, o Sérgio Santos, sai, por passagem do testemunho, do cargo. Eu, Miguel Pais, o antigo coordenador adjunto passo, portanto, a coordenador, com a mesma vontade de sempre de trazer a todos este grande projecto, com a ajuda do Joel Ramos. A mudança é algo natural, e que acontece, a actual equipa da revista já começa a alterar-se e alterar-se-á muito mais, é preciso, portanto, sangue novo. Ele que venha!

Miguel Pais

Google ameaça desactivar Gmail na Alemanha

O Google considera encerrar a versão alemã do Gmail caso o rascunho de uma nova legislação, que propõe banir contas de e-mail anónimas, seja aprovado naquele país.

O projecto de lei exige que os provedores de serviços de Internet e de e-mail recolham e armazenem dados do utilizador, aos quais as autoridades de segurança tenham acesso, caso seja necessário.

De lembrar que recentemente a União Europeia tem investigado o Google sobre a sua política de retenção de dados, tendo este dito estar pronto para reduzir para 18 meses o período de retenção de dados sobre os utilizadores.

ITIJ desafia hackers a solucionar bug no site da presidência da UE

O Instituto das Tecnologias da Informação na Justiça, em conjunto com o Ministério da Justiça, lançou um desafio dirigido a toda a comunidade open source com o objectivo de resolver um bug, que apenas afecta o Firefox, no site da presidência portuguesa da União Europeia. O prémio é de 1000 euros, sendo esta a primeira vez que é lançado um desafia do género por uma instituição portuguesa.

O bug do Firefox impede o reconhecimento do certificado do site da presidência portuguesa da UE no acesso a utilizadores registados. O certificado não é reconhecido como sendo emitido por uma entidade válida, embora funcione noutros browsers, entre os quais o Internet Explorer.

Apple lança o iPhone para o mercado

O dia pelo qual milhões de pessoas esperavam chegou a 29 de Junho: o lançamento do iPhone, o mais recente produto da Apple. Por volta das seis da tarde, a hora a que as 164 lojas Apple e as 2000 lojas da AT&T começaram a comercializar o tão esperado telemóvel, as filas já eram enormes.

Basta lembrar que o iPhone combina as funções de um iPod com as de um telemóvel, permite que os utilizadores acedam à Internet, vejam televisão, tirem fotografias... Tudo num ecrã de 3 polegadas.

O telemóvel foi lançado apenas nos Estados Unidos e apenas está disponível com a operadora AT&T para contractos de assinatura com a duração mínima de 2 anos, não havendo ainda uma data prevista para a sua chegada à Europa.

AJAX & PHP

Actualmente qualquer programador ou até utilizador da web já deve pelo menos ter ouvido falar em Ajax e atenção que não me refiro à marca de detergentes que muitas pessoas usa para lavar, limpar etc... Refiro-me sim a uma tecnologia que de certo modo revolucionou o mundo Web.

Ajax não é uma linguagem de programação mas sim uma técnica para criar melhores aplicações web, mais rápidas e mais interactivas. Embora o nome sugira uma linguagem de programação Ajax não passa de uma sigla Asynchronous JavaScript and XML, ou seja esta técnica permite que JavaScript comunique directamente com o servidor usando o objecto XMLHttpRequest do JavaScript. Assim é possível trocar dados entre o servidor e o cliente sem precisar de recarregar a página, poupando tempo e tráfego.

Ajax é baseado em Javascript, XML, HTML e CSS, estes Web Standards são suportados pela maioria do browsers permitindo assim que as aplicações Ajax sejam independentes de plataforma e browser. Em aplicações sem Ajax cada vez que o cliente dá entrada de um novo input, seja um form, um click button ou até mesmo um link, é feito um pedido ao servidor por GET ou POST, o servidor gera a resposta e envia para o cliente que por sua vez gera uma nova página com os dados enviados pelo servidor.

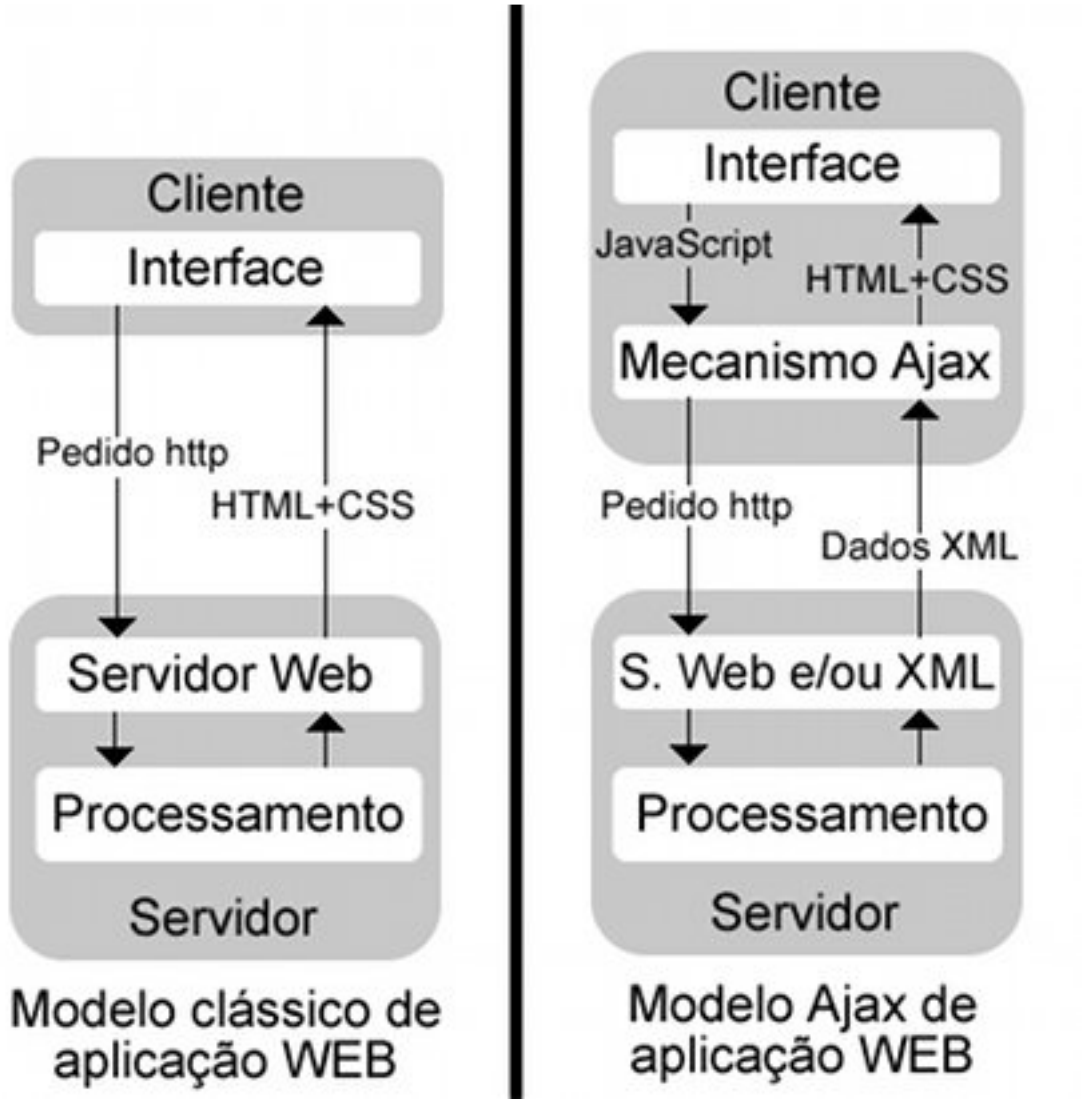
Este processo desperdiça principalmente uma coisa que pode ser bastante importante: TEMPO.

Imaginemos que temos uma página com um tamanho considerável, imagens, textos, menus, etc... e queremos apenas clicar num botão de forma a mudar o texto central deixando toda a restante página na mesma.

Utilizado uma aplicação sem Ajax cada vez que clicarmos para mudar o texto vamos ter de pedir ao servidor toda a página e carregá-la no cliente. Ou seja se a página contiver 300kbytes e o texto que queremos mudar for apenas 50 desses 300kbytes estamos a perder 250kbytes em tempo, quando podíamos perder apenas 50kbytes já que é o tamanho da única modificação que fizemos na página.

Com Ajax é possível pedir ao servidor apenas os dados referentes à modificação que desejamos fazer, deixando o restante conteúdo da página no estado inicial mudando apenas algumas coisas. Para isso usamos o XMLHttpRequest do JavaScript, usando este objecto juntamente com outras características do JavaScript é possível actualizar os dados da página depois desta estar carregada e sem precisar de a recarregar.

Podemos ver nesta imagem uma comparação entre o modelo de transferência de dados com e sem Ajax.



Vamos começar com um exemplo muito simples e útil de Ajax. Vamos ter um form com input de texto e cada vez que uma tecla for pressionada e libertada será executada uma função JavaScript que usa o XMLHttRequest para fazer o pedido de informação ao script PHP. De seguida é colocada a informação na página principal sem ter de a carregar em cada letra colocada no input. Primeiro vamos criar o formulário que pode ser feito num simples ficheiro html.

```
<html>
    <head>
        <script src="ajax.js"></script>
    </head>
    <body>
        <form>
            Texto:<input type="text" id="txt1" onkeyup="showHint(this.value)">
      </form>
      <p>
          Sugestoes: <span id="sug"></span>
      </p>
  </body>
</html>
```

Começamos por importar o ficheiro ajax.js, linha 3 que contém as funções JavaScript para o funcionamento do nosso programa, este ficheiro será criado mais para a frente. Nas linhas 8, 9 e 10 é criado um form com um input de texto que a cada event onkeyup executa a função showHint com o valor actual do input. Agora vamos ver o ficheiro ajax.js

```
 var xmlHttp

 function MakeXmlHttpObject(){

   var xmlHttp=null;

  try{
   // XmlHttpRequest para Firefox,
Opera, Safari e derivados.
     xmlHttp = new XMLHttpRequest();
      }

  catch (e){
  // XmlHttpRequest para Internet
Explorer.

   try{
   // Internet Explorer 6.0+
    xmlHttp = new
ActiveXObject("Msxml2.XMLHTTP");
      }

   catch (e){
   // Internet Explorer.
    xmlHttp = new
ActiveXObject("Microsoft.XMLHTTP");
       }
    }
   return xmlHttp;
 }

  function stateChanged() {
    if (xmlHttp.readyState==4){
document.getElementById("sug").innerHTML
=xmlHttp.responseText;
      }
  }
```

```
function showHint(str){
  if(str.length==0){

document.getElementById("sug").innerHTML
="";
    return;
  }

 xmlHttp=MakeXmlHttpObject()

 if (xmlHttp==null){
      alert ("O browser não suporta
AJAX!");
    return;
 }

 var pedido="pedidos.php";
  pedido=pedido+"?q="+str;


xmlHttp.onreadystatechange=stateChanged;
  xmlHttp.open("GET",pedido,true);
  xmlHttp.send(null);
}
```

Declarámos a variável xmlHttp que é uma variável "global", onde o XmlHttpObject será guardado depois de criado, permitindo ser usado por outras funções JavaScript. Nas linhas 3 a 23 temos a função MakeXmlHttpObject que vai criar o nosso XmlHttpObject, este objecto pode ser criado de formas diferentes dependendo do browser. Para possibilitar a criação deste objectos de forma dinâmica independentemente do browser utilizamos uma cadeia de try e cath. A primeira tentativa como podemos ver na linha 9 é o uso do método "XMLHttpRequest()", este método funciona em browsers como Firefox, Opera, Safari, entre outros; caso este método falhe então vamos tentar usar o método "ActiveXObject("Msxml2.XMLHTTP")", este método funciona em browsers Internet Explorer de versão 6.0 e superiores. Caso também este método falhe, então é usado o método "ActiveXObject ("Microsoft.XMLHTTP")" para browsers Internet Explorer de versão inferior a 6.0.

No caso de todos os métodos falharem então o nosso objecto será null, ou seja o browser usado não suporta Ajax. Após este método já temos a chave de todo o mecanismo Ajax, o XMLHttpRequest Object. Agora passamos a algo bastante importante no funcionamento deste mecanismo, os estados do XMLHttpRequest, ou seja os 5 estados pelo qual o pedido passa antes de chegar ao cliente. Os estados são os seguintes:

Estado 0 - Pedido não iniciado.
Estado 1 - Pedido configurado.
Estado 2 - Pedido enviado.
Estado 3 - Pedido em processo.
Estado 4 - Pedido completo e recebido.

Por norma basta usarmos o Estado 4 que corresponde ao pedido completo. Nas linhas 25 a 29 criamos uma função que vai ser usada mais à frente, para verificar se o estado actual é igual a 4 e caso seja, a resposta enviada pelo servidor será escrita na página principal sem precisar a recarregar.

Nas linhas 31 a 52 temos a função principal que vai usar as funções anteriormente definidas para executar o pedido ao servidor e colocar os dados retornados por este. Começamos por verificar se foi colocado algum carácter no nosso input do form, no caso de não ter sido colocado nenhum valor, então o campo das sugestões será colocado em "branco" (linha 34), no caso de ter sido colocado algum valor no input é executado o método "MakeXmlHttpObject", para criar o XMLHttpRequest Object (linha 38), caso este retorne null então será emitido um alert e termina a função (linhas 40 a 43), caso contrário é construído o pedido que neste caso será para o método GET.

Nas linhas 45 e 46 vemos que o pedido é constituído pelo URL da página à qual o pedido vai ser feito, juntamente com o valor contido no input box; o próximo passo (linha 48) é dizer que a função deve ser executada cada vez que o XMLHttpRequest muda de estado ou seja 0 -> 1, 1-> 2.... Para finalizar vamos usar a função open (linha 50) para definir o método de envio, neste caso "GET", a URL e o terceiro campo corresponde ao modo do pedido. Será assíncrono no caso de true e síncrono no caso de false, vamos colocar true para que não seja preciso esperar pela resposta do servidor, agora basta enviar o pedido ao servidor para isso usamos a função send (linha 51) que envia o pedido ao servidor. Como usamos o método GET a função send deve ter como argumento null.

Vamos ver o ficheiro pedido.php, que recebe o pedido e envia a resposta ao nosso script.

```
<?php

 // Opcionalmente pode ser colocado
este header para não usar a cache
header("Cache-Control: no-cache, must-
revalidate");

// Array com das sugestões possíiveis.
$a[]="A";
$a[]="AA";
$a[]="AAA";
$a[]="AAAA";
$a[]="B";
$a[]="BB";
$a[]="BBB";
$a[]="BBBB";
$a[]="C";
$a[]="CC";
$a[]="CCC";
$a[]="CCCC";
$a[]="D";
$a[]="DD";
$a[]="DDD";
$a[]="DDDD";
$a[]="E";
$a[]="EE";
$a[]="EEE";
$a[]="EEEE";
```

```
// Captura do parâametro q do URL.
$q=$_GET["q"];

// Verifica se o parâmetro enviado tem
comprimento maior que 0.
if (strlen($q) > 0){
 $hint="";
   for($i=0; $i<count($a); $i++){
    if
(strtolower($q)==strtolower(substr($a[$i
],0,strlen($q)))){
     if (empty($hint)){
      $hint=$a[$i];
       } else {
         $hint=$hint." , ".$a[$i];
       }
      }
   }
}

// Coloca na resposta de retorno a
string "Não existem sugestões"
// ou as sugestões possiveis.
if (empty($hint)){
  $resposta="Não existem sugestões.";
} else {
  $resposta=$hint;
}

// Retorno da resposta.
echo $resposta;
?>
```

De uma forma resumida o que o nosso pedido.php faz é receber um pedido sob a forma de uma string, cria um array com palavras e vai criar uma novo string com todas as palavras contidas no array que tem como prefixo a string enviada, caso não exista nenhum palavra no array com prefixo igual à do pedido então é retornada a string "Não existem sugestões." caso contrário é enviada em uma única string com todas as palavras encontradas separadas por uma ",".

O segundo exemplo é algo um pouco mais complexo, vamos utilizar o método de pedido POST ao invés do GET que foi utilizado na exemplo anterior, vamos também ver como podemos colocar uma pequena animação de loading enquanto esperamos pela resposta do servidor esta animação não tem grande utilidade neste exemplo visto não se tratar de um pedido muito complexo, mas é um bom exemplo para usar por exemplo em pesquisas demoradas de bases de dados, carregamentos de ficheiros etc...

```
<html>
<head>
  <script src="ajax2.js"></script>
</head>

<body onload=loaded()>
 <form name="Form1">
  <select name="regiao"
onchange="MostraCidade(this.value)">
    <option value="-1">Selecione uma
Regiao</option>
               <option
value="1">Alentejo</option> <option
value="1">Alentejo</option>
               <option
value="2">Algarve</option>
               <option
value="3">Estremadura e
Ribatejo</option>
               <option value="4">Lisboa
e Setúbal</option>
           </select>
           <select name="cidade"
onchange="Alert(this.value)">
           </select>
       </form>
       <img id="loading"
src="./loading.gif" alt="Loading..."/>

    </body>
</html>
```

Tal como no exemplo anterior começamos por importar o ficheiro JavaScript que contéem todas as funções para a interação, na linha 6 adicionamos uma pequena função que vai ser executada quando o corpo da pagina for carregado, mais à frente vamos ver o que faz esta função. Nas linhas 9 a 15 criamos uma combo box, esta box é também preparada para executar a função MostraCidade com o valor da opção selecionada no momento em que a opção é mudada (linha 9), nas linhas 16 e 17 criamos outra combo box mas vazia, esta box será preenchida com a resposta do servidor irá tal como a box região executar uma função cada vez que a opções selecionada mudar, mas neste caso a função executada será a Alert. Por fim vamos colocar a animação de loading (linha 19), vamos usar uma imagem animada que irá aparecer cada vez que um pedido for feito ao servidor e irá desaparecer sempre que a resposta for recebida. Analisemos agora as diferenças no ficheiro ajax2.js para o anterior ajax.js.

```
var xmlHttp
function MakeXmlHttpObject(){
  var xmlHttp=null;
  try{
     // XmlHttpRequest para Firefox,
Opera, Safari e derivados.
     xmlHttp = new XMLHttpRequest();
 }
 catch (e){
     // XmlHttpRequest para Internet
Explorer.
     try{
         // Internet Explorer 6.0+
         xmlHttp = new
ActiveXObject("Msxml2.XMLHTTP");
     }
     catch (e){
         // Internet Explorer.
         xmlHttp = new
ActiveXObject("Microsoft.XMLHTTP");
     }
 }
 return xmlHttp;
}
```

```
function loaded(){
document.getElementById("loading").style
.visibility = "hidden";
}

function loading(){
document.getElementById("loading").style
.visibility = "visible";
}

function stateChanged() {
    if (xmlHttp.readyState==4){
        loaded();
    document.Form1.cidade.innerHTML =
xmlHttp.responseText;
    }
}

function Post(regiao) {
    var parametros = "r=" +
encodeURI(regiao);
    return parametros;
}

function MostraCidade(regiao){
    loading();
    xmlHttp=MakeXmlHttpObject()
    if (xmlHttp==null){
        alert ("O browser não suporta
AJAX!");
        return;
    }

   var parametros = Post(regiao);

xmlHttp.onreadystatechange=stateChanged;
xmlHttp.open("POST","cidade.php",true);
xmlHttp.setRequestHeader("Content-
type", "application/x-www-form-
urlencoded");
xmlHttp.setRequestHeader("Content-
length", parametros.length);
xmlHttp.send(parametros);
}

function Alert(cidade){
    alert("A cidade selecionada é :
"+cidade);
}
```

A função MakeXmlHttpObject (linha 3 a 23) mantém a estrutura original criando e retornando um XMLHttpRequest Object, as funções loaded (linha 25 a 27) e loading (linha 29 a 31) são usadas para ocultar e mostrar respectivamente a imagem de loading, como podemos ver cada uma das funções muda a visibility da imagem.

A função stateChanged (linha 33 a 38) mantém a estrutura e funcionalidade inicial do exemplo anterior, apenas foi adicionada a função loaded para que quando a resposta do servidor chegar a imagem de loading fique oculta.Vamos agora começar a analisar as funções de pedido por POST. A função POST (linha 40 a 43) recebe os parâmetros a enviar por POST, neste caso apenas um e cria a String de pedido retornando-a. A função principal deste exemplo, MostraCidade (linha 45 a 65), que recebe o valor da combo box Região, começa por colocar a imagem de loading visível (linha 47), o XMLHttpRequest Object é criado e verificado tal como no exemplo anterior (linha 49 a 54), prepara a String de pedido POST (linha 56) e, tal como no primeiro exemplo, definimos qual a função a ser executada a cada mudança de estado (linha 58).

A grande diferença do pedido POST para o GET é que, para começar a função open, ao contrário do primeiro exemplo em que o primeiro argumento desta função era a String "GET", é neste caso a String "POST" (linha 60), assim o pedido será feito pelo método POST, e sendo assim temos de utilizar a função setRequestHeader para enviar alguns headers no pedido. Sem estes headers o pedido POST não é realizado, ao contrário do método GET em que estes não são necessários. na linha 61 temos a primeira header "Content-type". Esta header indica qual o tipo do conteúdo do pedido feito. Em seguida, na linha 62, temos a header "Content-length" que indica o tamanho do corpo do pedido que neste caso é o tamanho da String de pedido. Por fim a função send que, ao contrário do exemplo anterior em que esta função leva como argumento o valor null, no caso actual a função send deve ter como argumento a String de pedido (linha 63). A função Alert (linha 66 a 68) é uma simples função que recebe um valor e abre uma alert formatada com o valor dado.

Vejamos agora o conteúdo do ficheiro cidade.php que uma forma geral não difere muito a nível de funcionamento do ficheiro pedido.php apresentado no exemplo anterior.

```
<?php
$r = $_POST['r'];

sleep(2);
$cidades = "<option>Selecione um
Cidade</option>";

$a[]="Beja";
$a[]="Évora";
$a[]="Portalegre";
$b[]="Algarve";
$b[]="Portimão";
$c[]="Santarém";
$c[]="Leiria";
$d[]="Lisboa";
$d[]="Setubal";

if($r == 1){
    for($i=0; $i<count($a); $i++){
        $cidades .=
"<option>$a[$i]</option>";
    }
}
elseif($r == 2){
    for($i=0; $i<count($b); $i++){
        $cidades .=
"<option>$b[$i]</option>";
    }
}
elseif($r == 3){
    for($i=0; $i<count($c); $i++){
        $cidades .=
"<option>$c[$i]</option>";
    }
}
```

```
elseif($r == 4){
    for($i=0; $i<count($d); $i++){
        $cidades .= "<option>$d[$i]</option>";
    }
}

echo $cidades;

?>
```

Tal como no ficheiro pedido.php começamos por recolher a região enviada, em seguida usamos a função sleep em condições normais esta função não seria usada, a sua função neste caso é a de apenas colocar o script a "dormir" durante uns segundos de forma a que a imagem de loading apareça, visto que neste exemplo sem esta função o tempo de loading será quase nulo. Em seguida são definidos três arrays, cada um destes arrays corresponde a uma região e cada um tem nomes de cidades pertencentes a essa região, para finalizar o processo dependendo do array selecionado são criadas options para colocar na combo box "cidade" e envia sob forma de uma String.
Os exemplos aqui apresentados podem ser visualizados e/ou descarregados em http://www.wmagician.hf-tuga.net/Ajax/
Para complementar o artigo vamos agora ver a "API" do XMLHttpRequest Object. Começamos com por ver a tabela de métodos disponíveis:

Método	Descrição
abort ()	Cancela o pedido actual.
getAllResponseHeaders ()	Retorna todas as headers HTTP sob a forma de uma String.
getResponseHeader(header)	Retorna o valor de uma header HTTP dada.
open(método, URL , async)	Atribui o método, URL e o modo de pedido. Método pode ser GET, POST, HEAD, PUT, DELETE. URL pode ser qualquer endereço absoluto ou relativo. Modo (async) pode ser true para assíncrono ou false para síncrono.
send(conteúdo)	Envia o pedido.
setRequestHeader(nome, valor)	Adiciona o par nome/valor como Header HTTP ao pedido.

Agora a tabela de propriedades do objecto.

Propriedade	Descrição.
onreadystatechange	Contém a referêencia à função a ser executada a cada vez que o estado muda.
readyState	Retorna o estado actual do pedido 0,1,2,3,ou 4.
responseText	Retorna a resposta sob forma de String
responseXML	Retorna a resposta sob forma de XML.
status	Retorna o código HTTP do estado, (404 para "NOT FOUND") ou (200 para "OK")
statusText	Retorna o estado do pedido como String. "OK" ou "NOT FOUND".

E assim termina este artigo. A partir deste momento já tem bases para criar aplicações Ajax - PHP; para a frente é usar a imaginação.

Fábio Correia

Framework CakePHP

O que é o CakePHP?

CakePHP é uma framework open-source grátis para desenvolvimento rápido em PHP.

Porquê CakePHP?

O CakePHP tem diversas características que o tornam a escolha certa como estruturas para aplicações, tornando a criação de aplicações rápida.

Vantagens de utilizar o CakePHP

- Comunidade activa e amigável;
- Flexível para licenciamento de aplicações;
- Compatibilidade com PHP4 e PHP5;
- *Scaffolding* da aplicação;
- CRUD integrado para a interacção da base de dados e perguntas simplificadas;
- Template rápida e flexível;
- Listas do controlo de acesso flexíveis;
- Outras vantagens como Segurança, Ajax, Sessions entre outras.

Requisitos

- Este artigo é baseado na versão 1.1.14.4797 do CakePHP, a última versão estável na data da escrita do artigo. Para fazer download visite http://www.cakephp.org/ .
- Servidor Apache, PHP e MySQL.

Instalação

A primeira coisa a ser feita é verificar a configuração do arquivo httpd.conf do Apache para activar o mod_rewrite. Para activar basta descomentar as seguintes linhas (nalguns casos pode estar já activo, nesse caso deve passar-se ao passo seguinte):

```
LoadModule rewrite_module
modules/mod_rewrite.so
AddModule mod_rewrite.c
```

Encontre também a linha seguinte:

```
This should be changed to whatever
you set DocumentRoot to
```

E acrescente a seguir:

```
AllowOverride All
```

Após estas operações, vamos então descompactar o CakePHP.

Estruturas das directorias

Descompactamos o ficheiro, o qual coloquei da seguinte forma para poder aceder via http.

FIG 1

Repare na imagem acima que apenas coloquei a pasta do CakePHP na minha pasta revistaphp.

A pasta do CakePHP é composta pelos ficheiros e pastas abaixo:

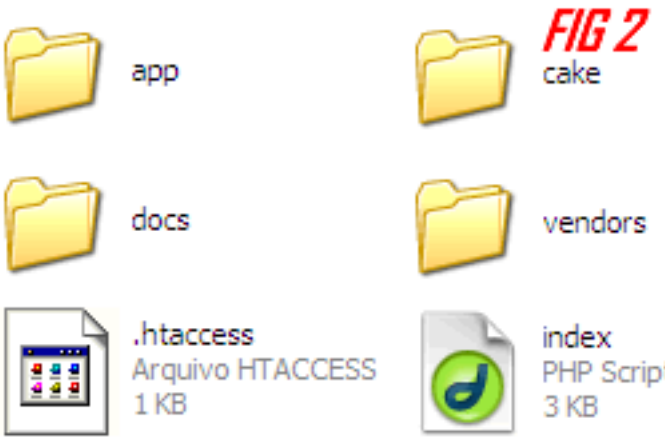


FIG 2

- Pasta app - A nossa aplicação
- Pasta cake - É o cérebro, onde temos as bibliotecas, configs, libs. Abaixo, vemos o conteúdo da pasta cake.

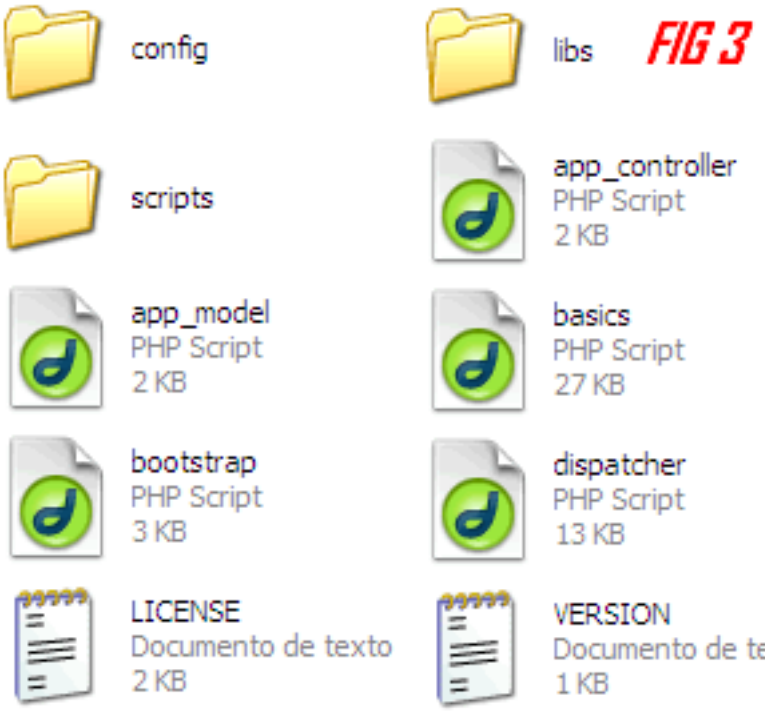


FIG 3

Agora vamos testar a nossa aplicação.

Resultado:

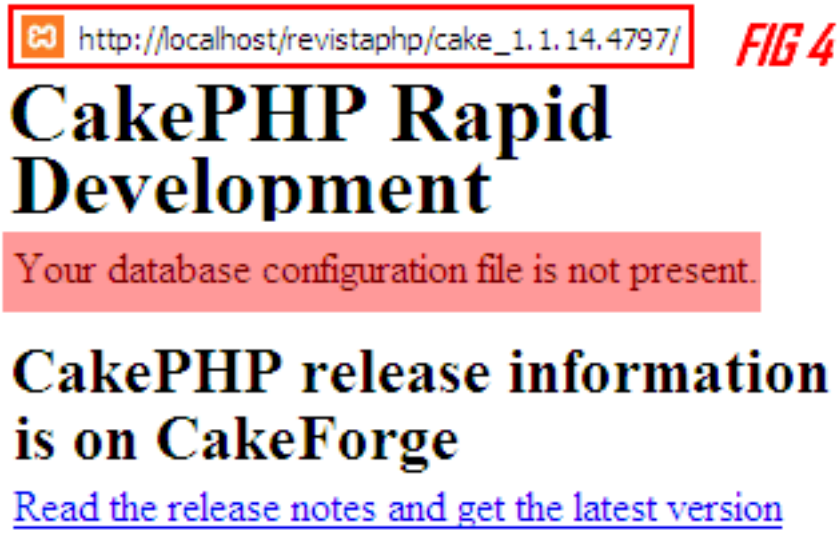


FIG 4

Repare que a mensagem em vermelho diz que não temos o arquivo de configuração da base de dados. Para resolver isso, vamos criar a nossa base de dados.

Criando a base de dados

```
CREATE TABLE posts (
id INT UNSIGNED AUTO_INCREMENT
PRIMARY KEY,
title VARCHAR(50),
body TEXT,
created DATETIME DEFAULT NULL,
modified DATETIME DEFAULT NULL
);
```

Preenchendo a base de dados:

```
INSERT INTO posts
(title,body,created)
VALUES ('The title', 'This is the
post body.', NOW());
INSERT INTO posts
(title,body,created)
VALUES ('A title once again', 'And
the post body follows.', NOW());
INSERT INTO posts (title,body,created)
VALUES ('Title strikes back', 'This is
really exciting! Not.', NOW());
```

Configurando conexão ao MySQL

O ficheiro de configuração da base de dados do CakePHP encontra-se em /app/config/database.php.default. Faça uma cópia deste ficheiro na mesma directoria e vamos renomear a cópia para database.php. Repare que se fizer refresh no browser, a mensagem da fig. 4 já terra mudado para "*Your database configuration file is present.*"
Abrimos o ficheiro database.php e configuramos a conexão conforme abaixo:

```
class DATABASE_CONFIG
{
var $default = array('driver' =>
'mysql',
'connect' => 'mysql_connect',
'host' => 'localhost',
'login' => 'cakeBlog',
'password' => 'c4k3-rUl3Z',
'database' => 'cake_blog_tutorial',
'prefix' => '');
var $test = array('driver' =>
'mysql',
'connect' => 'mysql_connect',
'host' => 'localhost',
'login' => 'user',
'password' => 'password',
'database' => 'project_name-test',
'prefix' => '');
}
```

Trabalhando com o MVC

O conceito MVC (Model View Controller), ou seja, em tradução literária: Controlador do Modelo de Visão. Basicamente relacionam-se os conceitos de Controlo, Modelo e Visão.

Visão: É a camada de visualização da sua aplicação, onde apenas se apresenta o que foi obtido através do controlo. Eu diria que é o que chega ao utilizador final, a parte visual, a interface. A visão não deve ter nenhuma lógica de código, apenas a exibição dos dados.

Controlo: É onde serão processadas todas as requisições feitas através da interface (visão). O controlo também acede ao modelo com o fim de obter determinadas informações. Toda a lógica da aplicação (validações, atribuições, etc) é feita no controlo. Eu diria que o controlo é o gestor da aplicação.

Modelo: É o modelo da sua aplicação, onde são definidos propriedades e atributos das suas personagens. Na maioria dos casos existe ainda uma camada de persistência dos dados, mas excepcionalmente neste artigo a nossa persistência será feita no modelo.

Desenvolvendo a nossa aplicação

Vamos agora criar os nossos ficheiros. Começamos por criar um ficheiro chamado post.php que será o nosso modelo.

```
<?php
class Post extends AppModel
{
var $name = 'Post';
}
?>
```

As pastas devem estar correctas e vamos salvar o nosso ficheiro em /app/models, ficando desta forma /app/models/post.php.

Controllers

Vamos agora criar o arquivo chamado posts_controller.php que será o nosso controlo que deverá estar na pasta /app/controllers ficando assim /app/controllers/posts_controller.php.

```
<?php
class PostsController extends
AppController
{
  var $name = 'Posts';

  function index(){
    $this->set('posts',$this-> Post-
>findAll());
    }

  function view($id) {
    $this->Post->id = $id;
    $this->set('post', $this-> Post-
>read());
   }

  function add() {
     if (!empty($this->data)) {
       if ($this->Post-> save($this-
>data)) {
          $this->flash('Your post
has been saved.','/posts');
       }
      }
    }

  function edit($id = null) {
    if (empty($this->data)){
      $this->Post->id = $id;
      $this->data = $this->Post-
>read();
    } else {
      if ($this->Post-> save($this-
>data['Post'])) {
        $this-> flash('Your post
has been updated.','/posts');
      }
    }
  }

  function delete($id) {
    $this->Post->del($id);
    $this->flash('The post with id:
'.$id.' has been deleted.',
'/posts');
  }
}
?>
```

Views

Vamos agora criar uma pasta chamada posts dentro da pasta view, ficando desta forma /app/views/posts. Dentro da pasta posts, vamos criar três ficheiros .thtml.

Ficheiro index.thtml:

```
<h1>Blog posts</h1>
<p><?php echo $html->link("Add Post",
"/posts/add"); ?>
<table>
  <tr>
   <th>Id</th>
   <th>Title</th>
   <th>Created</th>
   </tr>
  <?php foreach ($posts as $post): ?>
  <tr>
   <td><?php echo
$post['Post']['id']; ?></td>
    <td>
    <?php echo $html->
link($post['Post']['title'],
'/posts/view/'.$post['Post']['id']);
    echo $html->
link('Delete',"/posts/delete/{$post['P
ost']['id']}",null,'Are you sure?');
    echo $html->link('Edit',
'/posts/edit/'.$post['Post']['id']);?>
     </td>
   </td>
    <td><?php echo
$post['Post']['created']; ?></td>
  </tr>
  <?php endforeach; ?>
</table>
```

Ficheiro view.thtml:

```
<h1><?php echo
$post['Post']['title']?></h1>
<p><small>Created: <?php echo
$post['Post']['created']?></small></p>
<p><?php echo
$post['Post']['body']?></p>
```

Ficheiro add.thml:

```
<h1>Add Post</h1>
<form method="post" action="<?php
echo $html->url('/posts/add')?>">
  <p> Title:
    <?php echo $html->
input('Post/title', array('size'
=> '40'));
    echo $html->
tagErrorMsg('Post/title', 'Title
is required.'); ?>
  </p>
   <p> Body:
     <?php echo $html->
textarea('Post/body',
array('rows'=>'10'));
     echo $html->
tagErrorMsg('Post/body', 'Body is
required.'); ?>
   </p>
    <p>
     <?php echo $html->
submit('Save') ?>
    </p>
</form>
```

Ficheiro edit.thtml:

```
<h1>Edit Post</h1>
<form method="post" action="<?php
echo $html->url('/posts/edit')?>">
  <?php echo $html->
hidden('Post/id'); ?>
   <p> Title:
     <?php echo $html->
input('Post/title', array('size' =>
'40'));
     echo errorMsg('Post/title',
'Title is required.'); ?>
    </p>
    <p> Body:
     <?php echo $html->
textarea('Post/body',
array('rows'=>'10'));
     echo $html->
tagErrorMsg('Post/body', 'Body is
required.'); ?>
    </p>
    <p>
   <?php echo $html->submit('Save') ?>
    </p>
</form>
```

Testando a nossa aplicação

Repare no URL http://localhost/revistaphp/cake_1.1.14.4797/posts e nas opções em azul e Laranja.

Clicando na opção Add Post temos o seguinte:

Repare que não fizemos qualquer select, o CakePHP fez tudo por nós, isso é valido para incluir, editar, apagar, entre muitas outras opções. Podemos adicionar um novo post ao nosso blog.

Se voltar-mos à página onde são apresentados os posts, veremos que apareceu um novo post na lista com a sua data de criação, o qual podemos editar, assim como todas as outras entradas inseridas anteriormente.

Esta é apenas uma das facilidades de trabalhar com o CakePHP, podemos ainda incluir um padrão visual a nosso próprio critério.

Não só é possível fazer isto, mas muitas outras aplicações num curto espaço de tempo e com elevada qualidade. Basta ter imaginação e usar esta framework criada para isso mesmo: facilitar a vida ao programador.

Referências

http://www.cakephp.org
http://bakery.cakephp.org/
http://manual.cakephp.org/
http://en.wikipedia.org/wiki/CakePHP
http://cakephp.blogspot.com/

Carlos Silva

Sistema básico de templates em PHP

Como em qualquer site dinâmico, existe o desenho e a disposição de todos os elementos que o compõem e o conteúdo que irá preencher o site com algo de útil para os utilizadores que o visitam. Mas algo que muitos programadores se esquecem quando desenvolvem um site dinâmico, é separação do código entre ambos. Com este artigo, pretendo mostrar como podemos desenvolver uma pequena classe que nos irá permitir separar de forma simples e básica o conteúdo de todos os outros elementos do site. Desenvolvendo assim, o código torna-se muito mais atraente e bem estruturado. Tudo isto recorrendo à linguagem de programação PHP.

Não faz parte deste artigo explicar detalhadamente cada linha e/ou acção apresentada no código, mas sim mostrar ao leitor como construir um simples e básico sistema de templates para os seus sites dinâmicos. Dito isto, espera-se que o leitor já possua conhecimentos básicos/moderados de PHP, nomeadamente em classes. No entanto, grande parte do código encontra-se comentado; apesar de ser em inglês, penso que poderá servir de ajuda a alguém.

Uma pequena nota antes de começarmos. Este tutorial é baseado num que vi há muitos anos atrás e, devido a esse distanciamento temporal, não posso precisar o seu autor nem facultar link para o artigo original. No entanto, aquilo que permanece do original é pouco mais que a ideia geral do artigo, já que o código sofreu bastantes modificações ao longo dos anos derivadas de todo o uso

que lhe dei para satisfazer as minhas necessidades que, penso, irão satisfazer também as vossas. De qualquer forma, o principio deste Sistema Básico de Templates mantém-se o mesmo e os créditos vão para o seu autor original que lamento não poder precisar.

Vamos então começar...

Estrutura da class

O nosso Sistema Básico de Templates (nome que decidi atribuir a esta classe) vai ser composto por apenas 3 funções e 3 variáveis comuns por toda a classe. Mais à frente irei explicar a utilidade de cada uma das funções e de cada uma das variáveis; para já, deixo o seguinte código que irá representar o "esqueleto" da nossa classe:

```
class BasicTemplateSystem {
    var $template_html = array();
    var $template_vars = array();
    var $template_path;

    function
TemplateLoad($template) {

    }

    function
TemplateDefine($references) {

    }

    function
TemplateExport($template, $getonly
= false) {

    }
}
```

Começando pelo mais fácil, a variável *$template_path* irá apenas conter a localização no sistema operativo onde residem os ficheiros que fazem parte do nosso template. Certifique-se que o caminho aponta correctamente para a localização dos ficheiros ou o Sistema Básico de Templates não irá funcionar correctamente.

Seguidamente, temos as variáveis *$template_html* e *$template_vars*. A primeira vai armazenar, sem quaisquer alterações, todo o código html que a função *TemplateLoad()* irá posteriormente obter dos mais diversos ficheiros. Já a segunda, irá armazenar as referências por nós definidas e o conteúdo pelo qual essas referências devem ser substituidas no código html armazenado na primeira variável.

Passando agora para as funções, a *TemplateLoad()*, como o próprio nome indica, serve para carregar os nossos templates. Esta função recebe apenas um parâmetro que indica o nome de cada ficheiro a carregar bem como o nome pelo qual desejamos chamar o template. A função está construída de forma a que seja indiferente a passagem de apenas um ou de vários templates a carregar.

No caso da função *TemplateDefine()*, o nome já não significa tanto como na função anterior, mas isso não quer dizer nada porque é uma função ainda mais simples. A finalidade desta função é definir várias referências que irão existir nos nossos templates bem como o conteúdo que irá substituir essas referências. Mas antes que estas referências sejam definidas, a função irá limpar todas as referências anteriormente criadas; as consequências desta acção irão ser explicadas posteriormente.

Por último, a função *TemplateExport()*, serve para processar todos os dados recolhidos nas funções anteriores. Ou seja, usando as referências lidas com o uso da função *TemplateDefine()*, procura por essas mesmas referências no código html carregado na função *TemplateLoad()* e substitui-as pelo conteúdo também obtido através da função *TemplateDefine()*. O código html processado com novo conteúdo irá ser devolvido pela função *TemplateExport()* após a substituição das referências pelo respectivo conteúdo.

No entanto, e como é bem possível que determinado template não necessite de ter alguma referência a ser substituída por conteúdo, esta função possui 2 parâmetros. O primeiro diz-nos qual o template que desejamos processar e o segundo, caso seja Verdadeiro - pois o seu valor predefinido é Falso - indica que apenas queremos que o código html seja devolvido sem necessitarmos de procurar por referências a substituir por conteúdo.

Depois desta explicação básica das funcionalidades de cada função e de cada variável, segue-se a listagem de todo o código que compõe a classe, juntamente com alguns comentários ao mesmo:

```
<?php

class BasicTemplateSystem {
    var $template_html = array();
    var $template_vars = array();
    var $template_path;
    //---------------------------
    // Load template html files
into variables
    //---------------------------
    function
TemplateLoad($template) {

      // Goes through every
template and loads it
      foreach ($template as $index
=> $value) {

     // Just load the template or
check if we should load it?
         if (is_array($value)) {
             foreach ($value as
$file => $load) {
                 // Should we load
this template?
                 if ($load) {
                     $this->
template_html[$index] =
file_get_contents($this->
template_path.$file);
                 }
             }
         } else {
             $this->
template_html[$index] =
file_get_contents($this-
>template_path.$value);
             }
        }
    }
```

```
//-----------------------------------
// Define template content for some
references
//-----------------------------------

     funnction
TemplateDefine($references) {

     // Clears the current references
and their content

     $this->template_vars = array();

    // Saves the content with it's
reference associated

     foreach ($references as
$reference => $content) {
         $this->
template_vars['{'.$reference.'}'] =
$content;
                         }
                }

//-----------------------------------
// Replace references in template
with saved content
//-----------------------------------

   function TemplateExport($template,
$getonly = false) {

       // Should we parse the
tempalte references or just get the
html?

       if (!$getonly) {
            // Split the html code
into lines

           $html_code = $this-
>template_html[$template];

           $html_code = explode("\n",
$html_code);
           $line_count =
count($html_code);
```

```
// Replace only the lines needed

       for ($i = 0; $i <
$line_count; $i++) {
           foreach ($this->
template_vars as $reference =>
$content) {
               if
(strstr($html_code[$i],
$reference)) {

$html_code[$i] =
str_replace($reference, $content,
$html_code[$i]);
               }
           }
       }

// Join the html code lines and
return the final code

       $html_code = implode("\n",
$html_code);
              } else {
       $html_code = $this-
>template_html[$template];
       }

// Return our template html code

       return $html_code;
    }
}

?>
```

O próximo passo será copiar todo este código para um ficheiro, ie: 'template.php'.

Conteúdo com referências

Presumindo que efectuou o passo anterior, gravando o código para o ficheiro 'template.php', peço agora que guarde o seguinte código html num ficheiro com o nome 'exemplo.html'. Note que o código que se segue não é XHTML válido pela W3C:

```
<html>
<head>
    <title>{webpage.title}</title>
</head>
<body bgcolor="{background.color}">
<span style="color:
{spantext.color}">Sistema Básico de
Templates</span>
</body>
</html>
```

Como podem facilmente reparar, esta simples página não possui um verdadeiro título, uma cor de fundo e uma cor de texto correctas. No seu lugar, usamos várias referências que identificam o que lá deveria estar. As referências podem ser qualquer tipo de texto, com ou sem pontos no meio das palavras como no meu exemplo, todas em letras minúsculas ou maiúsculas, tanto faz. Apenas existe uma regra que deve ser cumprida, cada referência tem de começar por '{' e acabar com '}'.

NOTA: É possível ainda repetir quantas vezes quiser a mesma referência, mas todas as ocorrências dessa referência irão ser substituidas pelo mesmo conteúdo, não sendo possível ter duas (ou mais) referências com o mesmo nome e substituí-las por conteúdos diferentes.

Utilização da class

Como em qualquer outra aplicação Web, o primeira passo a fazer é incluir o código da classe. Seguidamente, e usando a cláusula New, criamos um novo objecto que irá englobar todas as variáveis e funções que a classe possui. Exemplo:

```
include('template.php');
$BTS = New BasicTemplateSystem();
```

Para usarmos correctamente este nosso Sistema Básico de Templates, o primeiro passo a dar é carregar todos (ou grande parte) dos templates que o nosso código irá usar.

A função *TemplateLoad()* pode ser chamada quantas vezes for necessária para carregar um ou vários templates em várias partes do código consoante as nossas necessidades, sendo ainda possível carregar mais de um template na mesma instrução. Esta função recebe um vector como parâmetro que poderá ter um ou mais elementos; este vector possui ainda uma pequena particularidade, é um vector multidimensional. A sua chave irá identificar o template carregado para mais tarde ser processado e o valor associado à chave é o nome do ficheiro que será carregado. Exemplo:

```
$BTS->TemplateLoad(array('exp' =>
"exemplo.html"));
```

NOTA: Se usar a mesma chave para identificar vários templates, tenha em atenção que só o último template a ser carregado irá ser identificado por essa chave. Todos os outros templates carregados anteriormente irão ser destruídos.

A função *TemplateLoad()* possui ainda uma outra funcionalidade, a possibilidade de carregar ou não um template segundo o valor de um outro argumento. Se desejar usar esta funcionalidade para algum template terá de substituir o nome do template a carregar por um outro vector multidimensional com apenas um elemento, onde a chave deste novo vector deverá ser o nome do template a carregar e o valor correspondente - um valor booleano que irá indicar se o template deve ou não ser carregado. Exemplo:

```
$BTS->TemplateLoad(array(
    'abc' => array('abc.html' =>
true),
    'exp' => "exemplo.html",
    'new' => array('new.html' =>
false)
);
```

O código acima apenas vai carregar os templates 'abc.html' e 'exemplo.html', ignorando o 'new.html' pois o valor booleano que indica se devemos ou não carregar o template é Falso. Este exemplo demonstra ainda como seria possível carregar mais de um template na mesma instrução.

Agora que temos os nossos templates carregados, está na altura de decidirmos quais as referências que queremos ver substituídas por determinado conteúdo. Seguindo uma sintaxe parecida com a anterior na função *TemplateLoad()*, a função TemplateDefine() é invocada da seguinte forma:

```
$BTS->TemplateDefine(array(
    'webpage.title'    => "Título da
Minha Página",
    'background.color' => "#660000",
    'spantext.color'   => "#FFFFFF")
);
```

Como se vê facilmente, esta função atribui determinado conteúdo - neste caso, apenas o título da página e duas cores, uma para o fundo e outro para o texto - às respectivas referências. Esta função não faz ainda qualquer tipo de substituição das referências pelo conteúdo, apenas define que referências irão ser substituídas por qual conteúdo. O parâmetro passado para a função *TemplateDefine()* é também um vector multidimensional e continua a ser possível passar apenas um elemento ou vários.

Como em qualquer função existente nesta classe, é possível chamar a função *TemplateDefine()* mais que uma vez e com os elementos que quisermos. No entanto e, como foi dito na secção "Estrutura da Class" no início do artigo, esta acção tem consequências; isto significa que de cada vez que a função é invocada, as referências anteriores irão ser destruídas. Ou seja, as referências só serão realmente substituídas quando invocarmos a função

TemplateExport() e se tivéssemos atribuído o nosso conteúdo às referências assim:

```
$BTS->TemplateDefine(array(
    'webpage.title'    => "Título
da Minha Página")
);

$BTS->TemplateDefine(array(
    'background.color' => "#660000",
    'spantext.color'   => "#FFFFFF")
);
```

Quando a função *TemplateExport()* fosse invocada, apenas as referências {background.color} e {spantext.color} seriam substituídas pelo conteúdo a elas atribuído. A referência {webpage.title} e o seu conteúdo deixam de existir quando invocamos pela segunda vez a função TemplateDefine().

Durante o meu uso deste meu Sistema Básico de Templates, deparei-me com o facto de que muitas das referências que estavam definidas ao longo de todo o código e com conteúdo atribuído apenas foram utilizadas uma única vez no início do código e, dessa forma, estavam a desperdiçar recursos do sistema e a guardar conteúdo desnecessário, pois nunca mais o iria utilizar. Mas não se preocupem que esta medida apenas está aqui para vos ajudar a manter as vossas aplicações optimizadas e em nada irá dificultar o processo de utilização da classe.

Já carregámos os templates necessários e já definimos as referências e o conteúdo pelo qual desejamos que essas referências sejam substituídas. Só nos resta fazer a substituição das referências pelo conteúdo e ver o resultado. É exactamente isso que a função TemplateExport() faz e esta é invocada da seguinte forma:

```
echo $BTS->TemplateExport('exp');
```

Esta função passa um argumento e devolve outro, daí o "echo" antes da invocação da função. O "echo" poderia ter sido facilmente substituído por uma variável seguido de um '=', permitindo assim armazenar o nosso template com as referências substituídas pelo respectivo conteúdo para posteriormente imprimirmos o resultado no ecrã. O único parâmetro passado à função é a chave que usamos para identificar o template carregado na função *TemplateLoad()*. Ou seja, este parâmetro indica qual o template que vamos usar para procurar por uma ou várias ocorrências das últimas referências definidas e substituí-las pelo respectivo conteúdo. Após a substituição de tudo o que for encontrado, a função *TemplateExport()* devolve o template html processado.

Posto isto, o código final do nosso exemplo seria o seguinte:

```
<?php

include('template.php');

$BTS = New BasicTemplateSystem();

$BTS->TemplateLoad(array('exp' =>
"exemplo.html"));

$BTS->TemplateDefine(array(
    'webpage.title'    => "Título da
Minha Página",
    'background.color' => "#660000",
    'spantext.color'   => "#FFFFFF")
);

echo $BTS->TemplateExport('exp');
?>
```

Copiem o código e guardem num ficheiro, por exemplo, 'bts.php', juntamente com os ficheiros anteriores, 'template.php' e 'exemplo.html', num servidor HTTP com suporte para PHP e executem-no para testar esta simples classe para gerir um Sistema Básico de Templates.

Exemplo mais avançado

Para finalizar este artigo, apresento-vos um exemplo ligeiramente mais avançado de como usar esta classe mas nada de muito complicado (porque estou sem ideias práticas):

index.html:

```
<html>
 <head>
  <title>Exemplo + Avançado</title>
 </head>
 <body>
  <ul>
   {list.items}
  </ul>
 </body>
</html>
```

li.html:

```
<li>{item.name}</li>
```

teste.php:

```
<?php

include('template.php');
$BTS = New BasicTemplateSystem();
$BTS->TemplateLoad(array(
    'idx' => "idx.html",
    'li'  => "li.html")
);
for($i = 0; $i < 5; $i++) {
    $BTS->TemplateDefine(array(
        'item.name' => "Numero
".rand())
    );
    if($i == 0) $items_list =
"{$BTS->TemplateExport('li')}\n";
    elseif($i < 4) $items_list .=
"{$BTS->TemplateExport('li')}\n";
    else $items_list .= $BTS-
>TemplateExport('li');
}
$BTS->TemplateDefine(array(
    'list.items' => $items_list)
);

echo $BTS->TemplateExport('idx');

?>
```

Guardem o conteúdo de cada um destes blocos de código num ficheiro individual utilizando o nome indicado por cima de cada bloco juntamente com o ficheiro 'template.php' e executem o ficheiro 'teste.php' para ver o resultado final.

Conclusão

Esta classe não pretende ser algo tão poderoso como o Smarty mas faz o trabalho para o qual foi designado de forma simples e eficaz. No entanto, com o uso desta, o código das vossas aplicações Web vai ficar muito mais limpo e eficaz. Apenas a vossa imaginação e capacidades na programação em PHP irá limitar o que podem fazer, além de terem sempre a possibilidade de modificar a classe e adaptá-la às vossas necessidades.

Como em qualquer programa, existem falhas e/ou inconvenientes que mais tarde ou mais cedo poderão descobrir, conforme o uso que derem à classe, que talvez venham (ou não) a ser resolvidos numa futura versão.

Desde já agradeço a vossa leitura até ao fim e, sinceramente, espero que esta classe vos seja tão útil como a mim foi, pois já não consigo fazer uma aplicação Web sem ela.

Ricardo Amaral

Tecnologia de Identificação por Frequências Rádio

Este artigo explica o funcionamento da tecnologia de identificação por rádio frequência e os avanços feitos nesta área de pesquisa, juntamente com as suas amplas possibilidades de uso. A Radio Frequency Identification é um termo genérico para tecnologias que usam ondas de rádio para identificar artigos individuais automaticamente.

O uso mais comum é registar um número de série num micro-chip que se encontra ligado a uma antena (o chip e a antena são chamados de RFID transponder ou de etiqueta RFID). A antena permite que o chip transmita a informação de identificação para um leitor. O leitor converte as ondas de rádio devolvidas da etiqueta de RFID de forma a que possam ser entendidas por computadores.

Palavras-Chave: Identificação por Rádio Frequência, Etiquetas Inteligentes, Tranponder.

Radio Frequency Identification

A Radio Frequency Identification (RFID), identificação por rádio frequência, é um método automático de identificação de objectos. Pode ser pensada como códigos de barra de nova geração que operam a distâncias bastante grandes comparativamente ao o seu antecessor.
A tecnologia RFID inclui o uso do transponder de RFID com a função de armazenar os dados. Este é denominado de etiqueta RFID ou tag.

Com o uso das etiquetas de RFID é possível adquirir informações como o número de série, validade, fabricante e demais dados do produto, conforme as necessidades da entidade.

De certa forma o RFID revolucionou o sistema de identificação, visto que possui algumas vantagens como leitura sem contacto físico e a possibilidade de identificar vários objectos simultaneamente. As etiquetas também podem ser embutidas nos objectos, sendo flexível ao produto que vai identificar. Estas podem possuir várias formas e tamanhos.

De acordo com a Ibope (2002), uma tecnologia de identificação que pode ser considerada recente são os Smart Card, também chamados de cartões inteligentes.
A diferença entre este cartão e os demais é que ele possui um micro-chip embutido capaz de armazenar, processar e trocar informações com outros sistemas como caixas de supermercado, etc.
Estes cartões dividem-se em dois grupos, os cartões com contacto e os sem contacto. Os cartões "sem contacto" utilizam a tecnologia Identificação por Radio Frequência (RFID) e a sua capacidade de armazenamento pode ir desde alguns caracteres até várias páginas de informações (Rfid Journal, 2006).

O uso de RFID tem aumentado de forma exponencial ao longo dos últimos anos, apesar de ser uma tecnologia que provém da altura da segunda guerra mundial. Um excelente exemplo desta tecnologia em Portugal é a Via Verde, que permite aos condutores passarem as portagens sem terem de parar para pagar, sendo que é lida a tag RFID quando o veículo passa por um reader estrategicamente posicionado.

A Tecnologia RFID

RFID é um termo genérico utilizado em tecnologias que usam ondas de rádio para identificar objectos automaticamente e em tempo real. Um sistema de RFID completo inclui:

- Micro-chip (transponder);
- Antena;
- Leitor (transceptor).

O micro-chip do sistema RFID é a etiqueta ou transponder. É composto pelo micro-chip preso a uma antena RFID. Nele fica armazenado o número de identificação ou outras informações. A antena permite à etiqueta transmitir informações para um leitor. O leitor converte as ondas de rádio de forma a que um computador possa processá-las. No fundo, conceito de RFID é o mesmo que do código de barras.

O Leitor

O leitor comunica com a etiqueta de RFID por ondas de rádio e passa as informações em formato digital para um sistema de computador. De acordo com TELECO (2005) a antena encaracolada do leitor cria um campo magnético com a antena encaracolada da etiqueta. A etiqueta retorna ondas para o leitor. Essas ondas são transformadas pelo leitor em dados digitais, o código electrónico do produto. Na figura 1 visualiza-se um exemplo de leitor RFID em barras.

FIG. 1

A Etiqueta

A etiqueta consiste num micro-chip preso a uma antena que responde ao leitor quando este emite sinais na forma de ondas de rádio, criando o campo magnético. Na etiqueta são armazenadas as informações para identificação do produto. Podendo conter um código ou até mais informações sobre o produto. Encontram várias formas de etiquetas RFID. Há vários tipos: adesivos, botões, cartões e encapsulados para implantação subcutânea.
Elas podem ser classificadas como etiquetas passivas, etiquetas activas e semi-passivas. As primeiras não possuem energia própria retirando energia do campo magnético. As activas retiram energia geralmente de uma bateria para transmitirem informações.

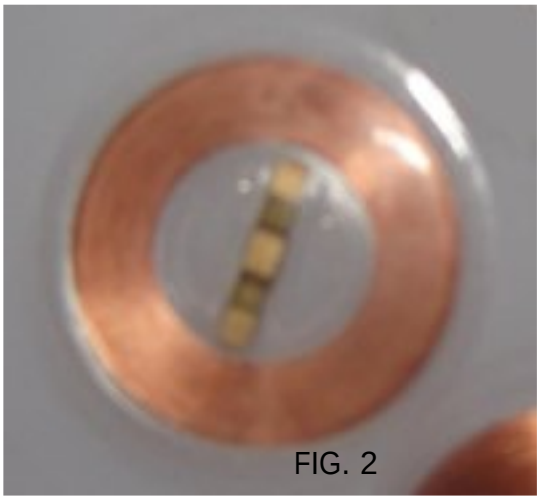


FIG. 2

A Antena RFID

A antena é o elemento condutivo que permite a etiqueta receber e enviar dados para o leitor. Tanto o leitor como a etiqueta RFID possui uma antena RFID,

sendo imprescindível para a comunicação por rádio frequência. A antena da etiqueta aproximando da antena do leitor forma um campo magnético, meio pelo qual as ondas de rádio são transmitidas.

Anti-Colisão

A anti-colisão é um método de prevenir que ondas de rádio de um dispositivo interfiram nas ondas de rádio de outro. É implantado por algoritmos anti-colisão que permite a leitura de vários dispositivos ao mesmo tempo por um único campo magnético de um leitor.

Aplicações e usos do RFID

A RFID é usada em inúmeros ramos de actividade. A cada dia, novas aplicações são concebidas para o uso desta tecnologia que vem revolucionando diversos sectores da economia como por exemplo no controlo de Abastecimento de Frota com o objectivo é evitar o roubo de combustível, na medicina utilizando chips para monitorizar a cicatrização em implantes ósseos, fornecendo todas as informações necessárias para recuperação do paciente, monitorizar tumores, em bibliotecas através do uso de etiquetas inteligentes para identificar os livros, controlando o empréstimo e possíveis tentativas de furto por meio de leitoras nas portas de acesso, entre muitas outras aplicações.

Conclusões

A tecnologia de identificação por frequência de rádio está em expansão.
O Sistema RFID é simples e pode ser usada em inúmeras áreas de actividade como: abastecimento de frota, medicina, bagagem de aviões, etc.

Pelo menos por enquanto a tecnologia não é um substituto do código de barras, pois o custo da impressão do código nas embalagens é insignificante se comparado ao custo de uma tag. Porém a grande vantagem é poder ter inúmeras informações do produto ou objecto, identificando inúmeros itens ao mesmo tempo, sem haver a necessidade de os tocar.
O seu uso também se dá em combinação com outras tecnologias como o Sistema de Posicionamento Global (GPS).
Com isso o RFID tende a ser uma solução promissora para o mercado na questão de identificação.

Devido a este crescimento tão significativo nos últimos anos, várias empresas começaram a apostar no desenvolvimento de soluções deste tipo. Em Portugal algumas empresas de diferentes áreas de negócio (mas aplicadas ao RFID) juntaram-se, como foi o caso da Sybase, CreativeSystems e Paxar, formando a parceria RetailID que providencia soluções RFID para a indústria do retalho. A Sybase, também lançou este ano um portal com o objectivo de oferecer suporte na área do RFID de modo a oferecer mais condições à propagação desta tecnologia, podendo consultado online em www.portalrfid.net .

Referências
ASK SOLUTION - Obid - Gate solutions for LR200 and LRM200.
COMPUTERWORLD - Porta-voz do Mercado de TI e Comunicação
IBOPE - Número de cartões com chip cresce 204% em 2001
TELECO - RFID: Tecnologia Entendendo a Tecnologia
RFID JOURNAL- The world's RFID Authority, RFID Tags How much information can an RFID tag store?

Evandro Oliveira, Fabiano Ferreira
Leandro Belo e Werichnilson Ataliba

Revista Linux

A Verdadeira Revista Portuguesa de Linux

Edição Bimestral em formato PDF

Download Gratuito

www.revista-linux.com

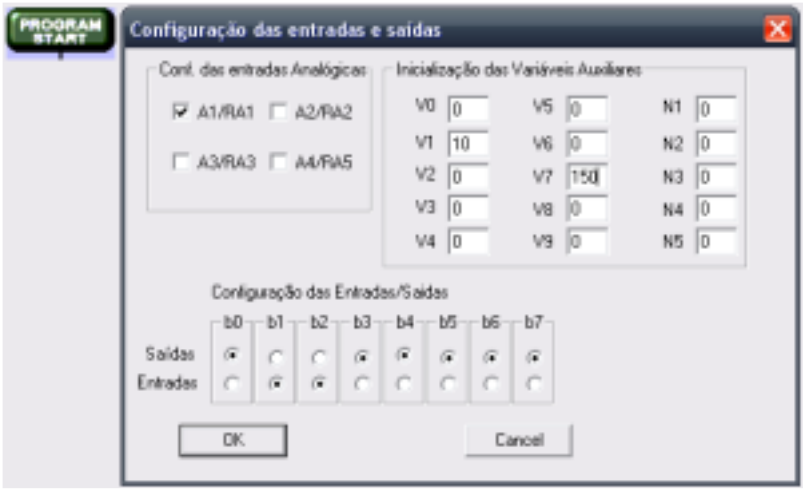
Programação em Ambiente Gráfico: Circular GT

A robótica é, sem dúvida, uma área em crescente expansão nas sociedades modernas, visto que hoje em dia existe uma grande necessidade de se realizar tarefas com eficiência e precisão, ou onde a presença humana se torna difícil, arriscada ou até mesmo impossível, como o fundo do mar ou a imensidão do espaço. Como tal, esta necessidade de descobrir e melhorar "seres" autónomos capazes de nos ajudarem, contribui para que também na área lúdica e do lazer se desenvolvessem dispositivos capazes de maravilhar o ser humano.

Neste sentido, o kit, que pode ser comprado à empresa IdMind, mostra muito bem esse prazer não só lúdico, mas também técnico, que tais dispositivos proporcionam. Com efeito, neste artigo (que constitui apenas a primeira parte) falarei da programação gráfica disponibilizada pela empresa aquando da compra do robô, mais concretamente nos blocos de programação, visto a linguagem chamada Circular GT ser em ambiente gráfico.

Função de cada bloco de programação

Antes de explicar a função de cada bloco, torna-se necessário explicar a configuração das entradas e saídas do microcontrolador. Assim, para se efectuar essas definições, deve-se carregar com o botão esquerdo do rato no bloco "PROGRAM START", abrindo um quadro semelhante ao da seguinte imagem:



Como podemos observar, podemos agrupar as variáveis em três grupos, consoante a função que desempenham. Assim sendo, num primeiro grupo temos as entradas analógicas (entradas onde se ligam sensores analógicos, ou seja, sensores que trabalham com tensões, mas que posteriormente são convertidas por um sensor analógico-digital). Para podermos utilizar uma porta de entrada analógica (dispomos de quatro, de A1 a A4) devemos marcar no quadro qual a porta onde se encontra ligado esse dispositivo. Por exemplo, se ligarmos uma bússola analógica no pino 1 do PortoA, devemos marcar o quadrado à frente da entrada analógica correspondente, tal como se mostra na figura. Por sua vez, relativamente às entradas e saídas digitais (que aceitam apenas o valor lógico 0 ou 1) basta, para cada pino (dispomos de 8 – de b0 a b7), identificar as que correspondem às entradas e quais correspondem às saídas. Estas variáveis são utilizadas, por exemplo, para ligar sensores de contacto, em que o sensor ou está actuado (valor lógico 1) ou não está actuado (valor lógico 0). Por fim, as 15 variáveis auxiliares (V0 a V9 e N1 a N5) inteiras são variáveis para a exclusiva utilização do programador.

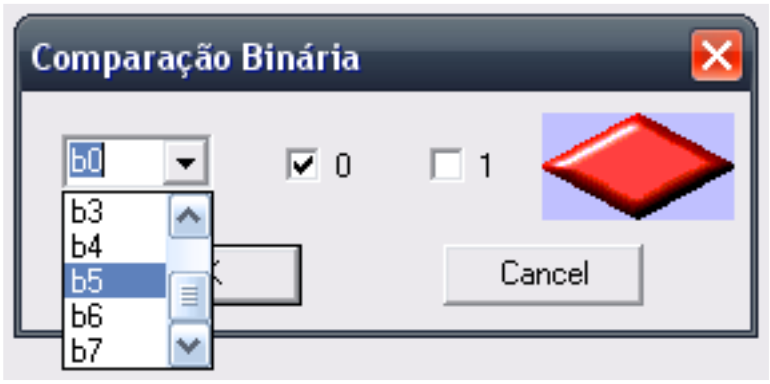
Este poderá utilizar estas variáveis como contadores, valores de nível para os vários sensores analógicos ou como variáveis de sistema que permitem armazenar valores de entradas analógicas. No menu de configuração é possível pré-definir os valores iniciais destas variáveis. Exemplo: na figura V1 é configurada inicialmente como tendo o valor 10, V7 o valor 150, todas as outras variáveis auxiliares são configuradas com o valor 0. Concluindo, agora que já foi esclarecido a que correspondem todas as variáveis, já se torna mais simples explicar a função de todo o tipo de blocos.

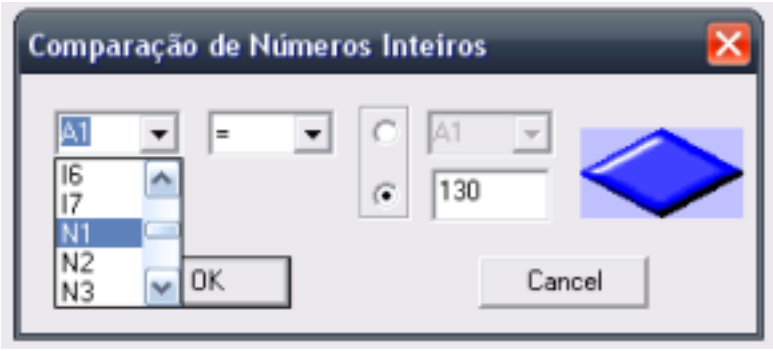
Na linguagem gráfica do Circular GT, existem diversos blocos de programação. No entanto, pelas funções que estes desempenham, podem ser agrupados em quatro categorias distintas.

Assim, na primeira categoria são apresentados os blocos que executam operações de comparação simples e operações de comparação com AND lógico. Relativamente aos de comparação simples estes podem ser de dois tipos:

- Comparações binárias  ;
- Comparações de números inteiros  ;

O primeiro bloco permite, então, que o programador faça comparações ou condições binárias, isto é, permite verificar o estado lógico de uma variável. As variáveis que podem ser encontradas neste tipo de comparação são as seguintes: b0 a b7. Para poder utilizar estas variáveis como entradas, o utilizador deverá configurá-las no menu de configuração. Por sua vez, o segundo bloco permite fazer comparações de números inteiros. Por outras palavras, podemos verificar se uma dada variável é menor, igual ou maior que outra, ou ainda comparar uma variável com um número. Aqui, as variáveis que podem ser utilizadas nestas comparações são as seguintes: I0 a I7, A1, A2, A3 e A5, V0 a V9 e N1 a N5.

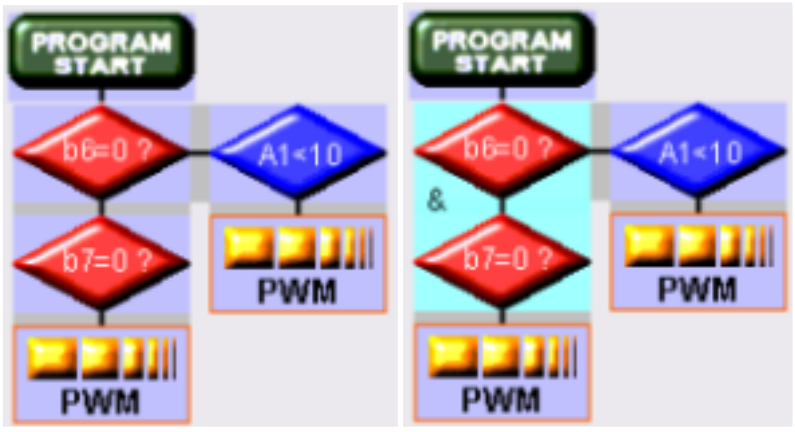




Partindo agora para os blocos de comparação com AND lógico, estes assemelham-se aos blocos de comparação simples, mas com a particularidade de poder encadear vários conjuntos de blocos de comparação, permitindo assim a existência de comparações com mais informação:

- Comparações AND binário  ;
- Comparações AND lógico com números inteiros  ;

Na verdade, a olho nu e sem encadeamento de blocos, a única diferença que nos vem à cabeça é a cor que envolve a figura. No entanto, para que se perceba bem as diferenças entre os blocos simples e com AND's torna-se importante perceber as duas figuras abaixo:



As imagens apresentadas representam dois programas diferentes. No lado esquerdo é apresentado um programa com três comparações simples e no lado direito é apresentado o mesmo programa mas com comparações com AND Lógico. No primeiro caso o programa vai iniciar a sua execução com a verificação do bit b6. Se este estiver a 0 então o programa passa para o bloco seguinte de comparação, onde se verifica se b7 também é igual a 0. É, então, neste ponto que os dois tipos de blocos diferem.

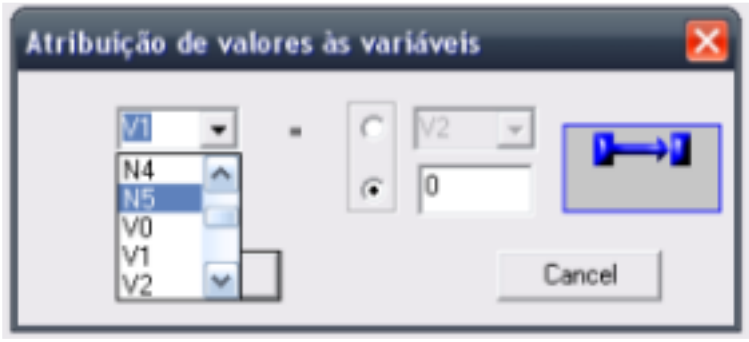
No caso dos blocos de comparação simples, se b7 for igual a 1, como não existe nenhum bloco de comparação ligado à sua direita, o programa vai recomeçar. Por seu turno, no caso da utilização de um bloco de comparação com AND Lógico, caso b7 seja igual a 1, o programa não recomeçará sem que primeiro faça o teste se A1<10. Por outras palavras, nos blocos de comparação simples as condições são verificadas uma a uma, e caso não hajam instruções a sua direita o programa vai para o início. Contudo, no outro tipo de blocos, uma vez que eles estão ligados por AND's lógicos, então funcionam como um único bloco, pelo que se falhar uma condição dos blocos ligados, o programa vai na mesma executar as tarefas nos blocos à direita, antes de iniciar um novo ciclo do programa.

Na segunda categoria podemos encontrar novamente quatro blocos, mas agora de actuação. Por exemplo, com estes blocos é permitido actuar em variáveis inteiras e actuar os sinais digitais. Como tal, para actuar em variáveis inteiras temos 3 possibilidades:

- Atribuição de valores às variáveis  ;
- Incrementa variáveis  ;
- Decrementa variáveis  ;

O primeiro bloco permite-nos atribuir um valor a uma variável auxiliar (V0 a V9 e N1 a N9).

Contudo, os valores atribuídos a essas variáveis podem ser conferidos de duas maneiras diferentes, ou seja, podem ser valores de outras variáveis inteiras (A1 a A4, I0 a I6, V0 a V9 e N1 a N5), ou valores inteiros de 0 a 25. Os dois restantes blocos permitem incrementar ou decrementar o valor de uma das variáveis auxiliares (importantes para funcionarem com contadores, ou seja, para porem o robô a realizar uma tarefa num determinado período de tempo).



Explicados os blocos para actuar variáveis inteiras, neste momento só me resta falar do actuador de saída digital :

- Set/Reset saída digital  ;

Com esta função, é possível, tal como o próprio nome indica (Set/Reset), ligar ou desligar uma das saídas b0 a b7. No entanto, para actuar a saída é necessário que esta esteja configurada como tal (ver configuração das entradas e saídas.

Para a terceira categoria estão reservados os três blocos que permitem actuar a velocidade dos motores:

• PWM (Modelação da largura de impulso) dos motores 

;

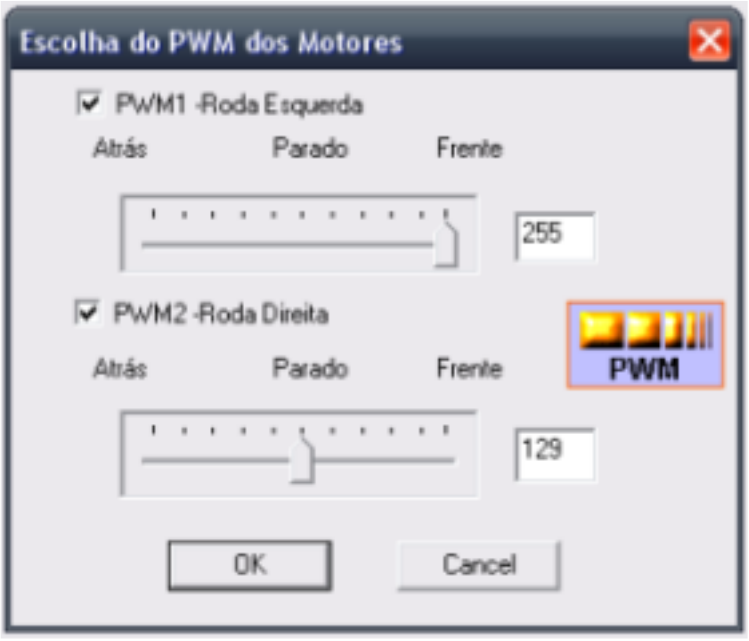
• Configuração da PWM com emissores/receptores (infra-vermelhos) . 

;

• Configuração da PWM mas com a bússola 
;

Neste sentido, o primeiro bloco representa a forma mais simples de actuar e regular a velocidade de ambos os motores, ou apenas um.

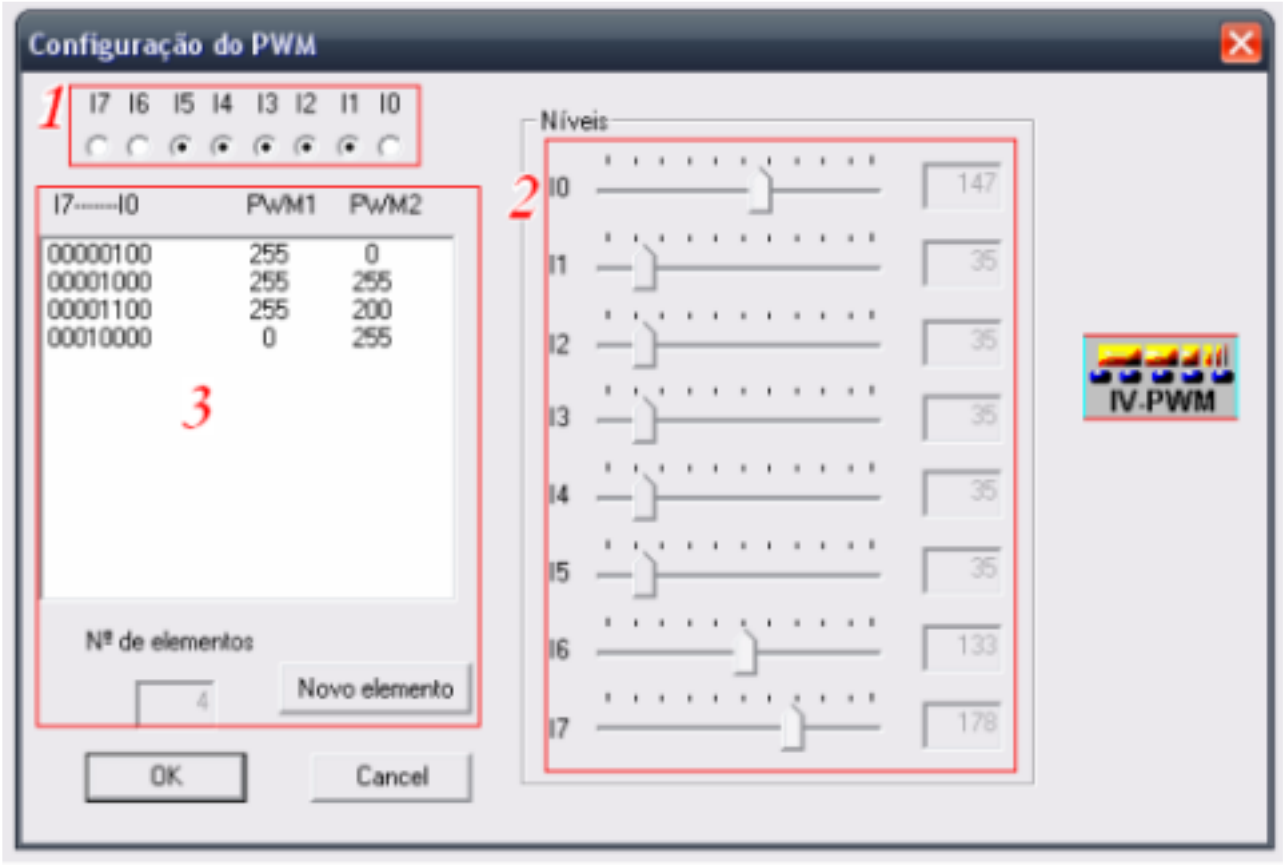
Assim, o programador pode escolher a velocidade numa escala de 0 a 255, escala esta em que o valor médio equivale a ter o motor parado, enquanto os extremos representam as velocidades mais elevadas para a frente ou para trás, conforme se pode visualizar na figura seguinte:



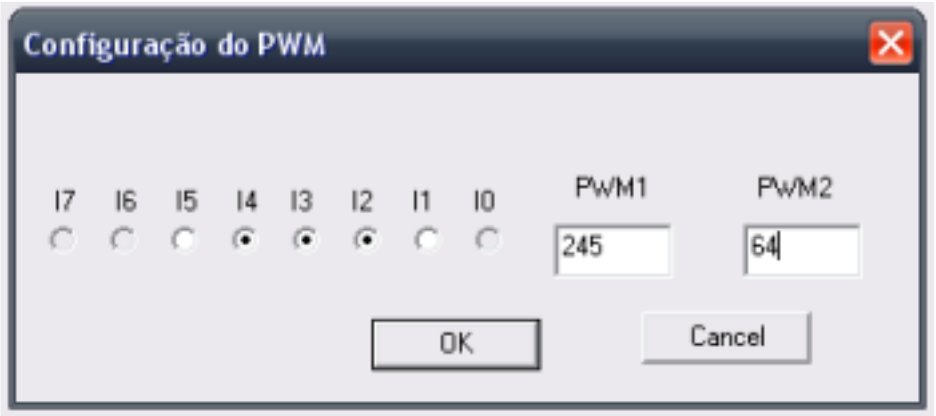
Por sua vez, o segundo bloco desta categoria permite relacionar as leituras dos sensores de infravermelhos e LDR's (resistências que variam com a luz) existentes no robot, I0, I1, I2, I3, I4, I4, I5 e I6, com diferentes velocidades para cada uma das rodas.

Na verdade, analisando a imagem aqui apresentada, podemos distinguir no menu três campos distintos (assinalados a vermelho). Assim, no topo superior esquerdo desta janela (zona identificada com o 1) é apresentada a configuração dos sensores que se pretende utilizar. Neste caso, a figura mostra a configuração para a utilização de I1, I2, I3, I4 e I5. No entanto, se se pretender utilizar os outros sensores de infravermelhos, teremos de assinalar aqui o sensor pretendido, pois nas etapas seguintes deixa de estar disponível. Por sua vez, na parte direita da janela, podemos visualizar a configuração dos níveis dos sensores (zona identificada com o 2). Ora os níveis são basicamente valores de tensão entre 0 e 5V fornecidos pelos sensores e convertidos pelo microcontrolador (contém um conversor analógico-digital) em valores entre 0 e 255.

Por outras palavras, depois dos sensores escolhidos serem lidos, os valores são comparados com os níveis, sendo convertidos em binário, zero ou um, conforme estejam acima ou abaixo do nível. Neste sentido, se o valor lido for superior ao valor do nível, neste caso o bloco considera que o valor do sensor é 1 (actuado), colocando imediatamente os motores em funcionamento conforme os personalizaremos na etapa seguinte. Por outro lado, se o valor lido for inferior ao valor do nível, então o bloco considera que o valor do sensor é 0 (inactivo), não colocando, portanto, os motores em funcionamento conforme especificado na PWM.



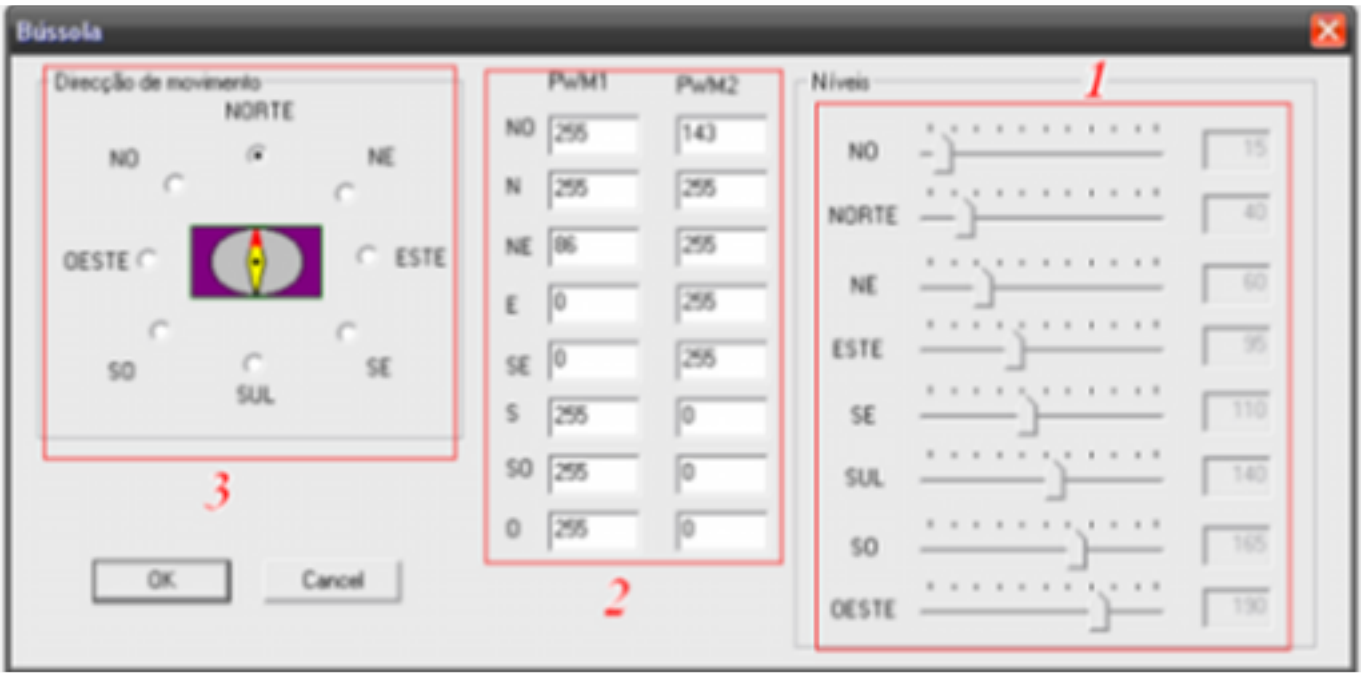
Por fim, o terceiro campo, situado na parte do meio e inferior esquerda da janela, diz respeito à configuração da modelação de largura de impulso (PWM). Com efeito, depois dos sensores serem lidos e obtidos os valores lógicos, 0 (não actuado) e 1 (actuado), é possível tratar todas as situações, correspondendo à tabela lógica, até 8 bits. O utilizador poderá, para cada uma das situações, programar a velocidade dos dois motores. Para introduzir um novo elemento na lista basta carregar na opção de introdução "Novo Elemento". É aberta uma nova janela que permite seleccionar a nova configuração de infravermelhos, bem como os dois novos valores a serem enviados para os motores. É de notar que só se poderá adicionar elementos utilizando os sensores anteriormente assinalados. No entanto, é de grande importância revelar que este tipo de bloco é apenas uma alternativa mais simples daquilo que se poderia fazer com uma série de blocos.



Por exemplo, em vez do bloco específico de PWM através de infravermelhos poderíamos substituir, para este caso, por:

Para terminar a parte de regulação da velocidade dos motores, falta falar do bloco de configuração da PWM com bússola electrónica. Assim, este bloco tem certas semelhanças com o anterior, embora seja num contexto diferente.

No entanto, antes de explicar o bloco em questão, convém referir que só o podemos utilizar se o robot possuir uma bússola electrónica devidamente ligada e configurada ao porto analógico PortA1. Assim, ao utilizar este bloco, vai aparecer o seguinte painel de configuração:



Na verdade, como podemos visualizar, este painel encontra-se dividido em três partes (da direita para a esquerda): a primeira (assinalada com o 1) diz respeito aos níveis de detecção para uma determinada orientação, em que o programa considera que o robô se encontra orientado para Noroeste se o valor da leitura analógico estiver entre o valor 0 e o valor definido no 1º campo (NO - 15), para Norte se o valor analógico se encontrar entre o valor definido para Noroeste (15) e Norte (40), e assim sucessivamente (é de assinalar que os valores podem ser alterados pelo utilizador, todavia corre o risco de depois não corresponder verdadeiramente o valor com o respectivo ponto cardeal).

A segunda, a coluna central, corresponde às acções a tomar (velocidade dos motores) quando os sensores se encontram nas determinadas posições, de forma a conduzir o robô para norte (objectivo – baliza adversária).
Por último, na coluna da esquerda podemos definir qual a orientação que queremos que o robot siga, não sendo necessário a alteração de nenhum dos campos anteriormente definidos.
Por fim, a quarta e última categoria é constituída por um bloco temporizador que permite "obrigar" o robô a fazer uma determinada acção durante um determinado período de tempo:

- Bloco temporizador  ;

Com este bloco, apenas temos de configurar o tempo de execução da manobra definida no momento anterior. O tempo encontra-se escalado em períodos de 0,25 segundos, e deverá estar dentro do intervalo de 0 a 64 segundos.
Finalizada a primeira parte do artigo, o leitor encontra-se apto e pronto para que na próxima edição compreenda com normalidade os programas e exemplos práticos que analisarei, ficando também a promessa de elucidar a forma como se processa a comunicação com o compilador. 

Link: www.idmind.pt

Sandro Pinto

Criar um RSS Feed em PHP + MySQL

Com o crescimento exponencial que a Internet tem, cada vez mais se torna necessário desenvolver tecnologias que permitam cativar o visitante de um website. Neste artigo vamos aprender a desenvolver um sistema de RSS que permite a qualquer visitante tomar conhecimento das novidades do seu website mesmo sem o abrir no "browser".

Este script será desenvolvido com recurso a PHP e MySQL, de forma a criar um RSS dinâmico, e a estar sempre actualizado cada vez que for solicitado por parte de um leitor de RSS.

Para tornar este artigo um exemplo prático, vamos criar então uma base de dados com o nome de "exemploRSS" e uma tabela de notícias que irá conter todos os itens que irão estar no canal de RSS.

Para tal podemos utilizar o seguinte código SQL:

```
CREATE DATABASE `exemploRSS`;
USE `exemploRSS`;

CREATE TABLE `noticias` (
  `id` int(11) NOT NULL
AUTO_INCREMENT,
  `titulo` varchar(255) NOT NULL,
  `noticia` text NOT NULL,
  `data_hora` datetime NOT NULL
DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM;
```

Se o leitor já tem a sua base de dados e uma tabela que pretende ver incluída no seu canal de RSS, ignore o passo anterior, pois mais à frente iremos ver como personalizar o script para qualquer base de dados e tabela.

Agora passamos à parte de PHP onde iremos criar a classe de RSS e as definições necessárias à geração do canal. Para tal criamos um ficheiro com o nome "rss.php" e incluímos:

```
class RSS
{
  var $rss_version     = '2.0';
  var $rss_title       = '';
  var $rss_link        = '';
  var $rss_description = '';
  var $rss_language    = 'pt-pt';
  var $rss_pubDate     = '';
  var $rss_lastBuildDate     = '';
  var $rss_docs        =
'http://blogs.law.harvard.edu/tech/rss
';
  var $rss_generator   = '';
  var $rss_managingEditor    = '';
  var $rss_webmaster   = '';
  var $itens           = '';
```

Estas serão as variáveis de classe que iremos utilizar para podermos gerar o nosso canal de RSS, mas vamos ver em detalhe para que serve cada uma delas:

• $rss_version : Versão correspondente ao canal RSS gerado.
• $rss_title : Título do canal RSS. Normalmente é o nome do website.
• $rss_link : Link para o website.
• $rss_description : Uma breve introdução sobre o website e seus conteúdos.
• $rss_language : O idioma no qual o canal RSS é escrito. (listagem completa de idiomas em http://backend.userland.com/stories/storyReader$16)
• $rss_pubDate : Detalhes da data e hora da última actualização.
• $rss_lastBuildDate : Detalhes da data e hora da última geração.
• ss_docs : Link para a documentação referente à versão utilizada neste canal RSS.
• $rss_generator : Indicação do sistema que fez gerar o respectivo canal de RSS. No nosso caso é o link do website.
• $rss_managingEditor : Nome ou email da pessoa responsável pelos contéudos conteúdos.
• $rss_webmaster : Nome ou email do webmaster do site.
• $itens : Variável que irá conter todos os itens do nosso canal.

Agora vamos desenvolver o construtor da classe, que será responsável por carregar grande parte das variáveis de classe logo na sua criação:

```
function RSS( $values ) {
      foreach ( $values as $key =>
$value ) {
            $this->$key = $value;
      }
}
```

O construtor recebe uma variável em forma de array contendo índices com os mesmos nomes que as variáveis de classe, tornando simples o carregamento das variáveis de classe através de um simples ciclo do tipo 'foreach'.

Através do construtor de classe, sempre que esta for instanciada é são logo carregados os dados principais referentes ao canal de RSS. Este é o primeiro passo de um total de três, ficando a faltar o carregamento de todos os itens a aparecer no canal e a finalização e geração do respectivo canal.

Vamos então passar à inclusão dos itens para o canal:

```
function set_item( $values ) {
      $this->itens .=  '<item>
            <title>'.$values[
'item_title' ].'</title>
            <link>'.$values[
'item_link' ].'</link>
            <description>'.$values[
'item_description' ].'</description>
            </item>';
      }
```

Com o método "set_item()" da class, recebemos um array que contem os dados de cada item, e como um canal pode conter vários itens, este método é responsável por receber cada item e acumulá-los na variável $itens da class. Cada item que incluímos contém: um título, um link directo para a notícia e a descrição da notícia.

Já estamos muito perto do final da nossa classe, só nos falta compilar toda a informação retida e gerar o nosso canal de RSS.

```
 function output_rss( ) {

  $output =  '<?xml version="1.0"?>

<rss version="'.$this->
rss_version.'">

    <channel>

     <title>'.$this->
rss_title.'</title>

     <link>'.$this->
rss_link.'</link>

     <description>'.$this->
rss_description.'</description>

     <language>'.$this->
rss_language.'</language>

     <pubDate>'.$this->
rss_pubDate.'</pubDate>

     <lastBuildDate>'.$this->
rss_lastBuildDate.'</lastBuildDate>

     <docs>'.$this->
rss_docs.'</docs>

     <managingEditor>'.$this->
rss_managingEditor.'</managingEditor>

     <webMaster>'.$this->
rss_webmaster.'</webMaster>';

     $output .= $this->itens;

    $output .= ' </channel>
   </rss>';

  return $output;
 }

}
```

O método 'output_rss()' é responsável primeiramente por utilizar as variáveis de classes carregadas na instanciação para gerar o código XML que identifica o canal de RSS, juntando tudo numa variável temporária com o nome '$output'.

Depois juntamos à variável '$output' todos os itens existentes na variável $itens da class, e finalizamos o canal, retornando a variável temporária.
Desta forma finalizamos a nossa classe e está pronta a ser integrada em qualquer website que tenha base de dados MySQL.
Ainda no mesmo ficheiro vamos criar todas as configurações necessárias para tirar partido da classe que criámos.

```
$connection = mysql_pconnect(
'localhost' , 'nome_do_utilizador',
'password' ) or die( mysql_error( )
);
mysql_select_db( 'exemploRSS' ,
$connection );
$noticias = mysql_query( 'SELECT *
FROM noticias ORDER BY data_hora
DESC LIMIT 15' , $connection ) or
die( mysql_error( ) );
```

Com estas linhas de código iremos fazer a ligação à base de dados, seleccionar a base de dados e executar a consulta que nos permite recolher os dados a incluir no canal RSS.

Convém saber que ao executarmos a consulta não deveremos recolher a totalidade de registos existentes na tabela, porque imaginando que existem 150 registos, todos eles serão carregados no canal, e quando um utilizador subscrever o canal, o mesmo irá recolher os 150 registos. Assim utilizamos "...LIMIT 15" para recolhermos apenas 15, e não tornarmos o canal tão extenso.

Vamos então preparar um um array com os dados de identificação do nosso canal:

```
$myFeed[ 'rss_title' ] = 'Portugal-a-Programar';
$myFeed[ 'rss_link' ] = 'http://www.portugal-a-programar.org';
$myFeed[ 'rss_description' ] = 'A comunidade Portuguesa de Programadores';
$myFeed[ 'rss_pubDate' ] = date("D, d M Y H:i:s T");
$myFeed[ 'rss_lastBuildDate' ] = date("D, d M Y H:i:s T");
$myFeed[ 'rss_generator' ] = 'RSS gerando por Portugal-a-Programar';
$myFeed[ 'rss_managingEditor' ] = 'bruno.vaz@netvisao.pt';
$myFeed[ 'rss_webmaster' ] = 'bruno.vaz@netvisao.pt';
```

E vamos instanciar a classe de RSS com o array previamente carregado com os dados de identificação:

```
$rss = new RSS( $myFeed );
```

Utilizando a consulta anteriormente realizada, iremos então carregar a classe com os itens existentes na base de dados:

```
while( $linha = mysql_fetch_assoc( $noticias ) ) {
	$myItem[ 'item_title' ] = $linha[ 'titulo' ];
	$myItem[ 'item_link' ] = 'http://www.portugal-a-programar.org/noticia.php?id='.$linha[ 'id' ];
	$myItem[ 'item_description' ] = htmlentities($linha[ 'noticia' ] );
	$rss->set_item( $myItem );
}
```

Com toda a informação carregada na classe, só nos falta gerar o código XML do nosso canal de RSS para que qualquer leitor de RSS possa realizar a subscrição:

```
header( "Content-Type: application/xml" );
echo $rss->output_rss();
?>
```

Antes de gerar o código através da classe deveremos informar que o conteúdo gerado pelo ficheiro 'rss.php' é XML, e para tal utilizamos a função 'header()'.

Agora podemos então testar o nosso canal de RSS. É só colocar online, e utilizando qualquer leitor de RSS apontar para o respectivo link, por exemplo:

http://www.portugal-a-programar.com/rss.php

E iremos ver o nosso canal a ser subscrito pelo leitor. Naturalmente que ao subscrever não irá ter nenhuma notícia, mas o leitor pode adicionar quantas quiser na base de dados, e ver o leitor de RSS automaticamente a carregá-las.

Para tal pode usar-se o seguinte código:

```
INSERT INTO `noticias` (`titulo`, `noticia`) VALUES (`Sistema de RSS online!`, `Informamos os nossos utilizadores que est&#225; agora online um sistema de feeds RSS online. Quaisquer coment&#225;rios dever&#227;o ser feitos por email para rss@exemplo.com`)
```

Agora o leitor está habilitado a criar o seu RSS Feed. Espero ter ajudado.

Bruno Vaz

GOSTAS DE PROGRAMAÇÃO?

É DIFÍCIL ENCONTRARES TUTORIAIS EM PORTUGUÊS DE PROGRAMAÇÃO?

SENTES FALTA DE LER O QUE TE INTERESSA?

REVISTA

PROGRAMAR

acede já a

www.revista-programar.info

e vira uma página na tua vida

um projecto

portugal-a-programar.org

Compilar, testar e detectar erros de C em Linux

Praticamente todos os programas existentes em Linux foram escritos em C.

O próprio Linux foi escrito em C (embora não na totalidade), portanto é de esperar que seja possível programar-se em C usando uma distribuição instalada de raiz, e de facto é na maioria dos casos, sem recorrer a qualquer programa adicional.

Introdução

A primeira distinção necessária é a de editor versus compilador.

O editor é o programa que nós usamos para escrever o nosso código. Existem milhares de programas que o fazem, tanto em Windows como em Linux, desde os simples gEdit ou Kate aos mais complexos Vi, eMacs, Joe... a lista é praticamente interminável.

Deixo a escolha do editor ao vosso critério – é uma questão de gosto pessoal. Se são novos neste mundo da programação em Linux comecem com algo simples, e se se sentirem limitados passem para algo mais complexo.

Do outro lado da barricada, aquele em que nos vamos concentrar, encontra-se o compilador. O compilador é responsável por transformar o vosso código em algo interpretável pela máquina.

Ao longo do resto do texto vou partir do princípio que já programaram em C antes, pelo menos em Windows.

GCC

O gcc é provavelmente o programa mais usado para compilar C.

E depois de se mexer um pouco com ele percebemos facilmente porquê: é simples, rápido, com poucos erros, e permite um conjunto enorme de argumentos que nos permitem optimizar o nosso programa: programa rápido, programa pequeno, etc...

Depois de terem escrito o vosso primeiro programa em Linux usando o vosso editor favorito (um simples "Olá Mundo" é suficiente para começar) e de o terem guardado no disco, coloca-se a grande pergunta: e agora?

Simples:

```
gcc ola.c -o ola
```

E acabaram de criar o vosso primeiro programa, com o nome “ola”. Para o correr:

```
./ola
```

E o resultado é imediato:

```
Ola Mundo
```

Mas as opções são muitas, as flags de compilação que podemos usar são imensas:

```
-Wall : para avisar todos os
warnings
-O1 , -O2, -O3 : várias
optimizações de código (tempo de
processamento, tamanho do ficheiro
de saída)
-g : para permitir debugging.
Vamos vê-la mais à frente.
-pedantic
-ansi
```

Só para citar as mais comuns, porque existem centenas de outras.

Ao longo do texto vou compilar usando o seguinte formato: (o mais simples)

```
gcc file.c –o file -g
```

Debugging – o gdb

Programação é uma arte exacta. Os comandos produzem um resultado predefinido, e ao programador só resta ordená-los de modo a produzir o efeito desejado.

Quando os programas não fazem o que nós esperamos, então começa a fase de debugging: vamos percorrer o nosso código à procura do erro.

O gdb (gnu debugger) é um programa que nos ajuda nesta busca. Para se usar o gdb devemos compilar o programa com a flag –g.

Vamos começar por usar como exemplo o seguinte programa:

```
#include <stdio.h>

int main() {
        int k;
        scanf("%d",k);
```

Somente quando corremos o programa e inserimos um número qualquer (que o nosso programa era suposto ler) é que nos deparamos com um "Segmentation Fault", não nos dando qualquer ajuda extra. São os terrores dos programadores.

Como detectar a localização do erro sem grandes dificuldades? Neste programa até era relativamente simples, é um programa pequeno, mas quando o nosso código atinge centenas ou milhares de linhas, pode-se tornar uma tarefa devastadora, principalmente quando pequenos caracteres (ainda menos evidentes do que este) fazem toda a diferença.

Entra em acção o gdb. Começamos por correr o gdb com o nosso programa (chamei-lhe "ex1")

```
gdb ex1
```

Atenção que este passo deve ser dado DEPOIS de se compilar o programa usando a flag -g.
Entramos imediatamente no mundo do gdb, e o programa aguarda as nossas ordens.
Vamos mandá-lo correr o programa.

```
(gdb) run
```

Se o meu programa precisasse de argumentos extra, poderia passá-los depois do comando "run".
Vamos fazer aqui um pequeno aparte para nos ambientarmos com o gdb.
Se eu quiser parar a execução do programa posso fazer "kill":

```
(gdb) kill
```

E se me tiver arrependido de o ter parado, posso chamar "run ..".

```
(gdb ) run ..
```

Para sair do gdb, tenho sempre a opção de "quit". Podem-me perguntar se pretendo terminar a execução actual do programa.

```
(gdb) quit
The program is running. Exit anyway?
(y or n) y
```

Finalmente, para obter ajuda: "help"

Vamos voltar ao nosso programa.

Portanto, estamos no gdb com o programa carregado e a correr (com o "run").

Ele pede-nos um valor de entrada (o número que nós pretendemos ler) e nós introduzimos um número qualquer. Imediatamente, o programa responde:

```
Program received signal SIGSEGV,
Segmentation fault.
0x49e9006a in _IO_vfscanf_internal () from
/lib/libc.so.6
```

Como podemos ver, o nosso Segmentatin fault chegou. (junto com alguma informação, como o endereço de memória etc., detalhes mais avançados, mas que já nos poderiam ajudar).

Mas podemos ir mais longe, onde é que o programa deu segmentation fault? Podemos perguntar:

```
(gdb) where
```

E a resposta, do topo para a base, são as funções chamadas:

```
#0 0x49e9006a in
_IO_vfscanf_internal () from
/lib/libc.so.6
#1 0x49e94feb in scanf () from
/lib/libc.so.6
#2 0x08048378 in main ()
```

Como podemos ver, o segmentation fault deu-se na função scanf, que se encontra dentro da função main. Bastar-nos-ia analisar a função para rapidamente descobrir o erro.

Se quisermos aceder ao código não precisamos de sair do gdb. O comando "list" devolve-nos automaticamente as "proximidades" da linha geradora do erro. (neste caso até devolveu o programa todo..)

```
(gdb) list
#include <stdlib.h>

int main() {
            int k;
            scanf("%d",k);
            return 0;
}
```

Acabámos de detectar o nosso primeiro erro através do gdb. Usando windows perderíamos uma infinidade de tempo a analisar as linhas todas. (ok, talvez não muito porque é um programa pequeno, mas reparem que o resultado num programa grande seria o mesmo com o gdb, analisar as linhas todas num programa grande é mais complicado) Mas as opções do gdb não terminam por aqui. Vejamos este programa:

```
int main () {
            int x = 30;
            int y = 10;
            x = y;
            return 0;
}
```

Não precisam de procurar erros porque este está correcto. (pelo menos era a minha intenção). Vamos recorrer a ele para explorar outras opções do gdb. Quando o carregamos e corremos, nada de estranho acontece.

```
(gdb) run
Starting program:
/home/linux/Miguel/ex2
Program exited normally.
```

No entanto, vamos explorar o programa recorrendo aos breakpoints. Os breakpoints são paragens na execução do código, vejamos:

```
(gdb) break main
(gdb) run
Starting program:
/home/linux/Miguel/ex3

Breakpoint 1, main () at ex2.c:2
2 int x = 30;
```

Como podemos ver, o programa parou a execução na função main. Agora podemos ordenar ao gdb para executar uma instrução de cada vez. Para isso, usamos o "next".

```
(gdb) n /* n é uma abreviatura de
"next" */
```

Como é que sabemos que a instrução foi executada? Podemos ver qual o valor da variável X, usando o "print".

```
(gdb) p x /* p é a abreviatura de
"print", de seguida indicamos qual
a variável a imprimir */
```

Veja também qual o valor da variável Y, para descobrir que ainda não é 10. (de facto deve ser um valor qualquer, aleatório, uma vez que aquela posição de memória ainda não foi inicializada, contém o valor anterior lá armazenado).

O comando "next" executa a função/instrução actual, e passa para a próxima.

Se o erro estivesse dentro da função que vai ser executada, podíamos saltar para dentro dela usando o comando "step". O step entraria dentro na função, em vez de simplesmente a executar na sua totalidade. Podem testá-lo em qualquer programa que recorra a uma função (preferencialmente definida por vocês).

Podem usar "next"s repetidamente até ao final do programa ou a simples função "continue" para o deixar executar normalmente até ao final.

```
(gdb) c /* c é a abreviatura de
"continue" */
Continuing.
Program exited normally.
```

Para se ver uma lista dos breakpoints actuais pode-se usar o comando "info breakpoints".

Para se desactivar o breakpoint, usa-se o comando "disable". Vejam:

```
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x08048335 in
main at ex3.c:2
        breakpoint already hit 1 time

(gdb) disable 1
```

Curiosamente, se voltarem a listar os breakpoints continuam a encontrá-lo lá.

```
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep n 0x08048335 in
main at ex3.c:2
        breakpoint already hit 1 time
```

Reparem, contudo, que agora o campo “Enb” (abreviatura de enable) está a “n”, de "no".

De facto, se correrem o programa, podemos constatar que ele ignora o break point.

Outras opções:

Podem usar breakpoints temporários com o comando “tbreak”. "tbreak" define um breakpoints temporário que pára o programa uma única vez e depois é removido.

Podem também mandar o gdb ignorar alguns breakpoints.

```
(gdb) ignore 1 5
```

Vai ignorar as próximas 5 passagens no breakpoint número 1.

Conclusões

Como podem ver, o gdb é uma ferramenta poderosa, que quando totalmente dominada pode ser a melhor arma de um programador.

A sua complexidade de utilização inicial é uma característica da sua facilidade de utilização para utilizadores mais avançados, uma vez que (como quase todo o mundo linux) privilegia atalhos e um sistema de linha de comandos que nos deixa com algumas instruções fazer debugs complexos.

Gostaria de lembrar que só falei do básico do básico do gdb. Permite fazer muito mais, como chamar funções implementadas no C, analisar a memória, utilizar watch points, aceder aos registos do processador, ver o código assembly do programa, analisar a “call stack”

Olimpíadas Nacionais da Informática 2007

No passado dia 18 de Maio realizou-se a final das Olimpíadas Nacionais de Informática, um concurso nacional de programação que tem como objectivos seleccionar quatro representantes portugueses para as Olimpíadas Internacionais de Informática e promover o gosto pela programação entre os jovens. O evento seguiu-se a uma prova de qualificação previamente realizada, onde haviam sido seleccionados os 21 finalistas, entre os quais 5 utilizadores do fórum portugal-a-programar. Decorreu no Departamento de Ciência de Computadores (DCC) da Faculdade de Ciências da Universidade do Porto (FCUP), e juntou jovens programadores de todo o país, numa iniciativa da Associação para Promoção e Desenvolvimento da Sociedade da Informação, realizada em Portugal há já alguns anos.

Ao longo da tarde, os finalistas foram recebidos no DCC, onde puderam já trocar algumas impressões com os outros concorrentes. O almoço foi servido no bar; seguidamente os participantes foram conduzidos a um anfiteatro, onde lhes foram dadas indicações de última hora. Finalmente, às 15:00, a prova teve início: durante quatro horas, os concorrentes tentaram resolver três problemas de programação, submetendo as suas soluções num sistema automático de avaliação chamado Mooshak, criado na Faculdade do Porto.

Após a competição, os participantes foram novamente encaminhados para o anfiteatro, onde a organização lhes transmitiu algumas considerações finais sobre a prova. De seguida, o grupo dirigiu-se ao Hotel Ipanema, onde se realizou um jantar/convívio e entrega de prémios.

Todos os concorrentes receberam livros de programação. De realçar que os oito primeiros têm ainda a oportunidade de frequentar um estágio com professores universitários, donde sairão os quatro representantes portugueses. Finalmente, o vencedor Miguel Araújo recebeu também um exemplar do sistema operativo Windows Vista.

Salientamos o facto de todo o software e tecnologias utilizadas durante o evento serem open-source (excepto o sistema operativo Windows XP que, por opção, poderia ser utilizado pelos concorrentes durante a prova), o que vem contribuir para a divulgação deste tipo de filosofia em Portugal; durante a recepção os concorrentes receberam inclusive um DVD da distribuição de Linux, Knoppix.

É de saudar esta e outras iniciativas deste tipo, que incitam à aprendizagem e fomentam o prazer que é programar - nada melhor que fazê-lo quando ainda podemos ganhar prémios por isso! - aliando a programação ao convívio entre jovens. Lembramos que qualquer aluno do ensino secundário ou básico pode participar nesta prova, cujas inscrições costumam ter início alguns meses antes.

Toda a informação sobre a prova e abertura de inscrições da próxima edição pode ser consultada em www.dcc.fc.up.pt/oni

Pedro Abreu

Miguel Araújo, vencedor das Olimpíadas Nacionais da Informática 2007

(Da esquerda para a direita) Em baixo, Pedro Abreu e Pedro Diogo; Em cima, João Matos e Miguel Pais, alguns dos participantes membros do portugal-a-programar.

Web 2.0

Provavelmente já ouviu a expressão "Web 2.0". Quando a ouviu pela primeira vez, de certo se questionou do que seria esta "Web 2.0", se seria uma nova web. Este artigo tem como objectivo elucidá-lo sobre a Web 2.0, o que é e como funciona.

Web 2.0, o que é isso?

O termo "Web 2.0" é uma expressão usada pela primeira vez pela O'Reilly Media em 2003 para se referir a uma segunda "geração" de serviços e comunidades baseados na Internet.

Mas, porquê uma segundo geração? O que está a acontecer para nos referirmos a uma nova geração? Bem, muito coisa está a acontecer. A cada dia que passa, surgem novas formas de partilhar conhecimento, ideias e sentimentos, novos métodos de partilhar informação são popularizados.

Todos os programadores que desenvolvem sites e aplicações web já ouviram falar no AJAX, porque, segundo estes, torna os sites muito "in", muito "Web 2.0". Dissecando o AJAX, vemos que não é uma linguagem (como muitos pensam) mas sim uma buzzword usada para se referir a um aglomerado de tecnologias.

Estas tecnologias a que o AJAX se refere são:
- Apresentações que assentam em predefinições, ou seja, modelos que assentam em XHTML (eXtensible HyperText Markup Language), ou HTML (HyperText Markup Language), e CSS (Cascading StyleSheets) ;
- Manipulação dinâmica desses modelos usando o DOM (Document object model);
- Interacção com o servidor de forma assíncrona usando o objecto XMLHttpRequest do browser, embora por vezes se use Iframes por diversos motivos, para transmissão de dados.
- Esses dados são transmitidos no formato XML (eXtensible Markup Language), embora por vezes também seja usado JSON (JavaScript Object Notation), e a manipulação dos mesmos no lado do cliente é transmitida em XSLT (eXtensible Stylesheet Language for Transformation);
- JavaScript, usado para interligar todas estas tecnologias.

O uso de AJAX tem duas vantagens "principais":
- Separação dos dados, formatação, do estilo e da funcionalidade;
- A vantagem anterior traz consigo outra associada: a largura de banda usada. Uma vez que tudo está separado, torna-se mais fácil modificar cada coisa. Pode-se modificar uma imagem, um texto ou um link sem ter que o cliente tenha de receber tudo novamente.

Mas, todas as rosas têm os seus espinhos:
• Na integração com os browsers, existem diversos "problemas" ou incómodos. Como o conteúdo que é apresentado é dinâmico e, por isso, não é registado no histórico do browser, clicar em retroceder não produz o efeito desejado. Para este problema em específico, são por vezes usadas Iframes invisíveis para registar mudanças para que ao clicar no retroceder apenas mude essa Iframe, evitando assim incómodos ao utilizador. Este método é usado, por exemplo, no Gmail do Google. Outro incómodo está relacionado com a "dependência" de JavaScript para o funcionamento da página.

As implementações do JavaScript podem diferir nos vários browsers, as restrições ao mesmo definidas pelo utilizador ou plugins para o browser (como o NoScript para os browsers da Mozilla) podem limitar a funcionalidade das aplicações;
• A latência da rede pode por vezes tornar-se mais evidente e incomodativa para o utilizador quando não existe feedback da aplicação, um género de "aviso" de que a aplicação está a interagir com o servidor, o que o pode levar a clicar outra vez num botão, por exemplo, o que vai levar a uma segunda chamada ao servidor e pode levar a uma duplicação de dados no servidor, entre outros problemas;
• Este tipo de páginas são pouco search engine friendly devido à falta de dados aquando do caching da página pelo mesmo.

Uma solução para este problema poderá ser a "filtração" de user-agents, de forma a mostrar aos crawlers dos motores de busca páginas completas. Aos invisuais acontece o mesmo que aos motores de busca. A falta de dados aquando da aquisição da página torna impossível uma leitura por parte dos browsers especiais deste tipo de pessoas;

• A integração de serviços de estatísticas 3rd party por vezes não servem as necessidades do webmaster devido às actualizações dinâmicas do conteúdo, obrigando-o a escrever o seu próprio sistema de estatísticas.

No entanto, estas "tecniquices" suportam algo mais importante: a partilha de conteúdos e sentimentos. E é isso que a Web 2.0 "defende": uma Internet semântica, com significado para nós humanos. No YouTube partilhamos um vídeo gravado numa festa com os amigos, exibimos o nosso desktop ou mostramos o nascimento dos nossos filhos. No del.icio.us partilhamos links para uma receita do nosso bolo preferido, para opiniões que apoiamos sobre o problema da fome em África ou sobre o novo iPhone da Apple que detestamos. As pessoas partilham pensamentos e sentimentos pela Internet, tal como o fazem na vida real.

A fronteira entre o real e o virtual, o conhecimento e a ignorância, cada vez mais se torna maius ténue, tudo fruto de ideias, ideias como esta que a Web 2.0 representa.

Ler mais:
http://en.wikipedia.org/wiki/AJAX
http://pt.wikipedia.org/wiki/AJAX_%28programa%C3%A7%C3%A3o%29
http://en.wikipedia.org/wiki/Semantic_Web
http://pt.wikipedia.org/wiki/Web_sem%C3%A2ntica
http://en.wikipedia.org/wiki/Web_2.0
http://pt.wikipedia.org/wiki/Web_2.0
http://www.youtube.com/watch?v=6gmP4nk0EOE

David Ferreira

Factbites

Factibtes é o primeiro motor de pesquisa que, ao contrário de todos os outros existentes no mercado, privilegia o conhecimento ou a informação em detrimento da popularidade.
Desta forma se, por exemplo, desejar procurar por Portugal, obterá neste motor mais informação acerca do país que no Google, que maioritariamente apresentará planos de viagens.

http://www.factbites.com

VodPod

VodPod é um agregador dos vídeos disponibilizados nos diversos sites existentes para partilha dos mesmos.

Desta forma já não precisa de se preocupar em procurar o mesmo vídeo vezes sem conta em diversos sites como o youtube, o metacafe... o VodPod dá-lhe tudo.

http://www.vodpod.com

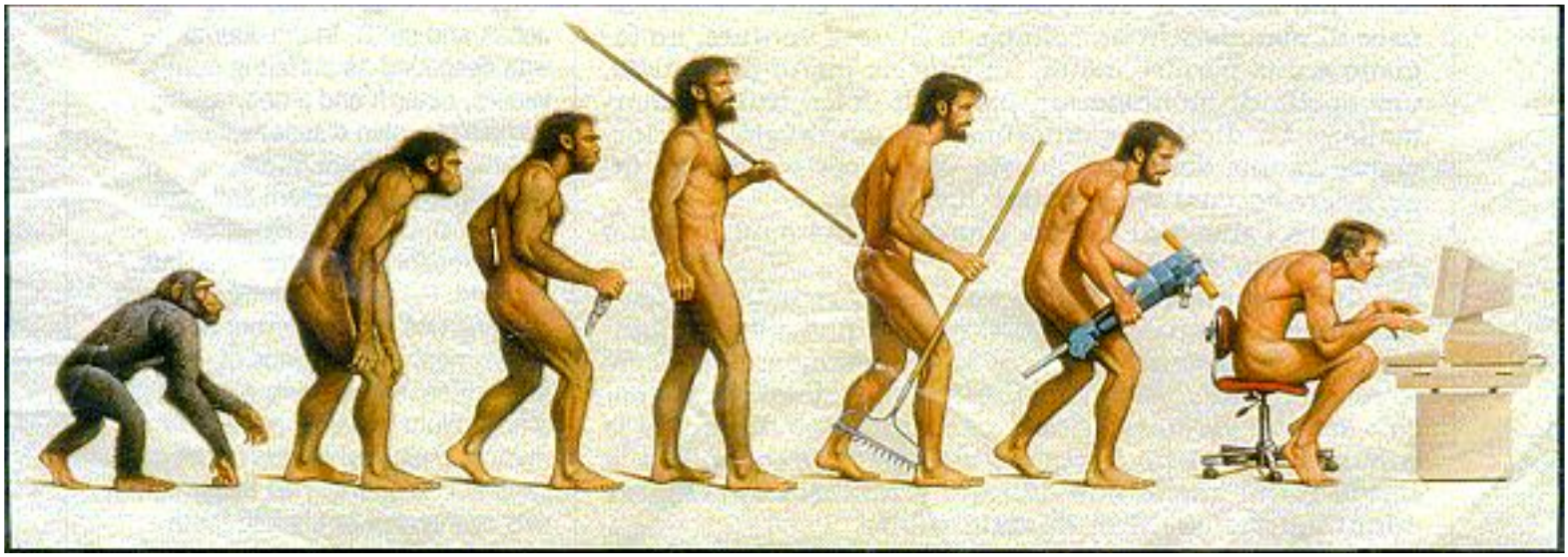
Shareminer

Uma das mais famosas aplicações na web, este query builder para o Google facilita a procura de ficheiros em vários provedores de alojamento de ficheiros, nomeadamente o RapidShare e o MegaUpload. Um site obrigatório nos favoritos.

http://www.shareminer.com

Às vezes a informática torna-se complicada...

A informática chega mesmo a todos...



A evolução do programador

Mais uma edição da tão esperada revista PROGRAMAR lançada e passamos à rotina do costume: tópicos e mais tópicos para responder, muitas discussões acesas, muitas ideias em destaque e, acima de tudo, muitos utilizadores a espalharem as suas influências e histórias pessoais pela comunidade. É, aliás, disso que cada comunidade se trata: de um enorme conjunto de influências e histórias espalhadas por cada post que foi escrito. É por isso que, de certo modo, para tomar decisões é preciso muita ponderação visto que estamos a influenciar pessoas. São as pessoas e as suas influências que fazem as comunidades. Sem pessoas não existiriam comunidades e muitas vezes alguns administradores esquecem-se desse pequeno facto. Mas no Portugal-a-Programar felizmente isso não acontece.

Os últimos dois meses têm sido meses essencialmente de reestruturação (tal como os dois anteriores). Após termos notado alguma estagnação por parte das mais variadas equipas que trabalham no Portugal-a-Programar – inclusive por parte do staff – decidimos tomar algumas medidas que serão postas em prática no decorrer das próximas semanas dos meses de Julho e Agosto. São mudanças que foram pensadas e feitas com base naquilo que achamos que as pessoas precisam e certamente trarão muito mais dinâmica à comunidade. Não vou explicar aqui de que mudanças se tratam, pois começam agora a ser feitas, pelo que sugiro que fiquem atentos nas próximas semanas à actividade no P@P.

Como disse anteriormente, alguns dos nossos projectos e equipas estão um pouco estagnados no momento. No entanto, existem equipas que muito se têm esforçado por apresentar resultados: a equipa da revista PROGRAMAR é um dos exemplos mais significativos. Graças aos esforços do Miguel Pais e do Joel Ramos em conjunto com todos os redactores, é que esta revista é lançada mais uma vez.

É a nona edição do nosso projecto mais bem sucedido até hoje, pelo que são já 18 meses de muito trabalho, organização e dedicação. Um agradecimento especial também ao Sérgio Santos, que liderou todas as 8 edições anteriores de forma exemplar e que, por motivos pessoais, teve de passar o cargo ao Miguel Pais, actual coordenador do projecto. Devo salientar que o número de artigos diminuiu nesta última edição, facto que nos levou a ter de recorrer a alguns artigos que estavam como reserva. Apelo, deste modo, a que todos os leitores que pretendam contribuir para o nosso projecto, fazer parte desta excelente casa e equipa, se registem no fórum, peçam acesso à secção privada da revista e comecem a apresentar artigos.

Para que a décima edição seja a melhor de todas e o culminar da nossa fase inicial do projecto. Que a partir da décima edição, consigamos atingir mais e mais metas, no sentido do progresso, evolução e expansão da comunidade.

Até à próxima edição.

Rui Maia

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

Vai a

www.revista-programar.info

para mais informação como participar
ou então contacta-nos por

revistaprogramar
@portugal-a-programar.org

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

contamos com a tua ajuda

Equipa PROGRAMAR

Um projecto Portugal-a-Programar.org