Formulário & Script CGI

Um Formulário de Pedidos On-line

Uma das melhores utilizações da CGI é permitir que um cliente em potencial visite seu site e encomende um produto on-line. Apesar de essas iniciativas comerciais não serem a característica mais importante da Internet, o mercado está conduzindo o desenvolvimento da Internet e de seus padrões. Fazer com que as empresas possam comercializar seus produtos on-line é o próximo passo na evolução da Inernet.

De um momento para outro, todos os comerciais de TV começaran a mostrar o endereço Web do anunciante, isso porque as empresas acreditam que a Internet tenha potencial de marketing. Originalmente as pessoas pensavem que a Web seria apenas um meio para exibir informações, mas CGI mudou isso.

Este Tutorial trata dos formulários de pedido que permitem que o visitante encomende um produto on-line. Depois de submeter o pedido, este é enviado por correio eletrônico para ser processado. Apesar de o exemplo de aplicativo deste tutorial envolver uma empresa fictícia de venda de automóveis, o mesmo formulário pode ser aplicado a todos os ramos de negócios: flores, computadores, doces ou qualquer outra coisa que possa ser vendida na Web.

A Página Web

Primeiro é preciso criar a página Web. Esta página permite que o visitante digite informações, selecione itens de listas ou checkbooks e submeta as informações digitadas, que então são enviadas para o script. Para que o documento HTML seja efetivo, ele precisa ser intuitivo. Simplesmente olhando a página, o cliente deve ser capaz de compreender qual informação precisa ser digitada em qual lugar. Quanto mais intuitiva for sua página, mais chances deu script terá de funcionar corretamente, evitanto o perigo de desencorajar o visitante.

A primeira seção do documento deve indicar ao visitante qual é a finalidade da página. No exemplo de página usado neste Tutorial, uma empresa de venda de carros deseja comercializá-los na Web.

Em seguida, deve-se permitir que o visitante digite informações pessoais como nome, endereço de correio eletrônico, endereço residencial e tudo mais que for preciso para processar o pedido e enviar o produto ao cliente.

Finalmente, ofereça ao visitante uma lista de itens para escolher. No exemplo de página usado neste tutorial, permita que o visitante escolha um carro e seus acessórios.

A Seção Introdutório do Documento HTML

Começamos criando a primeira seção do documento HTML. Nela, como mostra a lista abaixo, abrimos o documento HTML, damos a ele um título e fornecemos algumas informações introdutórias a seus visitantes

```
<HTML>
<HEAD>
<TITLE> Carros Novos Supkay </TITLE>
</HEAD>
<BODY>
<H1>Bem-vindo a Carros Novos Supkay! </H1>
<Nós temos aqui alguns <b>GRANDES</b> negócios apenas esperando você chegar e sair dirigindo com eles! Com nossa grande variedade de escolhas, estou certo de que você encontrará algo que irá <b>amar</b>!>
<HR>
<FORM ACTION="/cgi-bin/buycar.pl" METHOD="POST">
<PRE>
```

Nomeamos a página BUYCAR.HTML e o script BUYCAR.PL. Usamos o marcador <PRE> para facilitar a formatação da página, mas poderíamos ter usado tabelas.

Definindo as Informações Que Devem Ser Ocultadas do Cliente

Nosso próximo passo é decidir quais informações devem ser ocultadas do cliente. É absolutamente necessário permitir que o visitante escolha itens específicos. Também é preciso que o visitante forneça seu nome e endereço. Provavelmente teremos de determinar uma forma pela qual o visitante pagará o item que comprar, a menos que ele seja gratuito. Também será preciso decidr *NAME* curto, mas ao mesmo tempo significativo. A seguir temos uma lista de itens que o visitante deve preencher, com um nome atribuído a cada um:

Nome = *fname*
Sobrenome = *lname*
Email = *email*

Agora coloque essa lista no documento da seguinte maneira:

```
Digite seu primeiro nome: <INPUT TYPE="text" NAME="fname" SIZE=46>
Seu sobrenome: <INPUT TYPE="text" NAME="lname" SIZE=46>
Seu email: <INPUT TYPE="text" NAME="fname" VALUE="login@provedor.com.br" SIZE=46>
```

Separar o nome do sobrenome em vez de colocar ambos no mesmo campo é uma boa idéia. Assim fica mais fácil manipulá-los separadamente ao criar o script. Observe o uso do atributo *VALUE*. Isso ajuda a dar ao visitante uma idéia do tipo de informação de que você precisa. Além disso, use o atributo *SIZE* para limitar o tamanho de cada campo em 46. Isso serve apenas para manipular a aparência do documento HTML.

Agora precisamos de um endereço de correspondência para enviar o produto ao cliente:

Endereço = *street*
Cidade = *city*
Estado = *state*
CEP = *zip*

Também podemos pedir o número de telefone, se o cliente não puder ser acessado via correio eletrônico:

Telefone Residencial = *hphone*
Telefone Comercial = *wphone*

Essas informações devem ser suficientes para a sua página Web. Claro, é possível acrescentar qualquer outra informação que atenda às suas necessidades específicas. Agora colocamos essas informações no documento:

```
Endereço: <INPUT TYPE="text" NAME="street" SIZE=46>
Cidade: <INPUT TYPE="text" NAME="city" SIZE=12>
Estado: <INPUT TYPE="text" NAME="state" SIZE=4>
CEP: <INPUT TYPE="text NAME="zip" SIZE=9>
Fone Res.: <INPUT TYPE="text" NAME="hphone" SIZE=13>
Fone Com.: <INPUT TYPE="text" NAME="wphone" SIZE=13>
<HR>
```

Apresentando as Opções do Produto

Agora é preciso apresentar aos visitantes a lista dos produtos que eles podem comprar. É provável que você necessite incluir também outras opções. Por exemplo, uma florista que vende rosas deveria dar ao cliente a opção de escolha de cor, tamanho etc.

No exemplo deste tutorial, estamos vendendo carros, logo convém dar ao consumidor várias opções de modelo, cor, ano e algumas características adicionais. Com relação ao modelo, oferecemos as opções Tortoise, Hyena, Aphid e Diamond Black:

modelo = *Tortoise*
Hyena
Aphid
Diamond Black

Neste exemplo, modelo *NAME* e Tortoise, Hyena, Aphid e Diamond Black são os valores. Pode-se deixar o usuário selecionar um modelo pelo marcador <*SELECT*> ou usar o tipo de input *radio*. Neste Tutorial, usaremos o marcador <*SELECT*>. Quanto às opções de cores, temos verde, roxo, marrom e azul:

Cor = *verda*
roxo
marrom
azul

A seguir oferecemos ao cliente as opções para os modelos dos anos de 1995, 1996, 1997 e 1998:

Ano = *1995*
1996
1997
1998

Para fornecer essas opções no código, o exemplo abaixo usa <*SELECT*> para modelo e ano e o tipo de input *radio* para cor:

```
Aqui está, sua chance de escolher alguns carros de alta qualidade!
Modelo/Ano: <SELECT NAME="model">
<OPTION> Tortoise
<OPTION> Hyena
<OPTION> Aphid
<OPTION> Diamond Black
</SELECT><SLECT NAME="year">
<OPTION> 1995
<OPTION> 1996
<OPTION> 1997
<OPTION> 1998
</SELECT>
Cor: </PRE>
<INPUT TYPE="radio" NAME="color" VALUE="Roxo"> Roxo
<INPUT TYPE="radio" NAME="color" VALUE="Verde"> Verde
<INPUT TYPE="radio" NAME="color" VALUE="Marrom"> Marrom
<INPUT TYPE="radio" NAME="color" VALUE="Azul"> Azul
```

Nota: Observe que o marcador <PRE> foi temporariamente removido para que os botões sejam exibidos lado a lado. Se fôssemos incluir o marcador <*PRE*>, seria preciso colocar todas as quatro cores

em uma linha, e como a largura do monitor é limitada, não conseguimos ver nem metade das informações. Sem o marcador <*PRE*>, as opções são colocadas lado a lado (o HTML não reconhece espaços).

Agora apresentamos as opções adicionais: airbag do lado do passageiro, desembaçador no vidro traseiro, calotas de plásticos e um estepe. Atribua a cada uma destas opções seu próprio valor. Use também o tipo de input checkbox para que, caso o cliente não selecione uma opção, você não tenha de se preocupar com isso dentro do script. O script simplesmente verifica se o usuário selecionou o checkbox e em caso positivo, reage do modo apropriado.

Airbag do lado do passageiro	= *iarbag*
Calotas de plástico	= *pcaps*
Desembaçador no vidro traseiro	= *rdefrost*
Estepe	= *stire*

A página HTML teria o seguinte aspecto:

```
<PRE>
<INPUT TYPE="checkbox" NAME="airbag">Airbag do lado do passageiro
<INPUT TYPE="checkbox" NAME="pcaps">Calotas de Plástico
<INPUT TYPE="checkbox" NAME="rdefrost">Desembaçador do Vidro traseiro
<INPUT TYPE="checkbox" NAME="stire">Estepe
```

Observe que cada nome descreve o que o valor realmente é. Ao examinar seu código, é possível determinar com segurança o que representa cada valor. Os nomes são curtos, mais auto-explicativos. agora permitiremos que o visitante escolha um modelo de duas, três ou quatro portas:

	2
Portas =	*3*
	4

Podemos usar o marcador <SELECT> ou tipo de input *radio*. Aqui usaremos novamente o tipo de input *radio* para deixar o visitante escolher uma das três opções:

```
<INPUT TYPE="radio" NAME="doors" VALUE="2"> 2 portas
<INPUT TYPE="radio" NAME="doors" VALUE="3"> 2 portas com bagageiro
<INPUT TYPE="radio" NAME="doors" VALUE="4"> 4 portas
```

Depois de receber as informações pessoais do cliente e sua escolha de carro, é preciso saber como ela vai pagar. Usando o nome *Pagamento*, damos ao cliente duas opções: Fatura e Cartão de Crédito.

Pagamento =	*Fatura*
	Cartão de Crédito

(Claro, os vendedores de automóveis não costumam simplesmente "mandar a conta" para os clientes. O exemplo usa essas opções pouco realistas apenas para demostrar como é possível oferecer opções diferentes aos consumidores.) Se o cliente escolher a opção cartão de crédito, será preciso obter também as informações sobre o cartão. A seguir, apresentamos três campos associados normalmente a cartões de crédito: o nome do cartão, o número e a data de validade.

Nome do Cartão	= *Cname*
Número do Cartão	= *Cnum*
Data de validade	= *cexpire*

Ao escrever o script, verifique se o cliente escolheu a opção de cartão de crédito; neste caso, é preciso incluir as informações sobre o cartão ao enviar o pedido. Agora, basta formatar as opção de pagamento usando o código HTML, conforme mostra o exemplo abaixo:

```
<HR>
Pagamento
<INPUT TYPE="radio" NAME="billing" VALUE="bill"> Fatura
<INPUT TYPE="radio" NAME="billing" VALUE="Cartão de Crédito"> Cartão de Crédito
Nome no Cartão: <INPUT TYPE="text" NAME="Cname" SIZE=46>
Número do Cartão: <INPUT TYPE="text" NAME="cnum" SIZE=14>
Validade do Cartão: <INPUT TYPE="text" NAME="xexpire" SIZE=8>
</PRE> <HR>
```

Enviando o Pedido

Finalmente, devemos permitir que o cliente nos envie as informações. Para isso basta usar o tipo de input *submit*; convém incluir também o tipo de input *reset* para o caso de o cliente precisar refazer o pedido.

```
Agora, se é isto que você quer pressione <INPUT TYPE="submit" VALUE="AQUI">
Ou se acha que cometeu um erro, <INPUT TYPE="reset" VALUE="Apague formulário">
</FORM>
</BODY>
</HTML>
```

Como não adicionaremos mais nada ao formulário, fechemos o marcador <FORM>, assim como <BODY> e <HTML>. Agora que terminamos, nosso formulário de pedidos deve estar bastante parecido com o exemplo dado inicialmente.

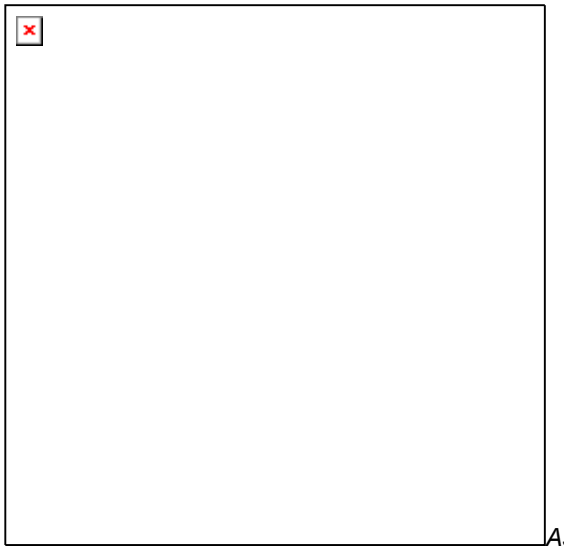
Claro, é possível acrescentar imagens gráficas, fundos diferentes ou links para outras páginas a fim de dar o formulário de pedidos um aspecto profissional. Pode-se até mesmo mudar a formatação HTML da página para que fique mais a seu gosto. Entretanto, para o propósito deste exemplo específico, o formulário está concluído. Agora é hora de trabalhar sobre o script.

O Script

Agora basta acessar as informações que o visitante digitou. Crie um script que envie essas informações pelo correio eletrônico para quem vai processá-las. Então informe ao visitante que seu pedido foi recebido com sucesso.

Determinando o Método

Primeiro é preciso descobrir se o script usou o método *POST* para receber as informações.



*As informações foram enviadas pelo método **POST** ?*
Se não, informe o visitante.
Se sim, continue.

Primeiro, indique ao script qual programa está sendo usado para interpretá-lo. Neste caso, estamos usando Perl. É preciso informar ao script o path e nome do arquivo do intérprete.

Nota: no UNIX, o modo mais fácil de encontrar o path de um arquivo é com os comandos *which* e *whereis*. Há muitos tipos de UNIX, portanto, se um destes comandos não funcionar, tente o outro.

Em seguida, defina o tiro *MIME* da saída do script.

```
#! /usr/local/bin/perl

If ($ENV{'REQUEST_METHOD'} NE 'POST')
{
print <<"HTML";
<HTML>
<HEAD>
<TITLE>Lamento!</TITLE>
</HEAD>
<BODY>
<H1>Uso de Método Errado!</H1>
Lamento, mas só utilizamos o método POST aqui.
</BODY>
</HTML>

HTML

exit;
}
```

Dividindo as Informações em Pares Nome/Valor

A seguir é preciso dividir as informações dos pares nome/valor. Siga o exemplo:

```
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
```

```
{
($name, $value) = split(/=/, $pair);
$value =~ tr/+/ /;
$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
$form{$name} = $value;
}
```

Primeiro leia o *STDIN* até o final do string, tal como é definido pelo uso de *$ENV{'CONTENT_LENGHT'}*. Em seguida, separe as linhas, dividindo a linha quando encontrar o E comercial (&), na lista @pairs. Neste ponto, temos vários pares nome/valor e substituir os sinais de adição pelos espaços apropriados. (Lembre-se de que, ao enviar o string de dados para o servidor, os espaços são substituídos pelo sinal de adição.) Então é preciso converter os caracteres especiais de volta a seu formato ASCII. Em seguida, coloque cada nome/valor na lista *$form*, para que então possamos chamar cada par nome/valor usando *$form{'name'}*.]

Manipulando os Elementos Que o Formulário Não Passa

Agora que terminamos a parte mais difícil do script, é preciso preparar o restante. O formulário não passou quatro itens de que o script necessita:

- A data.
- endereço de correio eletrônico da pessoa que recebe o pedido.
- path e nome do programa de correio.
- Uma linha de assunto para o correio eletrônico que informa ao destinatário que o formulário de pedido é para a compra de um carro. Essa linha ajuda o leitor a compreender a natureza do pedido sem ter de ler a mensagem inteira.

Obtendo a Data

Pode-se obter a data automaticamente usando a função *chop()* do Perl:

```
chop($date = `date`);
```

Este comando oferece como resultado algo parecido com isto:

```
Thu May 23 15:02:15 PDT 1998
```

É possível mudar o formato da string *$date*. Procure assunto referente a *date*.

Lidando com o Correio Eletrônico

Definiremos então as strings que determinam o destino da mensagem de correio eletrônico, o programa de correio e a linha de assunto:

```
$mailprog ="/usr/lib/sendmail";
$sendto = "rniles\@selah.net";
$subject = "Compra de automovel";
```

Agora, vamos enviar a mensagem. Primeiro use a função *open()* para enviar sua saída para sendmail:

```
open(MAIL, "|$mailprog -t") |
```

Esse comando simplesmente designa o descritor *MAIL* como um pipe para a string *$mailprog* definida anteriormente. O pipe altera *STDOUT*. Sem ele, todas as informações vão retornar para o visitante dentro de um documento HTML. Se mudarmos a direção do pipe, é possível enviar as informações para o programa UNIX sendmail.

As três linhas a seguir criam o cabeçalho da mensagem. A string *$sendto* definida acima determina a linha *To:*. A linha *From:* é uma combinação de três dos campos digitados pelo visitante. Usaremos o endereço de correio eletrônico do visitante, seu nome e sobrenome. Finalmente, acrescentamos o assunto.

Juntas, as três linhas ficam assim:

```
print MAIL "To: $sendto\n";
```

```
print MAIL "From: $form{'email'} ($form{'fname'} $form{'lname'})\n";
print MAIL "Subject: $subject\n\n";
```

Observe o uso de \n, que é alimentação de linha. Sem \n, tudo ficaria em uma só linha. Depois de imprimir *Subject:*; incluímos duas novas linhas que separam o cabeçalho do restante da mensagem.

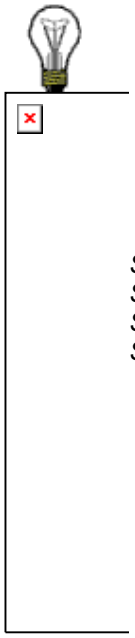
A seguir pedimos à senteça *print* para imprimir todo o conteúdo do descritor *MAIL* até encontrar *EOM*. Essa condição permite o uso da sentença *print* apenas uma vez para imprimir várias linhas.

```
print MAIL <<"EOM";
Em $date, $form{'fname'} $form{'lname'} decidiu comprar um carro conosco.
$form{'fname'} escolheu um $form{'model'} $form{'color'} $form{'year'} $form{'doors'}-porta(s).
$form{'fname'} escolheu os seguintes opcionais:
EOM
```

Observe como usamos as string do formulário para criar a sentença. Basta inserir as strings no texto, de modo que leas ajudem a criar a mensagem. Dependendo do que o cliente digitou no formulário, a mensagem pode parecer com:

```
Em Thu Jul 1 11:59:42 EST 1998, Eduardo Mota decidiu comprar um carro conosco.
Eduardo escolheu um Tortoise verde 1995 2 portas(s).
Eduardo queria os seguintes opcionais:
```

A seguir vamos analisar algumas sentenças **if( )**. O script diz essencialmente que:



*Se existe **airbag**, então imprima* Airbag do lado do passageiro.
*Se existe **pcaps**, então imprima* Calotas de plástico.
*Se existe **rdefrost**, então imprima* Desembaçador no vrido traseiro
*Se existe **stire**, então imprima* Estepe.

Com Perl codificamos essas instruções **if( )** da seguinte maneira:

```
if ($form{'airbag'})
    {
    print MAIL "- Airbag para o passageiro\n";
    }
if ($form{'pcaps'})
    {
    print MAIL "- Calotas de plastico\n";
```

```
    }
if ($form{'rdefrost'})
    {
    print MAIL "- Desembacador do vidro traseiro\n";
    }
if ($form{'stire'})
    {
    print MAIL "- Estepe\n";
    }
```

Então, para economizar trabalho, digitamos novamente o seguinte comando:

```
print MAIL <<"EOM";
```

Esse comando permite que usemos o comando *print* apenas uma vez. Veja se consegue descobrir o que acontece no exemplo abaixo:

```
Seguem algumas informacoes pessoais de $form{'fname'}:
$form{'fname'} $form{'lname'}
$form{'email'}
$form{'street'}
$form{'city'}, $form{'state'} $form{'zip'}

Fone res.: $form{'hphone'}
Fone com.: $form{'wphone'}

$form{'fname'} pediu para que nos:

EOM
```

Definindo as Opções de Pagamento do Cliente

Agora, precisamos determinar qual forma de pagamento o cliente escolheu. Novamente usaremos a instrução **if( )**, assim:

Verifique se o cliente escolheu Fatura.
Se sim, imprima Fatura.
Verifique se o cliente escolheu Cartão de Crédito.
Se sim, imprima Cartão de Crédito.

Como o formulário usa o tipo de input *radio*, o cliente precisa selecionar Fatura ou Cartão de Crédito. Basta marcar qual dessas opções o cliente escolheu.

```
if ($form{'billing'} eq "bill")
    {
    print MAIL "Mandaremos a fatura.\n";
    }
if ($form{'billing'} eq "ccard")
    {
    print MAIL "Cobraremos pelo cartao de credito";
    print MAIL "em nome de $form{'cname'}.\n";
    print MAIL "CC\#$form{'cnum'} e validade $form{'cexpire'}.\n\n";
    }
```

Agora, sempre que abrimos um descritor de arquivo, será preciso fechá-lo. Abrimos o descritor de arquivo *MAIL*, portanto ao terminar de usá-lo é preciso usar a função *close()* para fechá-lo:

```
print MAIL "Processar a requisição o mais rápido possivel.\n";
close(MAIL);
```

Depois de fechar o descritor, eviamos a mensagem.

Enviando uma Mensagem de Resposta ao Cliente

Neste ponto, poderíamos considerar o script concluído. Ele recebe as informações do cliente e envia o pedido via correio eletrônico. Entretanto, a finalidade principal de um script CGI é dar uma resposta nova ao visitante. Neste caso, você pode simplesmente querer dizer ao cliente que o programa processou o pedido normalmente e que ele pode prosseguir.

Nesta seção, criaremos rapidamente uma página que apresenta essa mensagem. Basta adicionar ao seu script o código que gera essa página. Depois de enviar a mensagem do pedido, o script manda uma mensagem para o cliente.

Primeiro é preciso indicar ao servidor o tipo **MIME** do script. Apesar de o script normalmente usar *text/html*, também é possível usar *text/plain*; se estivermos mandando um arquivo .GIF, usaremos *image/gif*. Há outras opções, mas estas são as mais usadas.

Usaremos agora o atalho mostrado no exemplo abaixo:

```
print <<"HTML";
<HTML>
<HEAD>
<TITLE>Agradecimento</TITLE>
</HEAD>

<BODY>
<H1> Obrigado pela requisição! </H1>

Obrigado pela requisição <B>$form{'fname'}</B>! Esta mensagem foi enviada para o departamento apropriado e nós iremos entregar o seu novo $form{'color'} $form{'year'} $form{'model'} $form{'doors'}-porta(s) assim que verificarmos o seu crédito com um pente fino.<P>
Se não entrarmos em contato nos próximos 8 meses, envie email para <A HREF=\"mailto:$sendto\">$sendto</A>, e pergunte por que seu pedido não foi atendido
rapidamente</B> como esperado. <P>
Foi muito bom tê-lo como cliente!!<P>
<I>Atenciosamente,</I><BR>
Sr. Supkay (ele mesmo).
</BODY>
</HTML>

HTML
```

Nota: Observe a barra invertida (/) na linha *mailto:*. Para garantir que você não conmfunda o interpretador Perl, às vezes é melhor considerar as aspas como caracteres especiais usando a barra

invertida. Então o interpretador Perl saberá que as aspas fazem parte do texto exibido e não do código Perl.

Finalmente saímos do programa usando a função:

```
exit()
```

Resumo

Apesar de este tutorial ter apresentado um exemplo de formulário de pedido, de maneira alguma esse exemplo esgota as possibilidades do script. É possível fazer com que o script salve essas informações em um arquivo de log ou até mesmo usar um banco de dados para armazenar as informações. As possibilidades são infinitas.

Neste Tutorial aprendemos a criar documentos HTML usando formulário e também a criar um script que permite que essas informações sejam enviadas para outros usuários. Essencialmente, criamos uma interface Web para enviar informações a outra pessoa via correio eletrônico. Com apenas algumas modificações, poderíamos criar um scrit que permitisse ao visitante usar o formulário para deixar mensagens para você. Também vimos o conceito mais importante da programação CGI: como enviar ao visitante um documento imediatamente criado*.

* N.R.T. Esses documentos criados pelo próprio script são chamados e documentos virtuais.

Créditos & Autoria

Este texto criado por: **Robert Niles** e **Jeffry Dwight**
Adaptado e Editado por: Eduardo Mota