Lazarus e CGI: Criando uma janela de login - Parte 2.

Silvio Clécio - 19/07/2010

Dando continuidade ao artigo anterior, neste iremos aprender como criar uma janela de login no *Lazarus*, usando um projeto do tipo *CGI Aplication*, e aproveitar um *template* gratuito. Também iremos coletar os dados digitados na janela de login. Vamos lá!

Introdução

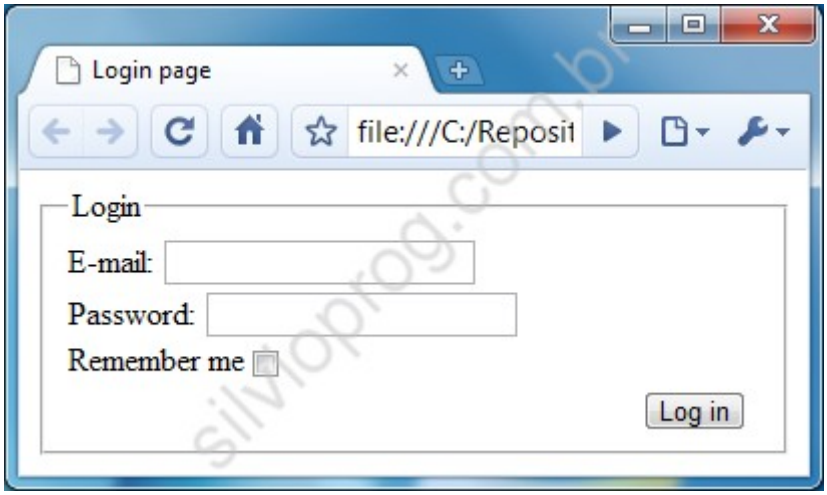
Se você não tem algum conhecimento sobre *CGI*, *HTML* e *CSS*, por favor, antes de prosseguir a leitura do artigo, leia o material disponível nos seguintes links:

- CGI - http://pt.wikipedia.org/wiki/CGI
- HTML - http://pt.wikipedia.org/wiki/HTML
- CSS - http://pt.wikipedia.org/wiki/Cascading_Style_Sheets

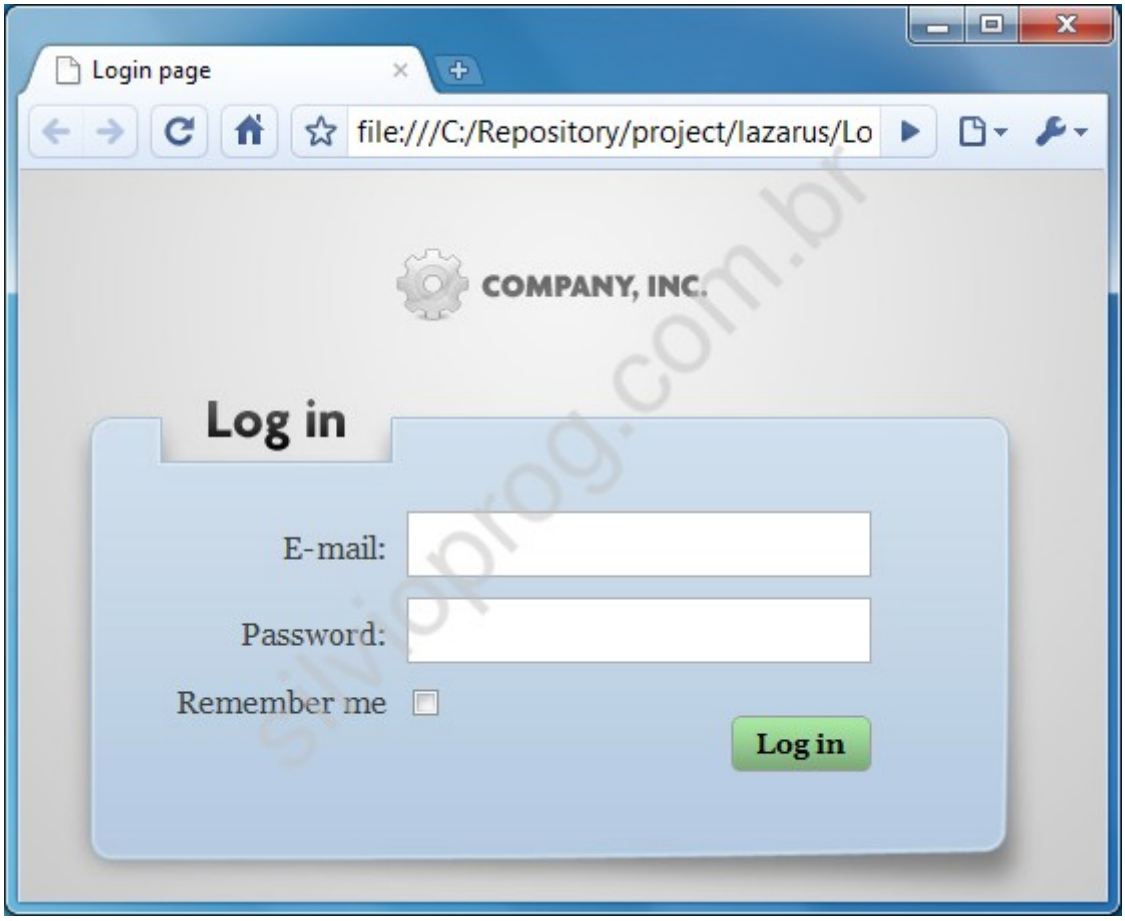
Faça suas pesquisas pessoais principalmente sobre *HTML* e *CSS*, pois será importante para você entender perfeitamente mais artigos que virão sobre *Lazarus* e *CGI*.

Aproveitando um template gratuito

Sem nenhuma formatação prévia, a nossa janela terá a seguinte estrutura e aparência (*HTML* puro):



Logo após aplicar a folha de estilo (*HTML + CSS*):



Legal heim? Você poderá usar essa janela tranquilamente, ela é gratuita, encontrei numa rápida busca que fiz ao *Google*. Opte sempre em procurar *templates* gratuitos prontos na web, com isso você poderá poupar muito tempo. Veja como é simples o código fonte de nosso *template*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <title>Login page</title>
         <link rel="stylesheet" type="text/css" href="css/login.css" />
</head>
<body>
   <form id="login-form" action="#" method="post">
   <fieldset>
      <legend>Login</legend>

         <label for="login">E-mail: </label>
            <input type="text" id="login" name="login"/>
               <div class="clear"></div>
                  <label for="password">Password: </label>
                  <input type="password" id="password" name="password"/>
                  <div class="clear"></div>
                  <label for="remember_me" style="padding: 0;">Remember me</label>

                  <input type="checkbox" id="remember_me" style="position: relative; top: 3px; margin: 0; "
name="remember_me"/>
                  <div class="clear"></div>
```

```
                <br />
                <input type="submit" style="margin: -20px 0 0 287px;" class="button" name="commit" value="Log
in"/>
    </fieldset>
</form>
</body>
</html>
```

Note que editei somente a parte action="#" para action="/cgi-bin/login.cgi/loginaction".

Código do arquivo *login.css*:

```
* { margin: 0; padding: 0; }

body { font-family: Georgia, serif; background: url(images/login-page-bg.jpg)
    top center no-repeat #c4c4c4; color: #3a3a3a; }

.clear { clear: both; }

form { width: 406px; margin: 170px auto 0; }

legend { display: none; }

fieldset { border: 0; }

label { width: 115px; text-align: right; float: left; margin: 0 10px 0 0;
    padding: 9px 0 0 0; font-size: 16px; }

input  { width: 220px; display: block; padding: 4px; margin: 0 0 10px 0;

font-size: 18px; color: #3a3a3a; font-family: Georgia, serif; }

input[type=checkbox]{ width: 20px; margin: 0; display: inline-block; }

.button { background: url(images/button-bg.png) repeat-x top center;
    border: 1px solid #999; -moz-border-radius: 5px; padding: 5px;
    color: black; font-weight: bold; -webkit-border-radius: 5px;
    font-size: 13px;  width: 70px; }

.button:hover { background: white; color: black; }
```

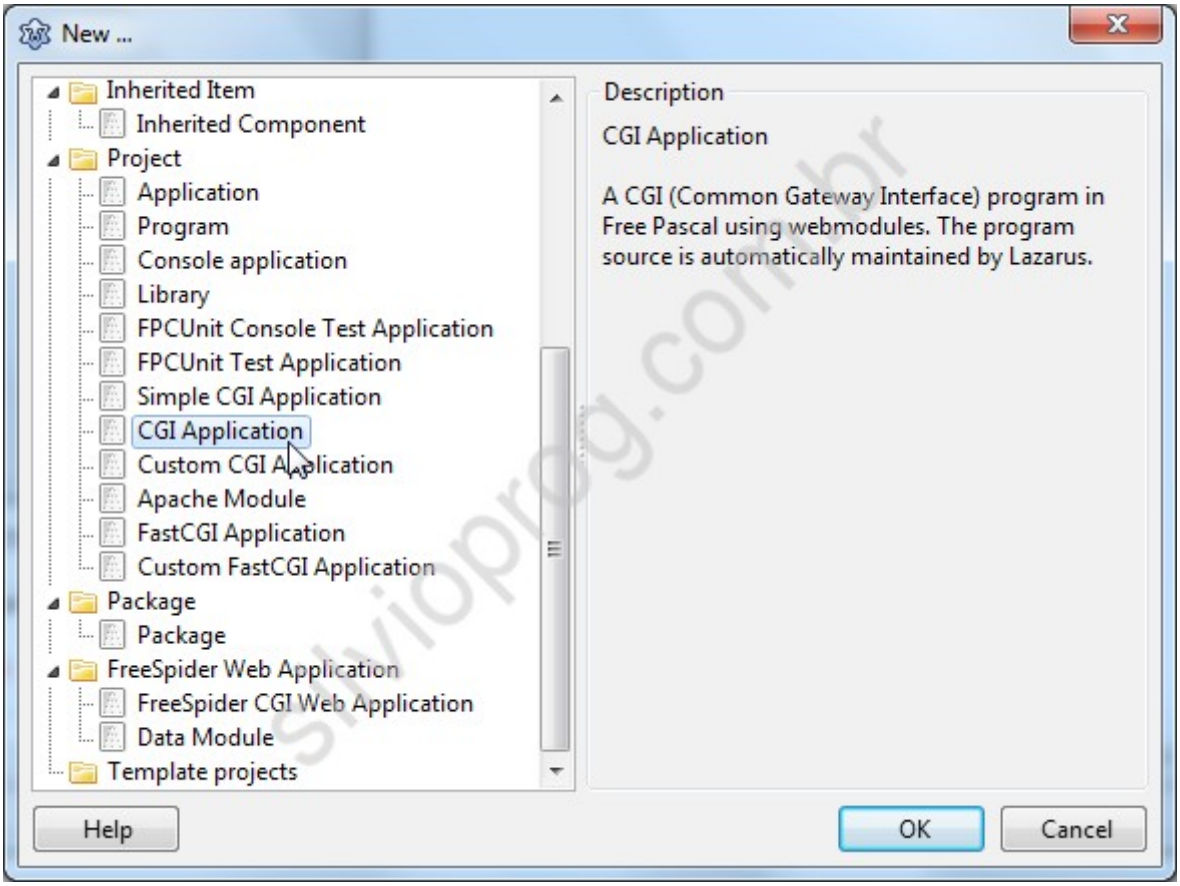
O código acima é que define a aparência da página, que é a nossa janela de login. Não entrarei em detalhes no código acima para não tornar o artigo muito longo, mas veja sobre *CSS* na web, atualmente existe muito material didático disponível.

Nota:
Apesar do *WebLaz* dar suporte a *templates*, para um exemplo tão simples eu optei em usar maior parte do código em *HTML* dentro de arquivos *.inc*, e pequenas partes nas *units*, deixando em arquivos externos somente a imagem de fundo em formato *PNG* e os arquivos *CSS* para melhorar a aparência da janela.

No código em *HTML* acima, os pontos mais importante são nossos componentes (em linguagem *HTML* chamamos de *campos*), que são *login*, *password* e *remember_me*. É deles que iremos capturar os dados informados na janela de login.

Criando o projeto do tipo CGI Application

Abra o *Lazarus*, vá ao menu *File* | *New ...* e escolha a opção *CGI Application*:



Renomeie o módulo web *FPWebModule1* para *MainFPWebModule*. Provavelmente você receberá a seguinte mensagem de erro quando tentar renomear:

---

unit1.pas(25,3) Error: expected end., but RegisterHTTPModule found

---

Não se desespere, isso é um bug presente no *Lazarus*. :-) Não posso garantir quando isso será resolvido, mas para nós, precisamos apenas adicionar a diretiva *initialization*, veja no código que segue:

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, HTTPDefs, websession, fpHTTP, fpWeb;

type
  TFPWebModule1 = class(TFPWebModule)
  private
    { private declarations }
  public
    { public declarations }
  end;
```

```
var
  FPWebModule1: TFPWebModule1;

implementation

{$R *.lfm}

initialization
  RegisterHTTPModule('TFPWebModule1', TFPWebModule1);

end.
```

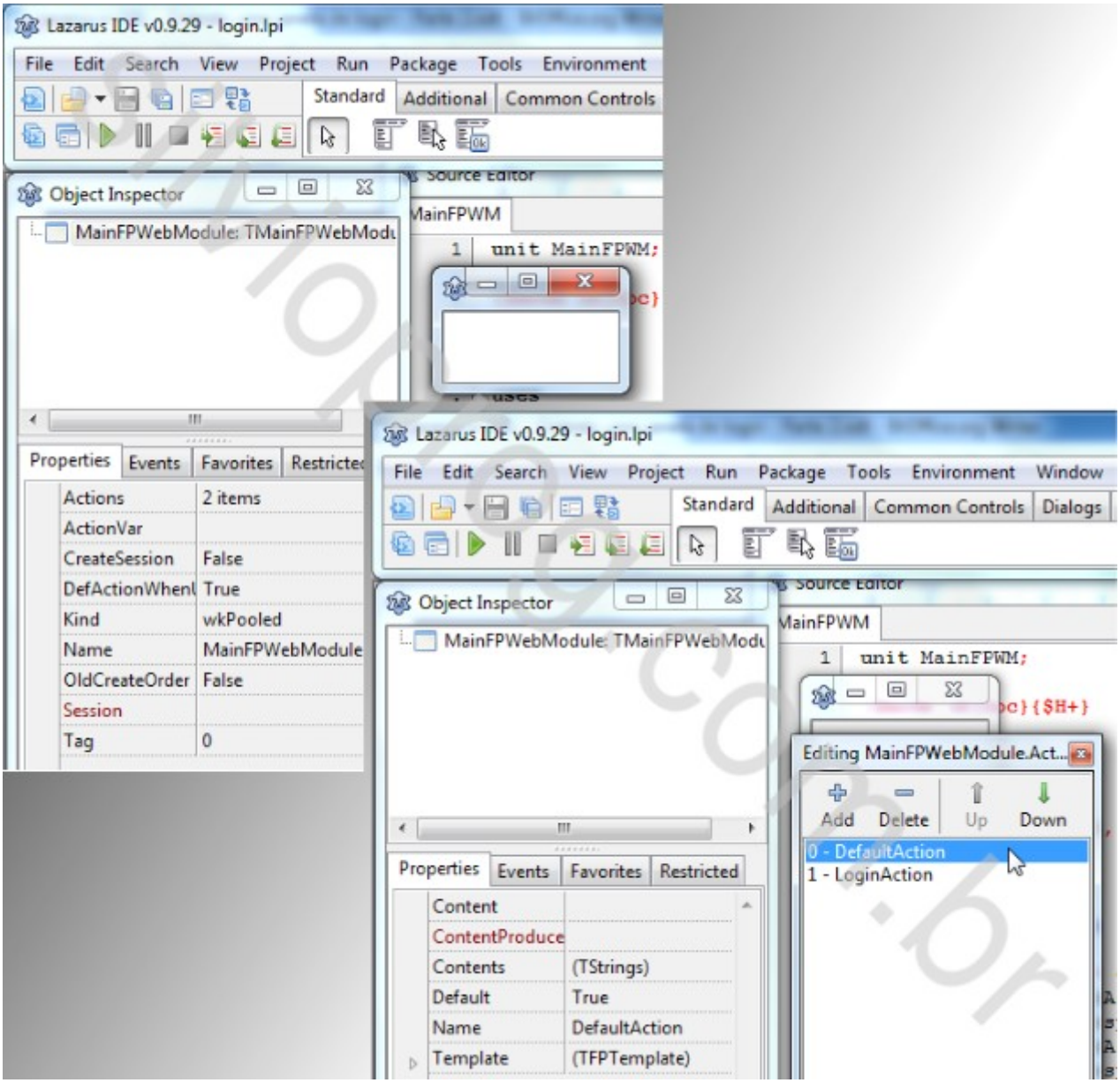
Devido outro bug do *Lazarus*, defina manualmente a antepenúltima linha da seguinte forma:

```
RegisterHTTPModule('TMainFPWebModule', TMainFPWebModule);
```

Feito isso, escolha um local e salve o projeto como *login.lpi*, no meu caso foi em:

C:\Repository\project\lazarus\LoginCGI\login.lpi

Salve a *unit* principal como *MainFPWM.pas*. Logo depois, clique em *Actions* e adicione duas a*ctions*, renomeando uma como *DefaultAction* e a outra como *LoginAction*. Na primeira, *DefaultAction*, defina a propriedade *Default* para *True*. Veja como ficará, na imagem seguinte:



Dica:

Procure usar *Shift+Ctrl+S* sempre que alterar algo em seus códigos, isso garante que você não perca seu trabalho numa eventual falha do sistema.

Criando os arquivos de inclusão

Com o projeto aberto no *Lazarus*, vá ao menu *File | New*, na janela que abre escolha *Text*, e salve como *customheader.inc*. Neste arquivo, cole o seguinte código:

```
'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">' + sLineBreak +
'<html xmlns="http://www.w3.org/1999/xhtml">' + sLineBreak +
'<head>' + sLineBreak +
'  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />' + sLineBreak +
'    <title>%s</title>' + sLineBreak +
'      <link rel="stylesheet" type="text/css" href="%s" />' + sLineBreak +
'</head>' + sLineBreak +
'<body>' + sLineBreak +
'  %s' + sLineBreak +
'</body>' + sLineBreak +
```

```
'</html>';
```

Repita o passo acima, mas salvando o segundo arquivo como *logincontent.inc*, definindo o seguinte código nele:

```
'<form id="login-form" action="/cgi-bin/%s/loginaction" method="post">' + sLineBreak +
'  <fieldset>' + sLineBreak +
'    <legend>Login</legend>' + sLineBreak +
'      <label for="login">E-mail: </label>' + sLineBreak +
'        <input type="text" id="login" name="login"/>' + sLineBreak +
'          <div class="clear"></div>' + sLineBreak +
'            <label for="password">Password: </label>' + sLineBreak +
'            <input type="password" id="password" name="password"/>' + sLineBreak +
'            <div class="clear"></div>' + sLineBreak +
'            <label for="remember_me" style="padding: 0;">Remember me</label>' + sLineBreak +
'            <input type="checkbox" id="remember_me" style="position: relative; top: 3px; margin: 0; "
name="remember_me"/>' + sLineBreak +
'          <div class="clear"></div>' + sLineBreak +
'          <br />' + sLineBreak +
'          <input type="submit" style="margin: -20px 0 0 287px;" class="button" name="commit" value="Log in"/>' +
sLineBreak +
'  </fieldset>' + sLineBreak +
'</form>';
```

Agora já temos todos os arquivos necessários para fazer nossa janela de login funcionar. Não compile ainda, pois iremos definir o nome do arquivo a ser gerado no decorrer do artigo. Note que eu aproveitei todo o código do *template* que encontrei, e fiz pequenas alterações para adaptar a minha necessidade. As partes sLineBreak são apenas quebra de linhas em *Object Pascal*.

Implementando o código em Object Pascal

Na seção public de nossa *unit*, defina as seguintes funções ...:

```
public
  function Header(const AStyle: string; const ATitle: string = '';
    const ABody: string = ''): string;
  function LoginContent(const APath: string = ''): string;
```

... use *Shift+Ctrl+C* e implemente o seguinte código:

```
function TMainFPWebModule.Header(const AStyle: string; const ATitle: string;
  const ABody: string): string;
begin
  Result := Format(CCustomHeader, [ATitle, AStyle, ABody]);
end;

function TMainFPWebModule.LoginContent(const APath: string): string;
begin
  if APath = '' then
    Result := Format(CLoginContent, [ExtractFileName(ParamStr(0))])
  else
    Result := Format(CLoginContent, [APath]);
end;
```

Agora, clique novamente na propriedade *Actions*, e na *action DefaultAction* implemente o seguinte código em seu evento *OnRequest*:

```
AResponse.Content := Header('/css/login.css', 'Login page', LoginContent);
Handled := True;
```

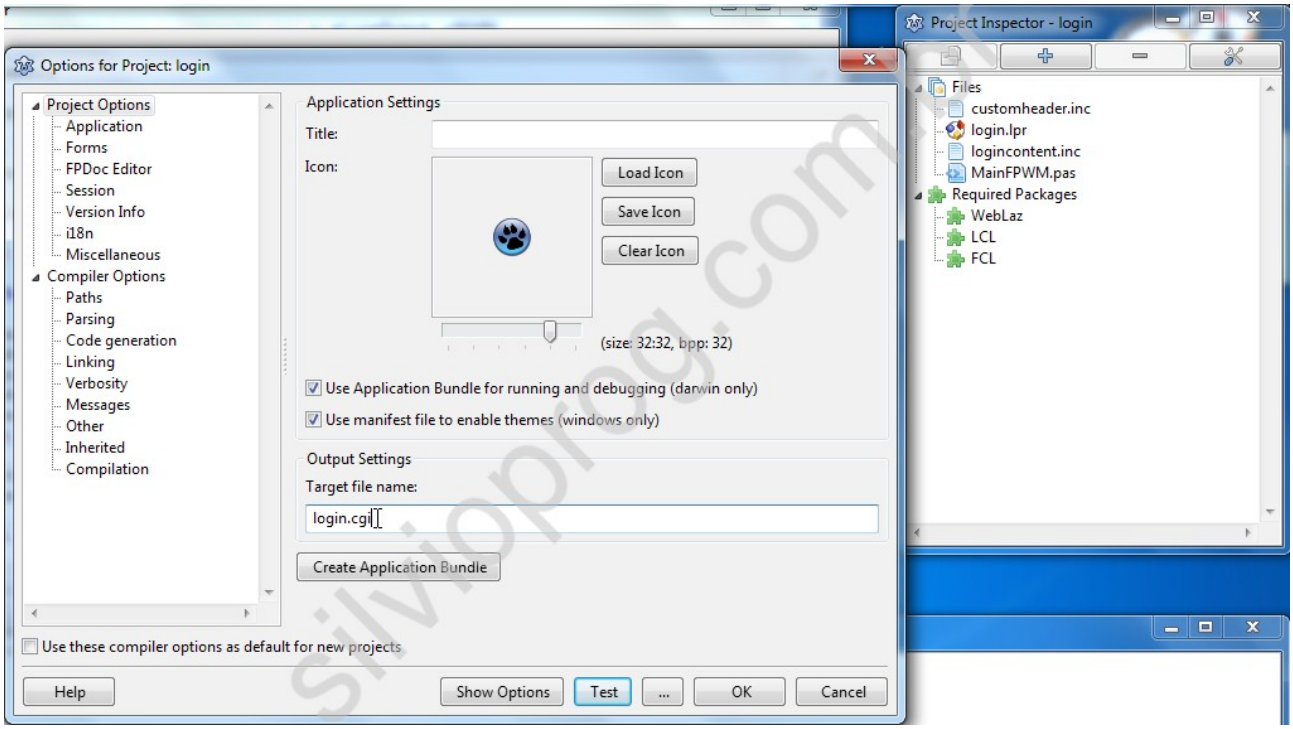
Repita o mesmo procedimento para a segunda *action*, e no mesmo evento desta segunda, implemente:

```
var
  VLoginResult: string;
begin
  VLoginResult := '<h1>Login successfully! :)</h1><br />' + sLineBreak +
    'E-mail: ' + ARequest.ContentFields.Values['login'] + '<br />' + sLineBreak +
    'Password: ' + ARequest.ContentFields.Values['password']  + '<br />' + sLineBreak +
    'Remember me: ' + ARequest.ContentFields.Values['remember_me'];
  AResponse.Content := Header('/css/logged.css', 'Logged', VLoginResult);
  Handled := True;
end;
```

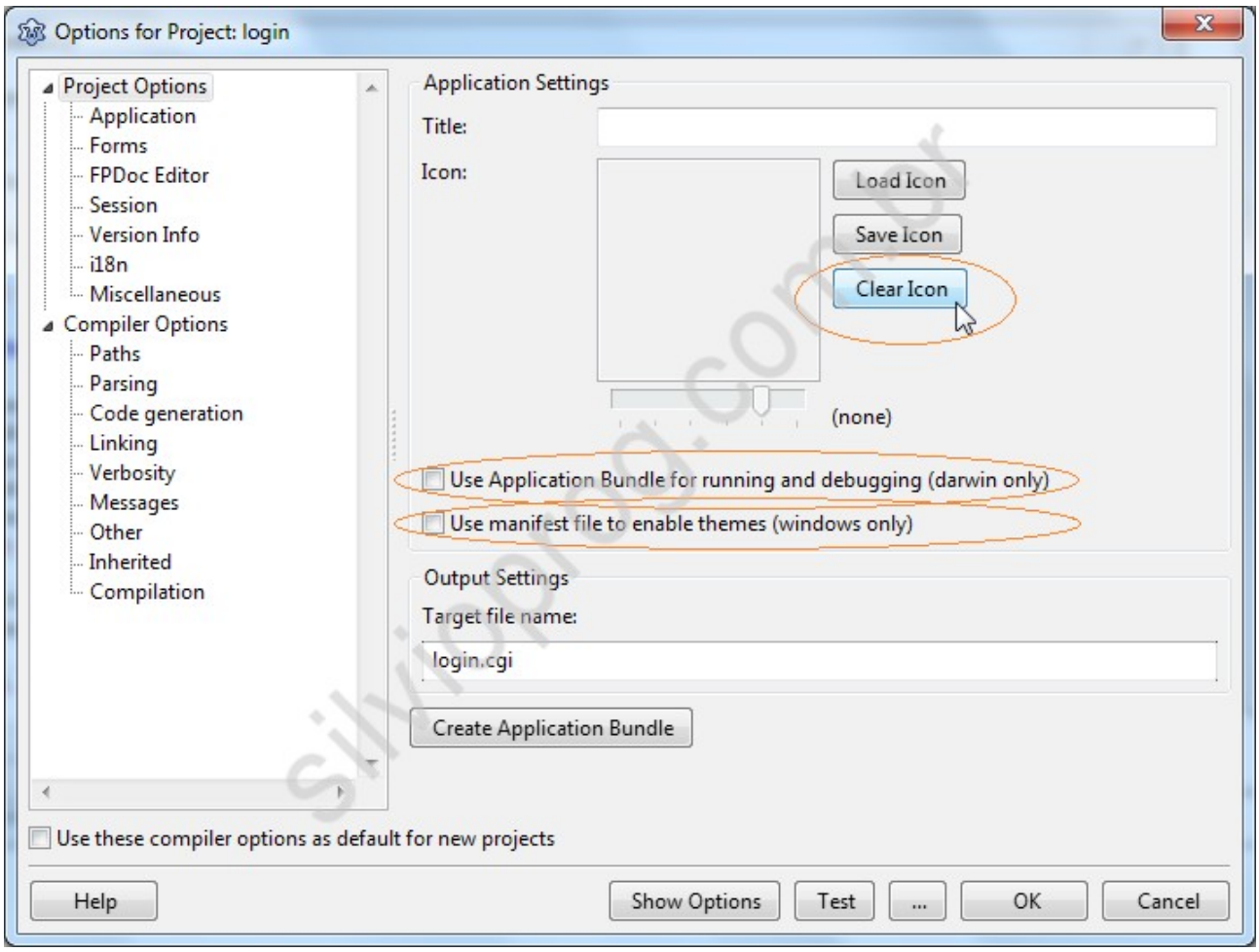
Logo acima de **implementation**, defina as seguintes constantes:

```
const
  CCustomHeader = {$I customheader.inc}
  CLoginContent = {$I logincontent.inc}
```

Por fim, acesse o menu *Project* | *Project Inspector*, clique no botão *Options*, e na janela que abre, na opção *Taget file name* defina *login.cgi*, veja detalhadamente na figura a seguir:



Eu costumo remover o que não irei utilizar, com isso, removi o ícone do *resource* e desativei as seguintes opções antes de clicar em OK:



Finalmente, depois de muito diálogo, dicas e código, compila tudo meu filho! :-) Mas somente compile, usando *Ctrl+F9*, não use F9 para rodar pois não é um aplicativo executável.

Para finalizar esta parte do artigo, veja o código completo a seguir:

```
unit MainFPWM;

{$mode objfpc}{$H+}

interface

uses
  SysUtils, HTTPDefs, fpHTTP, fpWeb;

type

  { TMainFPWebModule }

  TMainFPWebModule = class(TFPWebModule)
    procedure TFPWebActions0Request(Sender: TObject; ARequest: TRequest;
      AResponse: TResponse; Var Handled: Boolean);
    procedure TFPWebActions1Request(Sender: TObject; ARequest: TRequest;
      AResponse: TResponse; Var Handled: Boolean);
  public
    function Header(const AStyle: string; const ATitle: string = '';
      const ABody: string = ''): string;
    function LoginContent(const APath: string = ''): string;
  end;
```

```
const
  CCustomHeader = {$I customheader.inc}
  CLoginContent = {$I logincontent.inc}

implementation

{$R *.lfm}

{ TMainFPWebModule }

procedure TMainFPWebModule.TFPWebActions0Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; Var Handled: Boolean);
begin
  AResponse.Content := Header('/css/login.css', 'Login page', LoginContent);
  Handled := True;
end;

procedure TMainFPWebModule.TFPWebActions1Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; Var Handled: Boolean);
var
  VLoginResult: string;
begin
  VLoginResult := '<h1>Login successfully! :)</h1><br />' + sLineBreak +
    'E-mail: ' + ARequest.ContentFields.Values['login'] + '<br />' + sLineBreak +
    'Password: ' + ARequest.ContentFields.Values['password']  + '<br />' + sLineBreak +
    'Remember me: ' + ARequest.ContentFields.Values['remember_me'];
  AResponse.Content := Header('/css/logged.css', 'Logged', VLoginResult);
  Handled := True;
end;

function TMainFPWebModule.Header(const AStyle: string; const ATitle: string;
  const ABody: string): string;
begin
  Result := Format(CCustomHeader, [ATitle, AStyle, ABody]);
end;

function TMainFPWebModule.LoginContent(const APath: string): string;
begin
  if APath = '' then
    Result := Format(CLoginContent, [ExtractFileName(ParamStr(0))])
  else
    Result := Format(CLoginContent, [APath]);
end;

initialization
  RegisterHTTPModule('TMainFPWebModule', TMainFPWebModule);

end.
```

Copiando os arquivos para os diretórios do Apache

No diretório de nosso projeto foi gerado um módulo *CGI*, supondo que você esteja no *Windows* e seguiu atentamente **nosso artigo anterior**, copie o arquivo *login.cgi* para a seguinte pasta:

C:\Program Files (x86)\Apache Software Foundation\Apache2.2\cgi-bin

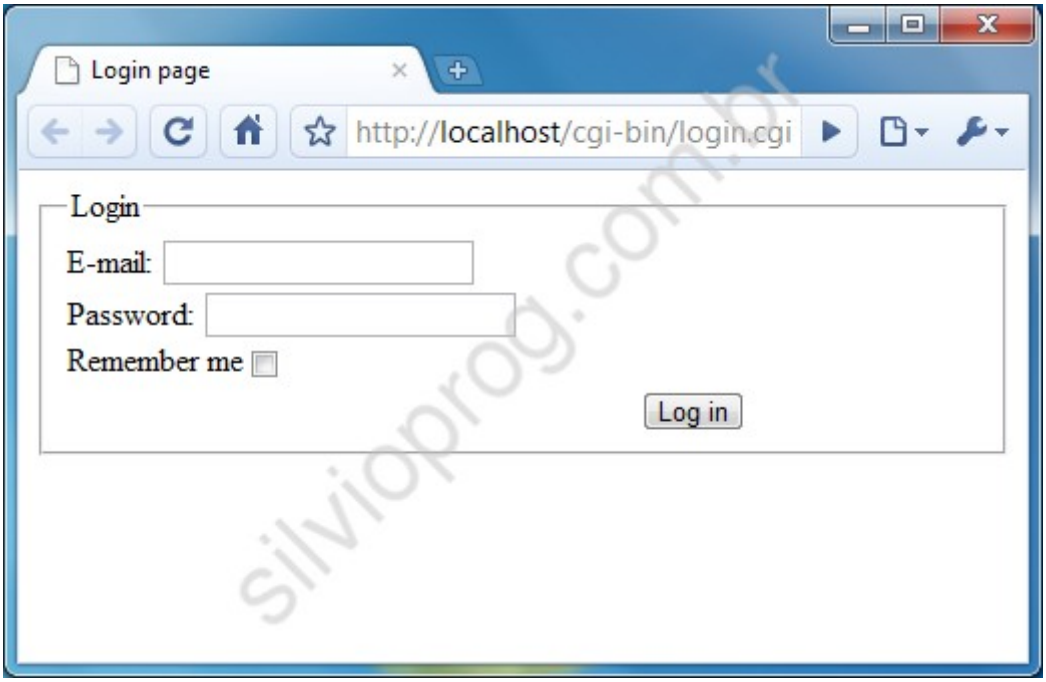
No *Linux* é bem simples, crie apenas uma ligação simbólica apontando para o módulo; Supondo que seu projeto esteja em em */home/seunomedeusuario/LoginCGI*, abra o terminal e digite:

```
$ sudo ln -s '/home/seunomedeusuario/LoginCGI/login.cgi' /usr/lib/cgi-bin
```

Após isso, certifique-se que seu *Apache* esta rodando digitando *http://localhost* em seu browser e recebendo a mensagem **It works!**, e logo depois, acesse o seguinte link que aponta para nosso módulo *login.cgi*:

http://localhost/cgi-bin/login.cgi

Deverá aparecer a nossa janela de login, porém com aparência padrão:



Silvio, e como faço para aparecer aquela janelinha bonitinha que você mostrou antes?

Simples, copie o pacote disponível **aqui**, descompacte e copie a pasta *css* para o seguinte endereço:

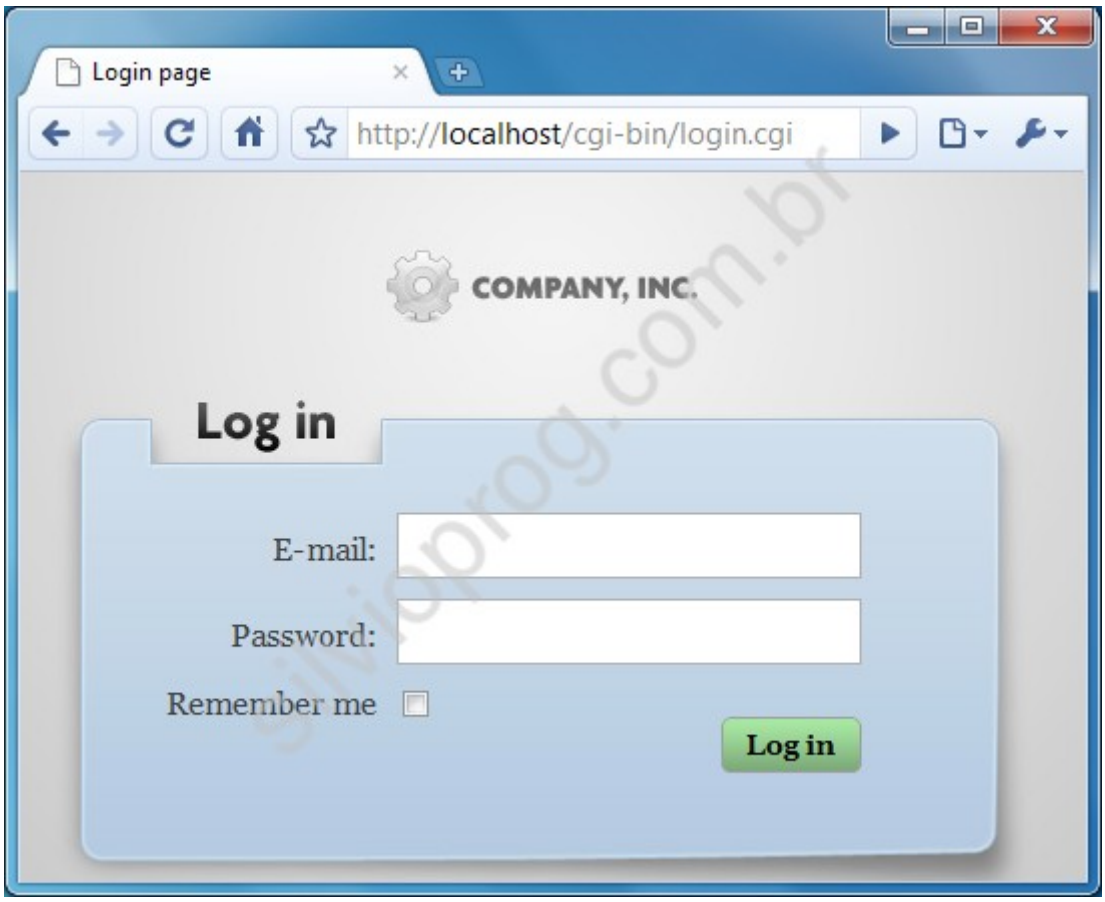
C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs

No *Linux* (seu usuário precisará de permissões de escrita):

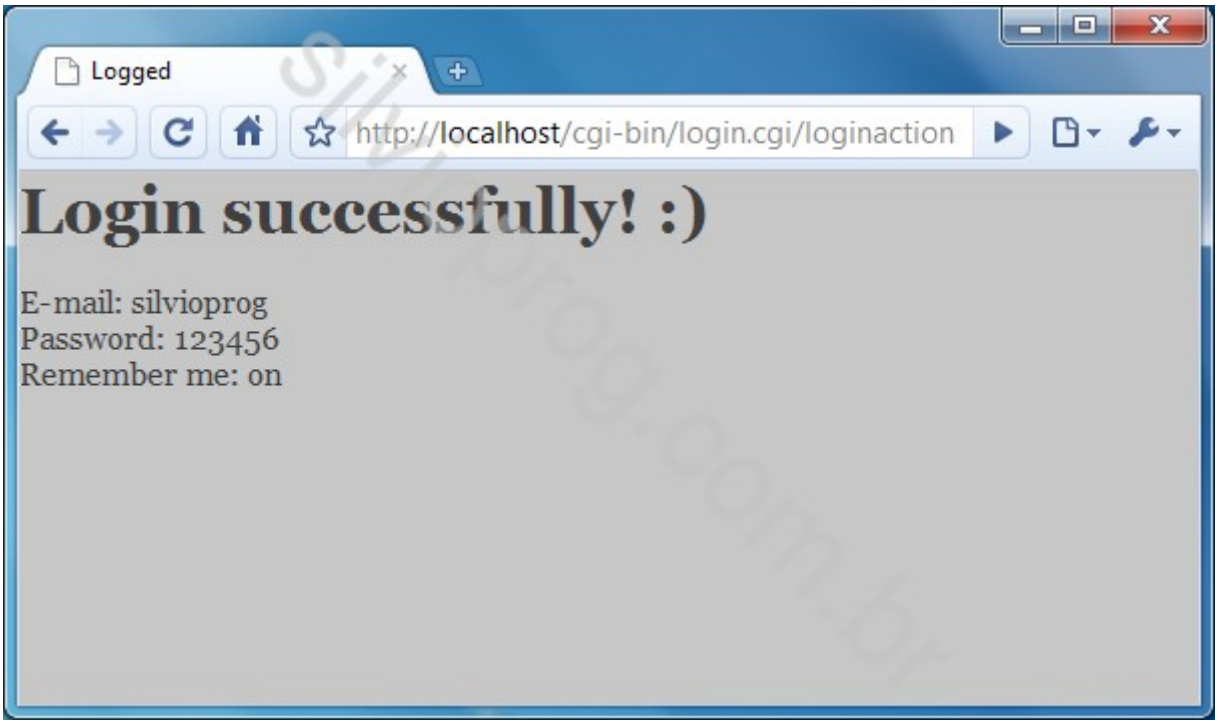
/var/www

Note que nele já existe o arquivo *index.html*, é nele que esta definida a mensagem **It works!** que recebemos ao informar *localhost*. É nesse diretório onde ficarão todos os arquivos *CSS*, *HTML*, *PNG* e etc. de seu site. Também será assim quando você for hospedar seus módulos em algum site, e provavelmente algum técnico da empresa que te fornecer o serviço irá cuidar disso para você, incluindo configurações no próprio *Apache*, caso necessário, claro! *(Caso algum site de hospedagem queira ser citdo em meus artigos, basta patrocinar os mesmos ;-))*.

Agora que você copiou a pasta *css* para o endereço indicado acima, atualize a página em seu navegador (*F5*), veja como a janela ficou com uma aparência excelente:



Informe um login e senha nela e tecle *Enter*, você receberá o que digitou na janela seguinte:



Comentários sobre os códigos em Object Pascal

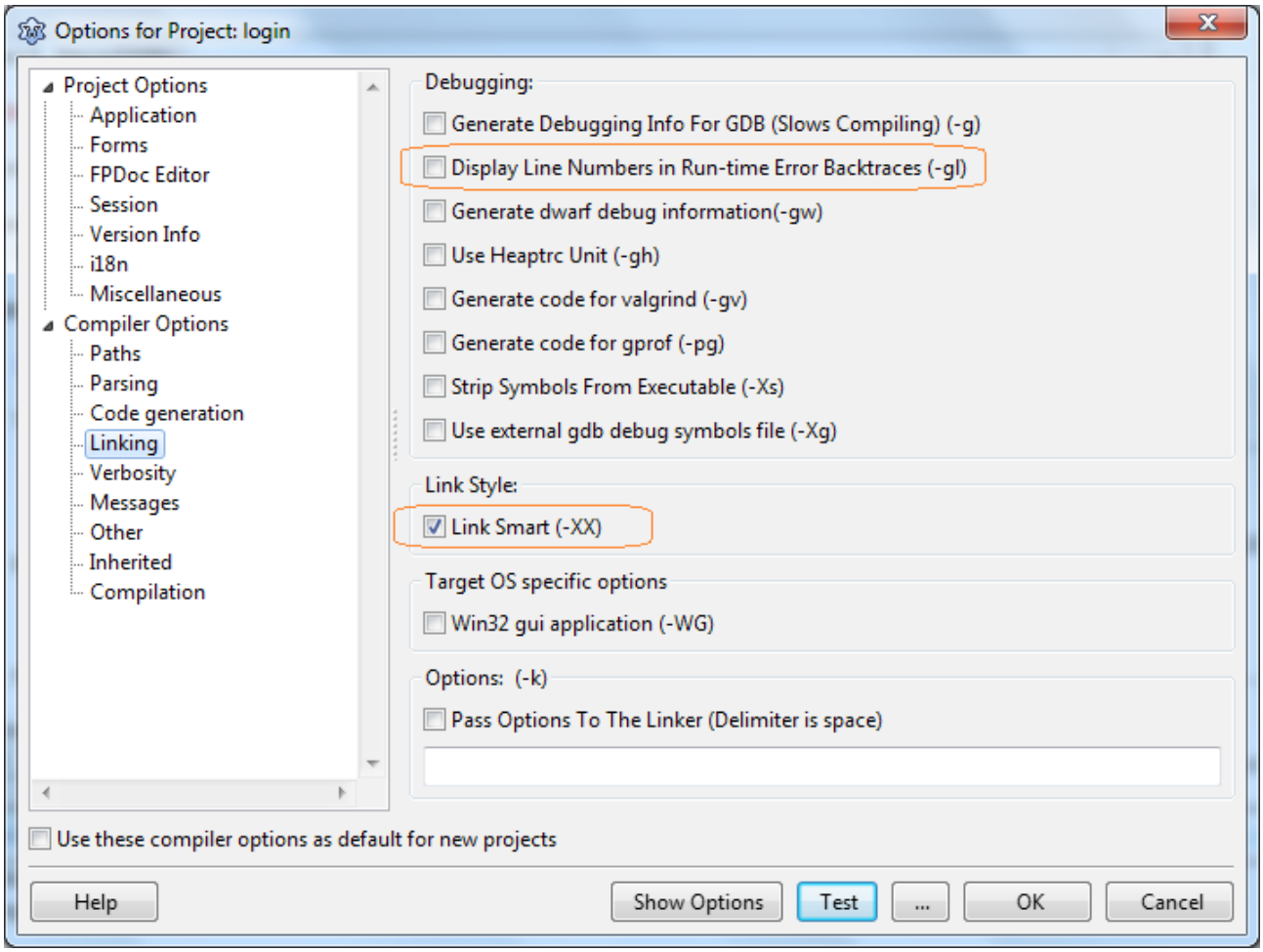
Na função Header eu chamei o código do arquivo *customheader.inc*, e preenchi as *tags* do *HTML* contido nele através dos parâmetros ATitle, AStyle, ABody da função, montando assim o nosso *HTML*

completo. Na função *LoginContent* fiz algo semelhante, mas adicionei uma condição para que, caso não seja informada nenhuma *action* (no nosso caso *loginaction*) no parâmetro APath, chamar a *action default*. No restante do código usei as duas funções, onde no *OnRequest* de *DefaultAction* usei a Header para exibição da janela de login, e no *OnRequest* da segunda *action* coletei os dados da janela através dos *fields* login, password e remember_me, "formatando" então a variável VLoginResult, chamando a função Header novamente e enviado o código *HTML* formatado para AResponse.Content.

Em um projeto com código maior, é interessante separar tudo isso em *units*, e caso note que irá aproveitar várias partes, crie *customs* de funções e de classes para ficar mais fácil um possível reaproveitamento, mas não exagere. Você verá na prática sobre isso no último artigo dessa série.

Cara, não gostei, para uma janelinha tão simples o binário ficou com 458KB!

Concordo, mas esquecemos de avisar ao *Free Pascal* para não guardar arquivos de *debug* no binário, com isso, vá ao menu *Project* | *Project Inspector*, na janela que abrirá vá a guia *Linking*, e configure conforme a imagem a seguir (logo depois compile tudo novamente):



Dúvidas frequentes

O CGI gerado no Windows roda no Linux, ou vice-versa?

Já aconteceu comigo, compilei um teste simples no *Linux* e rodei no *Windows*, mas, isso sempre te deixará numa incógnita, com isso, compile o seu *CGI* para o ambiente certo, uma vez que o *Lazarus* e *Free Pascal* são multiplataforma, dê o máximo de si para não ficar preso a *API* de sistema operacional algum. Isso não é fácil, mas também não é impossível!

OK, estou começando a me interessar por desenvolvimento web com Lazarus, então, onde encontro mais material didático sobre isso?

Além dos **artigos do nosso querido amigo Luiz Americo** e as séries de artigos que pretendo editar, você poderá ver nos exemplos do próprio *Lazarus* e no **wiki**. Além é claro de usar o nosso grupo no *Google* sobre *Lazarus*. **(não use o grupo para tirar dúvidas sobre este artigo, use nossos comentários logo abaixo)**.

Finalização

No próximo artigo irei finalizar a série ***Lazarus e CGI***, fique atento!

Agradeço a todos que tiveram a paciência de acompanhar os artigos I e II da minha primeira série sobre desenvolvimento web com *Lazarus*, e deixo a seguinte mensagem:

Você não imagina o quanto seu comentário é valioso, mesmo o mais modesto. ;-)

Abraços a todos,

Silvio Clécio