

# Capítulo 6

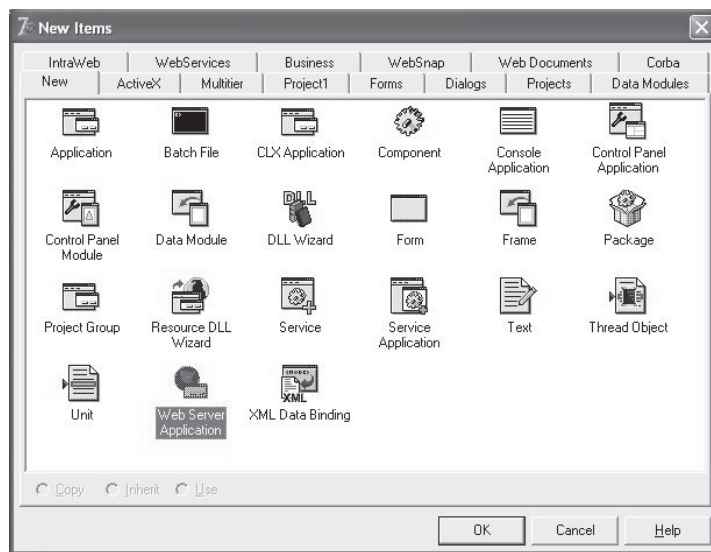
## Delphi x Web x WebBroker

Para entender como funcionam as aplicações servidoras desenvolvidas em Delphi, nada melhor do que aprender na prática. A cada exercício iremos evoluir no aprendizado de aplicações servidoras web.

### Primeiro Exemplo (Hello World)

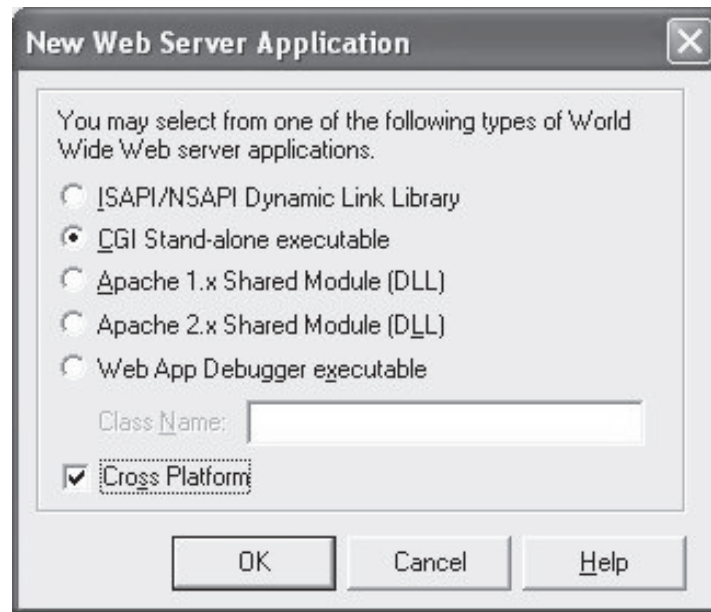
Peço licença aos saudosistas para utilizar o nosso famoso “Hello World” como exemplo.

A partir do Delphi, selecione as opções *File/New/Other...* e em seguida a opção *Web Server Application*, como ilustra a figura 6.1.



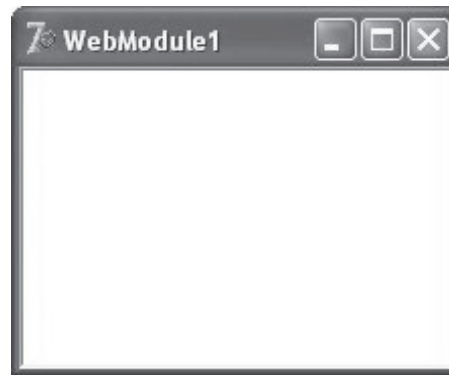
**Figura 6.1** Opção *Web Server Application*

Na janela seguinte selecione a opção *CGI Stand-Alone executable* (figura 6.2), e marque a opção *Cross Platform*, para que nossa aplicação possa ser compilada normalmente no Kylix.



**Figura 6.2** Seleção do tipo da aplicação

Em seguida teremos um *WebModule* (figura 6.3):



**Figura 6.3** *WebModule*

Mas o que é um *WebModule* ?

*WebModule* é um repositório de objetos, com a função de armazenar os objetos não visuais da aplicação tais como (*TPageProducer*, *TQueryPageProducer*, *TQuery*, etc.), bem como responder às requisições do servidor HTTP. Bem, para que uma aplicação servidora possa trabalhar, devemos delegar tarefas, através de *ActionsItems*, ou simplesmente *Itens de Ação*. Para explicar melhor o uso de *ActionsItems*, imagine uma aplicação para inclusão e alteração de clientes. Teremos o seguinte cenário:

### Aplicação (clientes.exe)

::: *ActionItem* (inclusão) – ação para incluir cliente

::: *ActionItem*(alteração)- ação para alterar cliente

É bastante simples, cada *ActionItem* tem uma função específica dentro da aplicação servidora. Para executar no browser uma determinada *ActionItem*, basta fazer como no exemplo:

`http://site/cgi-bin/clientes.exe/inclusao`

Repare que informamos o nome da aplicação (*clientes.exe*) e o nome da *ActionItem* (*/inclusão*). A aplicação servidora não possui limite de *ActionItems*, portanto podemos criar aplicações complexas.

Bem, seguindo o nosso primeiro projeto, através do duplo-clique no *WebModule*, acesse o editor de *ActionItems* (figura 6.4).

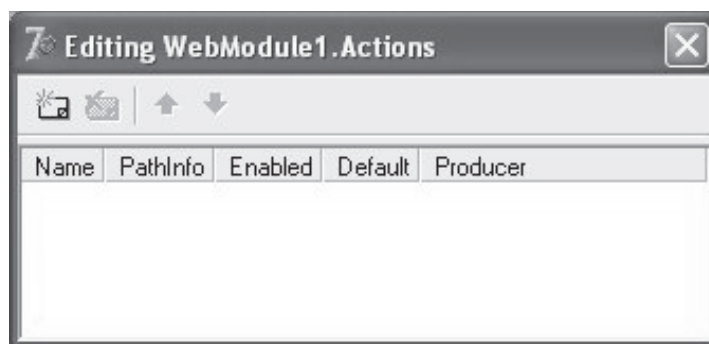


Figura 6.4 Editor ActionItems

Clique no primeiro botão do editor para inserir uma nova *Action* (figura 6.5).

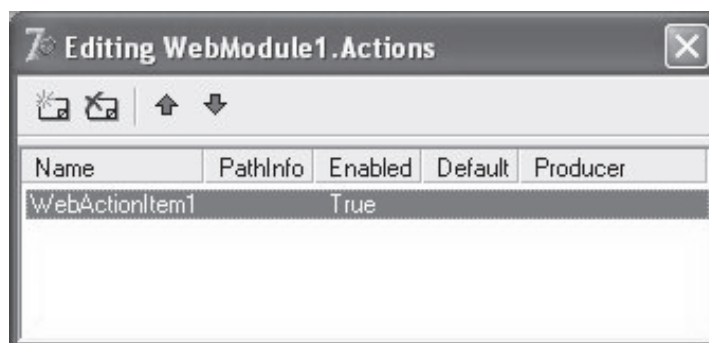

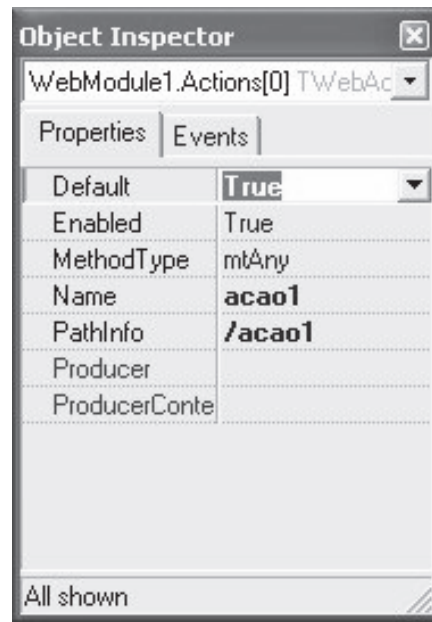


Figura 6.5 ActionItem

Em seguida altere as seguintes propriedades, como ilustra a figura 6.6.

OBJETO		
		
TWebActionItem		
Objeto	Propriedade	Valor
Acao1	Default	True
	Name	Acao1
	PathInfo	/acao1



**Figura 6.6** Propriedades da Action

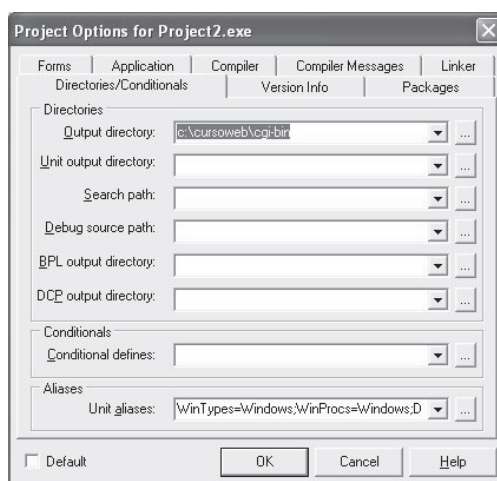
Embora a propriedade *PathInfo* possua o mesmo valor da propriedade *Name*, é ela que executa a *Action*, ou seja, no browser o que vale é o valor da *PathInfo*. No evento *OnAction* coloque o seguinte código: (coloque somente o código em negrito, o restante é criado automaticamente pelo Delphi).

```
procedure TWebModule1.WebModule1acao1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content := 'Olá Mundo' ;
end;
```

No código anterior, estamos utilizando o método **Response** da aplicação servidora. Este método é responsável pela resposta do servidor à uma requisição.

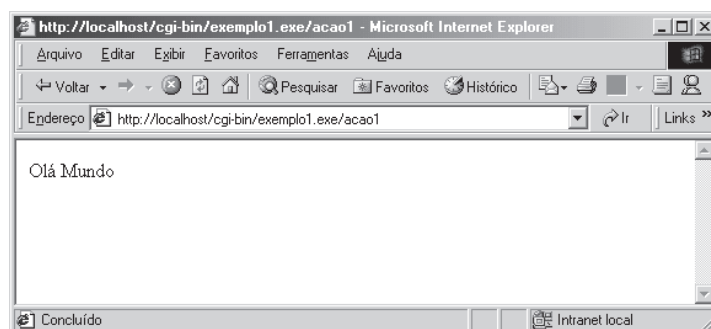
Antes de compilar o projeto, devemos configurar uma opção. Através da opção *Projects/Option* (menu), selecione a seção *Directories/Conditionals* e configure a opção *Output Directory* para *C:\cursoweb\cgi-bin*, como ilustra a figura 6.7.

Tudo pronto. Agora devemos gravar o projeto com o nome de *exemplo1.dpr* e a unit como *un\_1.pas*. Compile o projeto, e digite a seguinte URL no browser: *http://localhost/cgi-bin/exemplo1.exe/acao1*



**Figura 6.7 Configuração do diretório**

A figura 6.8 ilustra o nosso primeiro exemplo.



**Figura 6.8 Exemplo 1 (aplicação servidora)**

### Listagem 6.1

```
unit un_1;

interface

uses
  SysUtils, Classes, HTTPApp;

type
  TWebModule1 = class(TWebModule)
  procedure WebModule1acao1Action(Sender: TObject; Request: TWebRequest;
    Response: TWebResponse; var Handled: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  WebModule1: TWebModule1;

implementation
```

```
{ $R *.DFM }

procedure TWebModule1.WebModule1acaolAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content:='Olá Mundo';
end;

end.
```

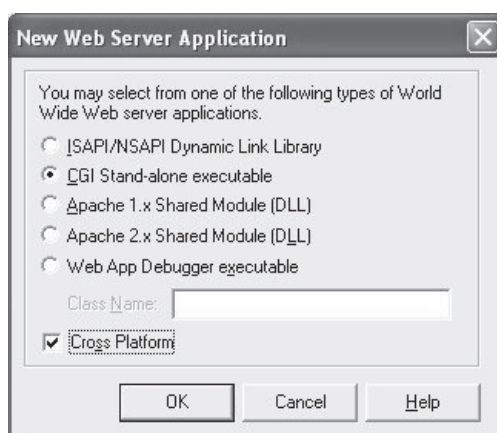
## Segundo Exemplo (Hello World com TPageProducer)

Agora vamos conhecer um excelente objeto para dinamizar nossas aplicações: o *TPageProducer*.

O objeto *TpageProducer* tem como principal função reproduzir uma string de comandos HTML. Na realidade podemos inserir comandos HTML com todas as *Tags* conhecidas, bem como as *Tags transparentes* que são substituídas por comandos HTML contidos no evento *OnHtmlTag*.

Crie uma nova aplicação no Delphi através das opções *File/New...*, selecione a opção *Web Server Application* e clique no botão **OK** para confirmar.

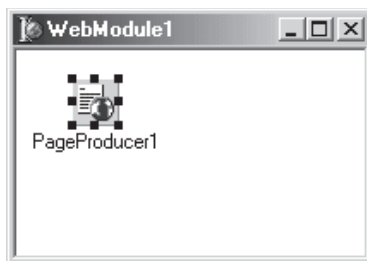
Na janela seguinte selecione a opção *CGI Stand-Alone executable* (figura 6.9).



**Figura 6.9 Seleção do tipo da aplicação**

No *WebModule* vamos inserir um objeto do tipo *TPageProducer*, que se encontra na paleta *Internet*.

Veja como ficou nosso *WebModule* (figura 6.10).

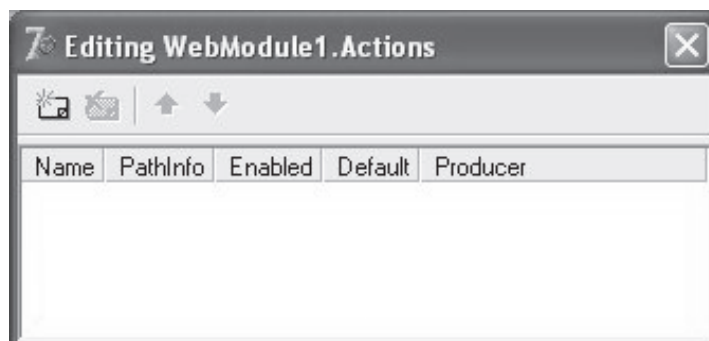


**Figura 6.10 Webmodule**

Para prosseguir altere a propriedade *HTMLDOC* do *PageProducer1* inserindo o seguinte código:

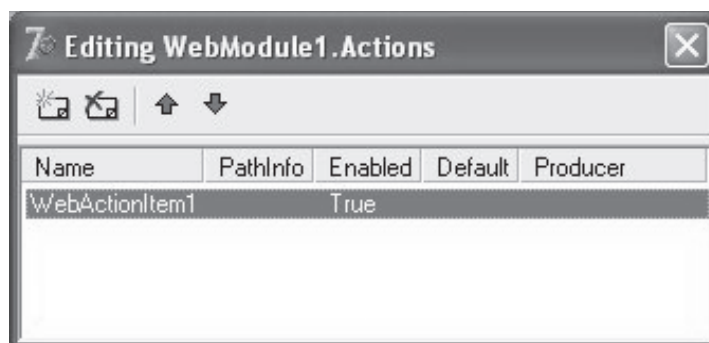
```
<P align=center><B>Olá Mundo</B></P>
```

Reparem que informamos código HTML. Seguindo o nosso projeto, através do duplo-clique no *WebModule*, acesse o editor de *ActionItems* (figura 6.11).




*Figura 6.11 Editor ActionItems*

Clique no primeiro botão do editor para inserir uma nova *Action* (figura 6.12).

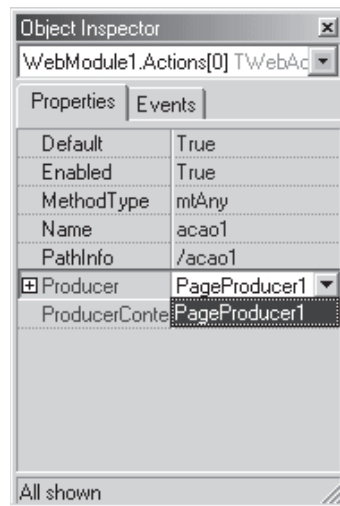


*figura 6.12 ActionItem*

Em seguida altere as seguintes propriedades.

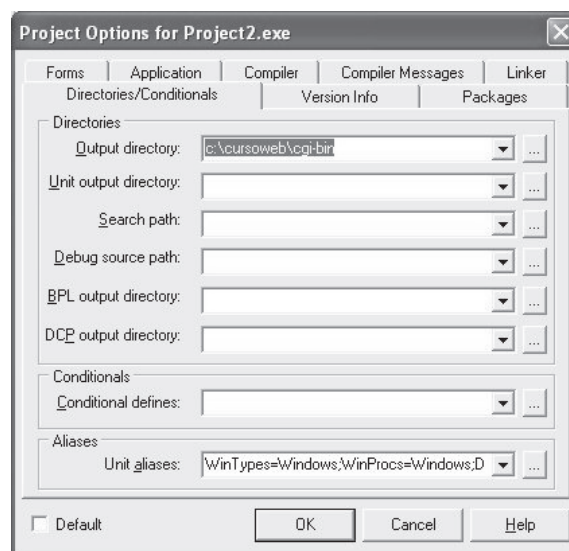
OBJETO		
 <b>TWebActionItem</b>		
Objeto	Propriedade	Valor
Acao1	Default	True
	Name	Acao1
	PathInfo	/acao1
	Producer	PageProducer1

A *figura 6.13* ilustra a alteração da propriedade *Producer*.



**Figura 6.13 Propriedade Producer**

Com isso vinculamos a nossa *Action Acao1* com o nosso *produtor de página PageProducer1*. Veremos que não é necessário disparar um código, como no *exemplo1*. Antes de compilar o projeto, devemos configurar uma opção. Através da opção *Projects/Option* (menu), selecione a seção *Directories/Conditionals* e configure a opção *Output Directory* para *C:\cursoweb\cgi-bin*, como ilustra a *figura 6.14*.



**Figura 6.14 Configuração do diretório**

Tudo pronto. Agora devemos gravar o *projeto* com o nome de *exemplo2.dpr* e a unit como *un\_2.pas*

Compile o *projeto*, e digite a seguinte URL no browser:

*http://localhost/cgi-bin/exemplo2.exe/acao1*

A *figura 6.15* ilustra o nosso exemplo.



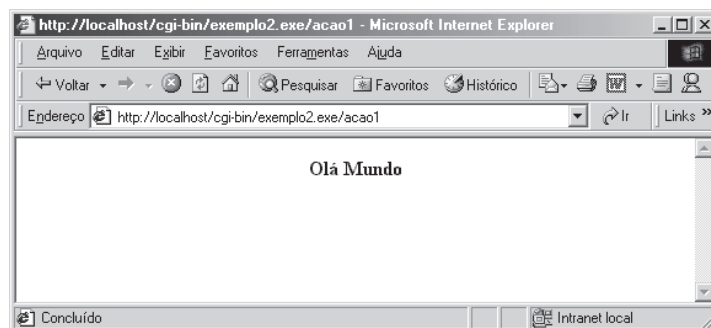


Figura 6.15 Exemplo 2

Perceberam que não colocamos uma linha sequer de código?

Vejam a listagem 6.2, todo o código existente foi gerado automaticamente pelo Delphi.

### Listagem 6.2

```
unit un_2;

interface

uses
  SysUtils, Classes, HTTPApp, HTTPProd;

type
  TWebModule1 = class(TWebModule)
    PageProducer1: TPageProducer;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  WebModule1: TWebModule1;

implementation

{$R *.DFM}

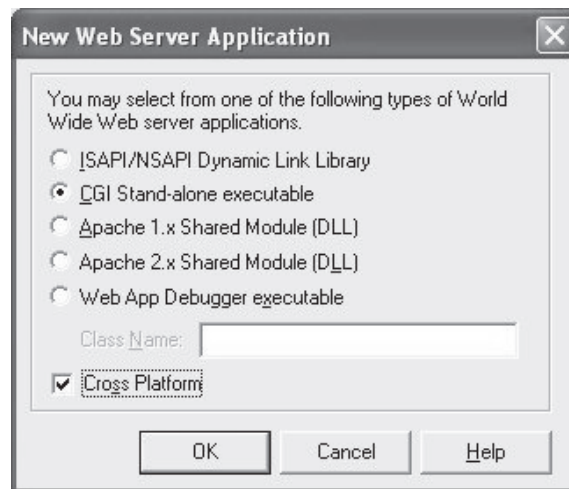
end.
```

## Terceiro Exemplo (PageProducer HTMLFile)

Em nosso terceiro *exemplo* utilizaremos a propriedade *HTMLFile* do objeto *TPageProducer*. O uso desta propriedade é altamente recomendável, pelo fator de utilizar um arquivo HTML externo e minimizar a manutenção no código da aplicação servidora.

Crie uma nova aplicação no Delphi através das opções *File/New...*, selecione a opção *Web Server Application* e clique no botão **OK** para confirmar.

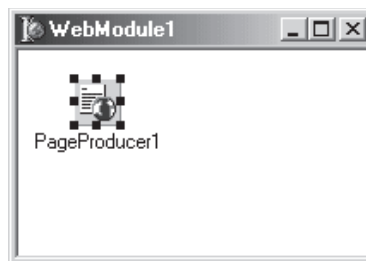
Na janela seguinte selecione a opção *CGI Stand-Alone executable* (figura 6.16).



**Figura 6.16 Seleção do tipo da aplicação**

No *WebModule* vamos inserir um objeto do tipo *TPageProducer*, que se encontra na paleta *Internet*.

Veja como ficou nosso *WebModule* (figura 6.17).



**Figura 6.17 Webmodule**

Para prosseguir, altere a propriedade *HTMLFILE* do *PageProducer1* apontando para o arquivo *olamundo.html* da seguinte forma:

**../olamundo.html**

Você deve colocar desta forma mesmo, dois pontos sequenciais seguidos da barra, e por fim o nome do arquivo. Devemos fazer isto desta maneira, pelo fato da nossa aplicação “rodar” num outro nível de diretório (CGI-BIN), e o arquivo HTML no primeiro nível.

Crie um arquivo HTML no diretório *C:\cursoweb* com o nome *olamundo.html* e insira o código que seguinte

```
<P align=center><B>Olá Mundo</B></P>
<HR>
<P align=center><B>utilizando htmlfile</B></P>
```

Seguindo o nosso projeto, através do duplo-clique no *WebModule*, acesse o editor de *ActionItems* (figura 6.18).

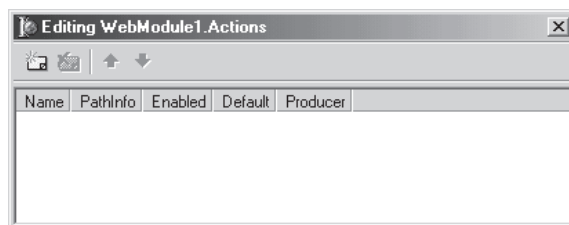


Figura 6.18 Editor ActionItems

Clique no primeiro botão do editor para inserir uma nova Action (figura 6.19).

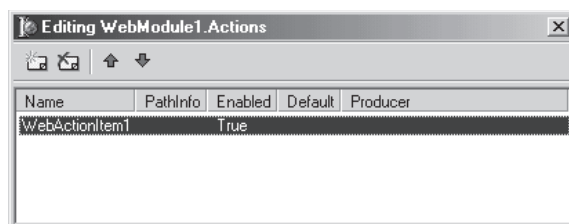



Figura 6.19 ActionItem

Em seguida altere as seguintes propriedades.

OBJETO		
 <div>TwebActionItem</div>		
Objeto	Propriedade	Valor
Acao1	Default	True
	Name	Acao1
	PathInfo	/acao1
	Producer	PageProducer1

A figura 6.20 ilustra a alteração da propriedade *Producer*.

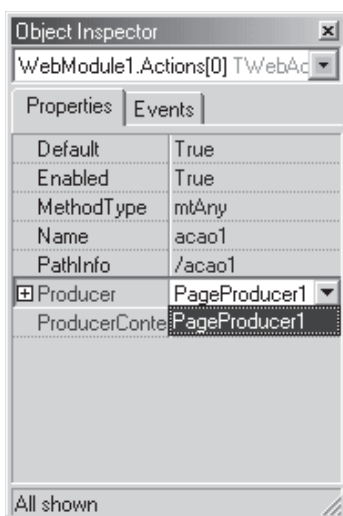
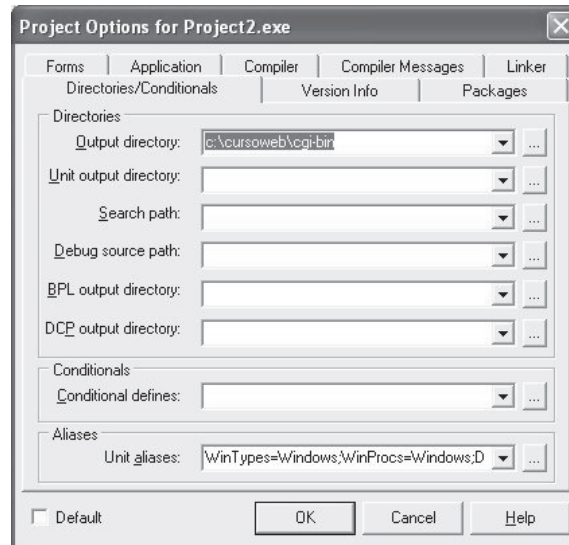


Figura 6.20 Propriedade Producer

Com isso vinculamos a nossa *Action Acao1* com o nosso *produtor de página PageProducer1*. Veremos que não é necessário disparar um código, como no exemplo1.

Antes de compilar o projeto, devemos configurar uma opção. Através da opção *Projects/Option* (menu), selecione a seção *Directories/Conditionals* e configure a opção *Output Directory* para *C:\cursoweb\cgi-bin*, como ilustra a figura 6.21.



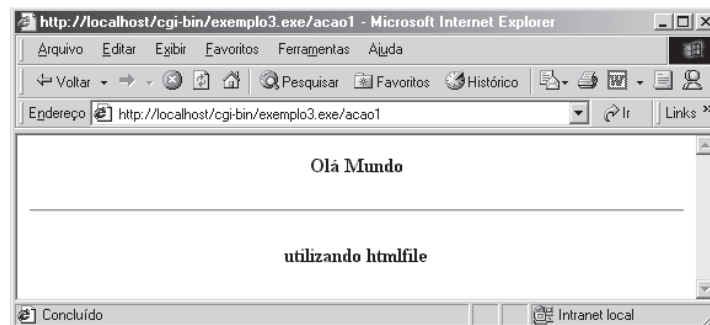
**Figura 6.21** Configuração do diretório

Tudo pronto. Agora devemos gravar o projeto com o nome de *exemplo3.dpr* e a unit como *un\_3.pas*

Compile o projeto, e digite a seguinte URL no browser:

*http://localhost/cgi-bin/exemplo3.exe/acao1*

A figura 6.22 ilustra o nosso exemplo.



**Figura 6.22** Exemplo 3

### Listagem 6.3

```
unit un_3;

interface

uses
  SysUtils, Classes, HTTPApp, HTTPProd;
```

```

type
  TWebModule1 = class(TWebModule)
    PageProducer1: TPageProducer;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  WebModule1: TWebModule1;

implementation

{$R *.DFM}

end.

```

## Quarto Exemplo (PageProducerOnHTMLTag)

Em nosso quarto exemplo utilizaremos o evento *OnHtmlTag* do objeto *TPageProducer*. O uso deste evento está relacionado à substituição de conteúdo das *Tags Transparentes*. Mas o que são *Tags Transparentes*?

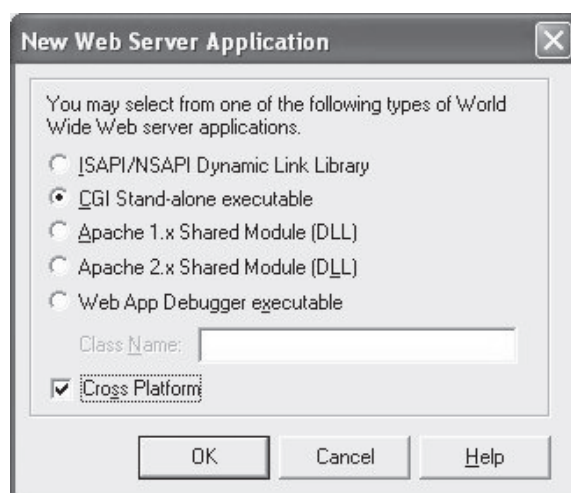
O Delphi interpreta como *Tag Transparente* a seguinte expressão: `<#nome_da_tag>`. Para definir uma *Tag Transparente* num documento HTML, basta inseri-la em qualquer ponto do documento, por exemplo:

**HORA** `<#hora>`

Neste exemplo criamos uma *Tag Transparente* com o nome **hora**. O conteúdo desta *Tag* poderá ser substituído por nossa aplicação servidora. Faremos o exemplo.

Crie uma nova aplicação no Delphi através das opções *File/New...*, selecione a opção *Web Server Application* e clique no botão *OK* para confirmar.

Na janela seguinte selecione a opção *CGI Stand-Alone executable* (figura 6.23).



**Figura 6.23** Seleção do tipo da aplicação

No *WebModule* vamos inserir um objeto do tipo *TPageProducer*, que se encontra na paleta *Internet*. Para prosseguir altere a propriedade *HTMLDOC* (embora eu recomende o uso do *HTMLFile*, neste caso utilizamos a outra forma para agilizar o curso), colocando o código que segue:

```
<P align=center><B>Teste de Tag Transparente</B></P>
<HR>
<P align=center><B>Hora Atual: <#hora></B></P>
```

No evento *OnHTMLTag* do *PageProducer1* insira o seguinte código:

```
if TagString='hora' then ReplaceText:=timetostr(time);
```

Repare que estamos substituindo o conteúdo da *Tag Transparente* **hora** pela hora atual (função *time*).

Seguindo o nosso projeto, através do duplo-clique no *WebModule*, acesse o editor de *ActionItems* (figura 6.24).

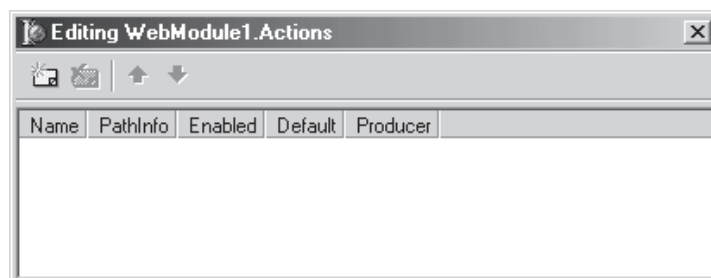


Figura 6.24 Editor ActionItems

Clique no primeiro botão do editor para inserir uma nova *Action* (figura 6.25).

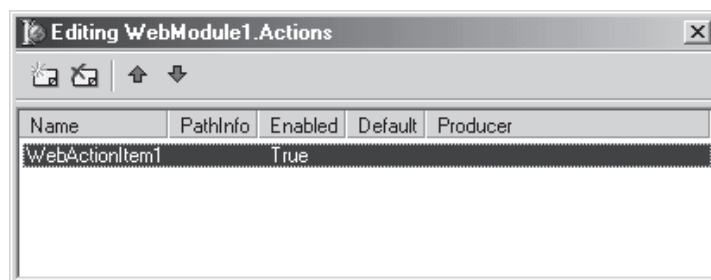

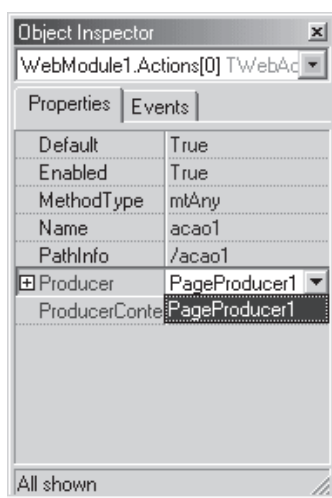


Figura 6.25 ActionItem

Em seguida altere as seguintes propriedades.

OBJETO		
 <div>TWebActionItem</div>		
Objeto	Propriedade	Valor
Acao1	Default	True
	Name	Acao1
	PathInfo	/acao1
	Producer	PageProducer1

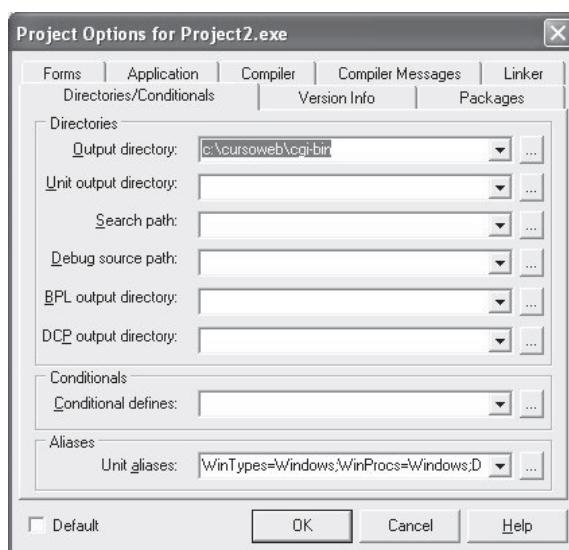
A figura 6.26 ilustra a alteração da propriedade *Producer*.



**Figura 6.26 Propriedade *Producer***

Com isso vinculamos a nossa *Action Acao1* com o nosso *produtor de página PageProducer1*. Veremos que não é necessário disparar um código, como no exemplo1 (isso está se tornando repetitivo, não acham? Mas é para o bem de todos).

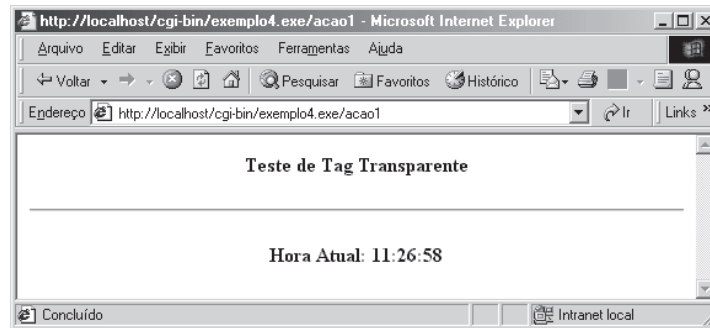
Antes de compilar o projeto, devemos configurar uma opção. Através da opção *Projects/Option* (menu), selecione a seção *Directories/Conditionals* e configure a opção *Output Directory* para *C:\cursoweb\cgi-bin*, como ilustra a figura 6.27.



**Figura 6.27 Configuração do diretório**

Tudo pronto. Agora devemos gravar o projeto com o nome de *exemplo4.dpr* e a unit como *un\_4.pas*. Compile o projeto, e digite a seguinte URL no browser: <http://localhost/cgi-bin/exemplo4.exe/acao1>

A figura 6.28 ilustra o nosso exemplo.



**Figura 6.28 Exemplo 4**

### Listagem 6.4

```
unit un_4;

interface

uses
  SysUtils, Classes, HTTPApp, HTTPProd;

type
  TWebModule1 = class(TWebModule)
    PageProducer1: TPageProducer;
    procedure PageProducer1HTMLTag(Sender: TObject; Tag: TTag;
      const TagString: String; TagParams: TStrings;
      var ReplaceText: String);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  WebModule1: TWebModule1;

implementation

{$R *.DFM}

procedure TWebModule1.PageProducer1HTMLTag(Sender: TObject; Tag: TTag;
  const TagString: String; TagParams: TStrings; var ReplaceText: String);
begin
  if TagString='hora' then ReplaceText:=timetostr(time);
end;

end.
```

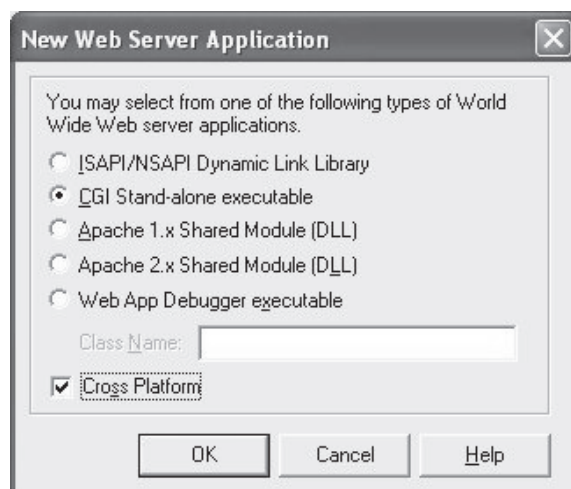


## Quinto Exemplo (Interatividade com Formulários)

Em nosso quinto exemplo, trabalharemos com formulários HTML, interagindo o Delphi e o documento HTML.

Crie uma nova aplicação no Delphi através das opções *File/New...*, selecione a opção *Web Server Application* e clique no botão **OK** para confirmar.

Na janela seguinte selecione a opção *CGI Stand-Alone executable* (figura 6.29).



**Figura 6.29** Seleção do tipo da aplicação

No *WebModule* vamos inserir um objeto do tipo *TPageProducer*, que se encontra na paleta *Internet*.

Para prosseguir altere a propriedade *HTMLFILE* do *PageProducer1* apontando para o arquivo *olamundo.html* da seguinte forma:

**../formulario.html**

Crie um arquivo HTML no diretório *C:\cursoweb* com o nome *formulario.html* e insira o código que seguinte

```
<html>
<head>
<title>Exemplo de Formulário</title>
</head>
<body bgcolor="#FFFFFF">
<form method="post" action="confirma" name="form1">
Digite seu nome
  <input type="text" name="NOME" size="50"
maxlength="50">
  <input type="submit" name="Submit">
</form>
</body>
</html>
```

Seguindo o nosso projeto, através do duplo-clique no *WebModule*, acesse o editor de *ActionItems* (figura 6.30).

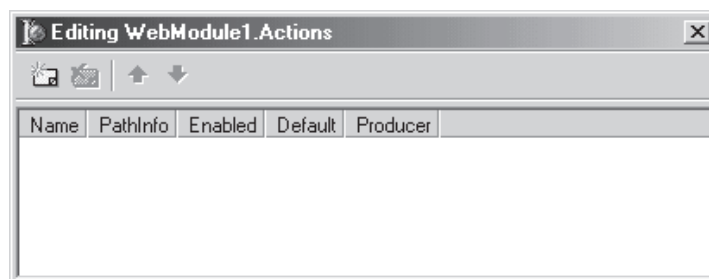


Figura 6.30 Editor ActionItems

Clique no primeiro botão do editor para inserir uma nova *Action* (figura 6.31).

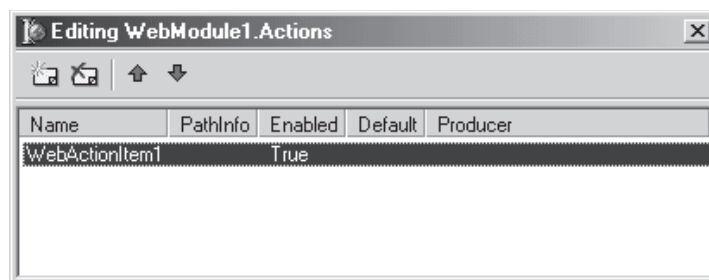

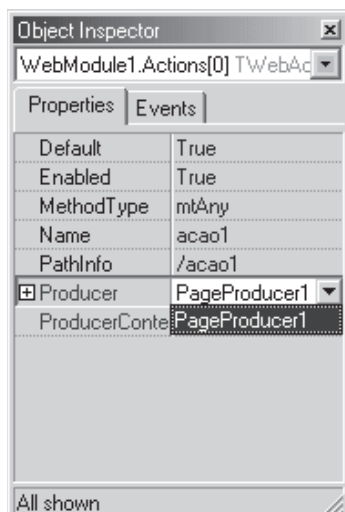


Figura 6.31 ActionItem

Em seguida altere as seguintes propriedades.

OBJETO		
 <b>TWebActionItem</b>		
Objeto	Propriedade	Valor
Formulário	Default	True
	Name	formulario
	PathInfo	/formulario
	Producer	PageProducer1


A figura 6.32 ilustra a alteração da propriedade *Producer*.



**Figura 6.32 Propriedade Producer**

Repare no código HTML do documento, que estamos encaminhando o conteúdo do formulário para a *Action* “*confirma*”. Para que o formulário possa ser processado em nossa aplicação servidora, devemos criar tal item.

Crie uma nova *Action* alterando as propriedades que seguem:

OBJETO		
		
<b>TWebActionItem</b>		
Objeto	Propriedade	Valor
confirma	Default	False
	Name	confirma
	PathInfo	/confirma

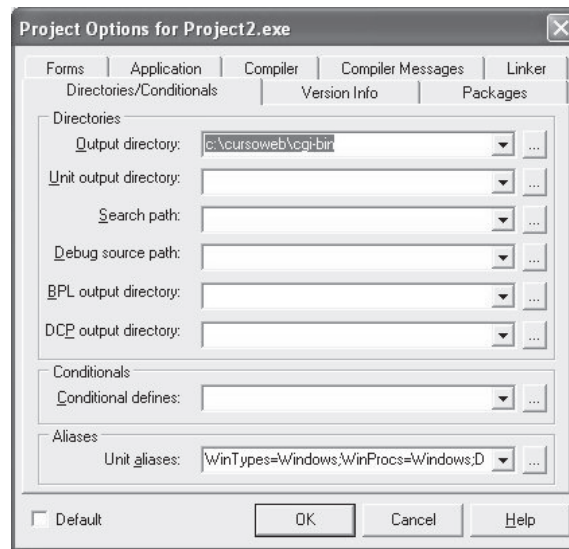
Repare que neste caso não vinculamos o *PageProducer*, pois daremos uma outra resposta para esta *Action*.

No evento *OnAction* da *Action* *confirma* insira o código que segue:

```
Response.Content:=' Seja bem vindo ' + Request.ContentFields.Values['nome'];
```

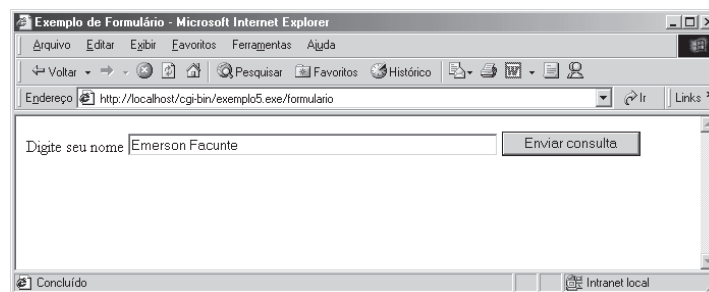
Neste caso estamos utilizando o objeto *Request*, que tem como principal função extrair informações do requerente, em nosso caso o formulário HTML. O método *ContentFields* extrai informações enviadas através do método *POST* do formulário. Em nosso caso, pegamos o conteúdo do campo *nome*.

Antes de compilar o projeto, devemos configurar uma opção. Através da opção *Projects/Option* (menu), selecione a seção *Directories/Conditionals* e configure a opção *Output Directory* para *C:\cursoweb\cgi-bin*, como ilustra a figura 6.33.

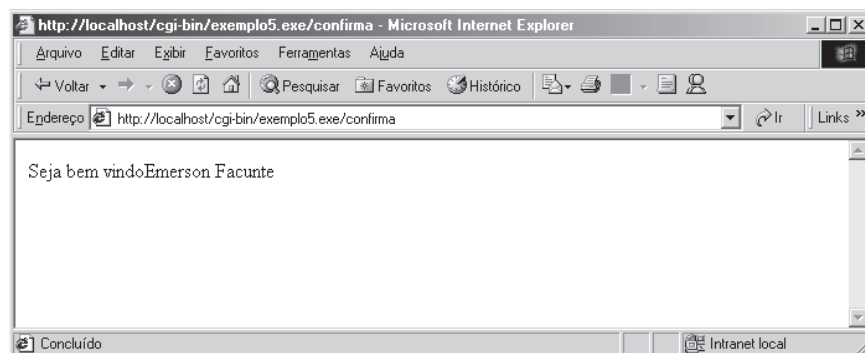


**Figura 6.33** Configuração do diretório

Tudo pronto. Agora devemos gravar o projeto com o nome de *exemplo5.dpr* e a unit como *un\_5.pas*. Compile o projeto, e digite a seguinte URL no browser: <http://localhost/cgi-bin/exemplo5.exe/formulario>. As figuras 6.34 e 6.35 ilustram o nosso exemplo.



**Figura 6.34** Inserindo o nome (Action Formulário)



**Figura 6.35** Recebendo a resposta da aplicação (Action Confirma)

### Listagem 6.5

```
unit un_5;

interface

uses
  SysUtils, Classes, HTTPApp, HTTPProd;

type
  TWebModule1 = class(TWebModule)
    PageProducer1: TPageProducer;
    procedure WebModule1confirmaAction(Sender: TObject;
      Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  WebModule1: TWebModule1;

implementation

{$R *.DFM}

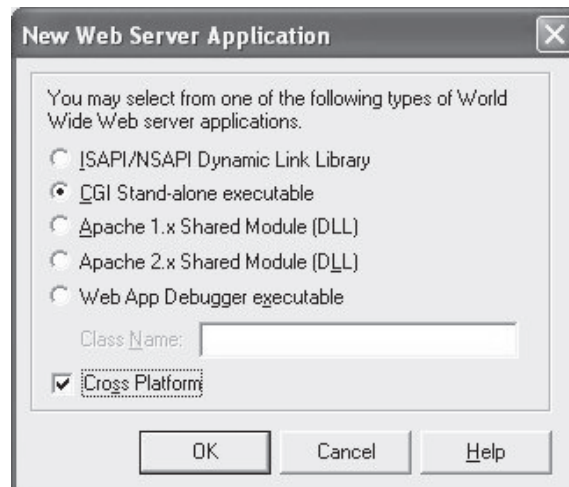
procedure TWebModule1.WebModule1confirmaAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  Response.Content:='Seja bem vindo'+ Request.ContentFields.Values['nome'];
end;

end.
```

## Sexto Exemplo (Formulários)

Em nosso sexto exemplo, também iremos trabalhar com formulários HTML, interagindo o Delphi e o documento HTML.

Crie uma nova aplicação no Delphi através das opções *File/New...*, selecione a opção *Web Server Application* e clique no botão **OK** para confirmar. Na janela seguinte selecione a opção *CGI Stand-Alone executable* (figura 6.36).



**Figura 6.36 Seleção do tipo da aplicação**

No *WebModule* vamos inserir um objeto do tipo *TPageProducer*, que se encontra na paleta *Internet*.

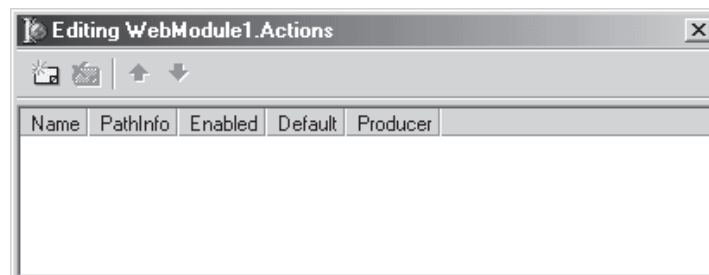
Para prosseguir, altere a propriedade *HTMLFILE* do *PageProducer1* apontando para o arquivo *olamundo.html* da seguinte forma:

**../formulario2.html**

Crie um arquivo HTML no diretório *C:\cursoweb* com o nome *formulario2.html* e insira o código que seguinte

```
<html>
<head>
<title>Exemplo de Formulário</title>
</head>
<body bgcolor="#FFFFFF">
<form method="post" action="soma" name="form1">
Digite o primeiro numero
  <input type="text" name="NUMERO1" size="10" maxlength="10">
<BR>
Digite o segundo numero
  <input type="text" name="NUMERO2" size="10" maxlength="10">
  <input type="submit" name="Submit">
</form>
</body>
</html>
```

Seguindo o nosso projeto, através do duplo-clique no *WebModule*, acesse o editor de *ActionItems* (figura 6.37).



**Figura 6.37 Editor ActionItems**

Clique no primeiro botão do editor para inserir uma nova *Action* (figura 6.38).

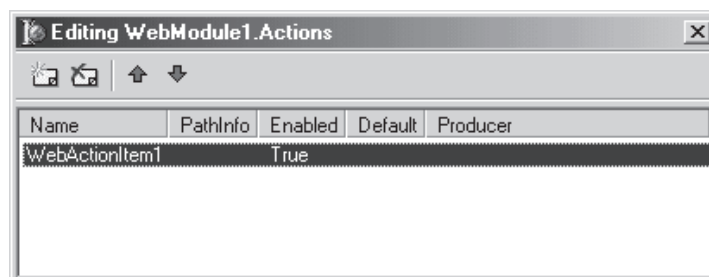




Figura 6.38 ActionItem

Em seguida altere as seguintes propriedades.

OBJETO		
	TWebActionItem	
Objeto	Propriedade	Valor
formulario	Default	True
	Name	formulario
	PathInfo	/formulario
	Producer	PageProducer1

Repare no código HTML do documento que estamos encaminhando o conteúdo do formulário para a *Action* “soma”. Para que o formulário possa ser processado em nossa aplicação servidora, devemos criar tal item. Crie uma nova *Action* alterando as propriedades que seguem:

OBJETO		
	TWebActionItem	
Objeto	Propriedade	Valor
soma	Default	False
	Name	soma
	PathInfo	/soma

Repare que neste caso não vinculamos o *PageProducer*, pois daremos uma outra resposta para esta *Action*. No evento *OnAction* da *Action* soma insira o código que segue:

(coloque somente o código em negrito, o restante foi criado automaticamente pelo Delphi)

```
procedure TWebModule1.WebModule1somaAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);

var
  numero1, numero2: single;
begin
  numero1:=StrtoFloat(Request.ContentFields.Values ['NUMERO1']);
  numero2:=StrtoFloat(Request.ContentFields.Values ['NUMERO2']);
```

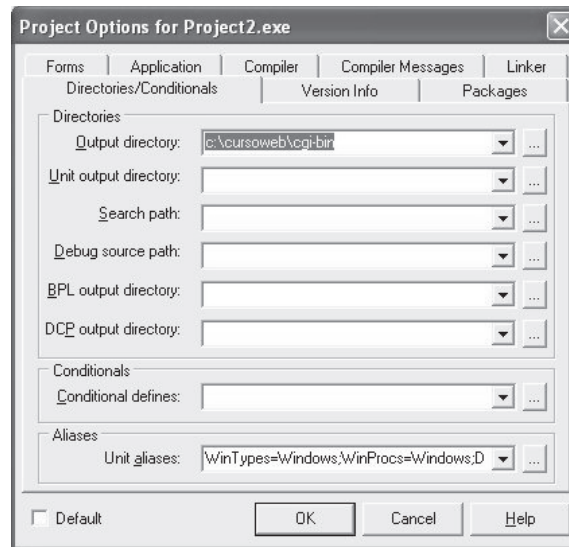
```

Response.Content:='A soma de '+
    floattostr(numero1)+' e '+
    floattostr(numero2)+' = '+
    floattostr(numero1+numero2) ;

end;

```

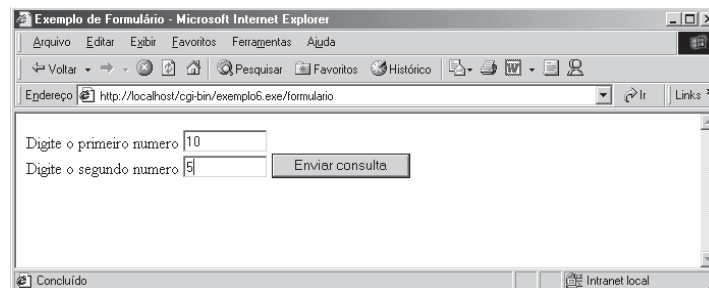
Antes de compilar o projeto, devemos configurar uma opção. Através da opção *Projects/Option* (menu), selecione a seção *Directories/Conditionals* e configure a opção *Output Directory* para *C:\cursoweb\cgi-bin*, como ilustra a figura 6.39.



**Figura 6.39** Configuração do diretório

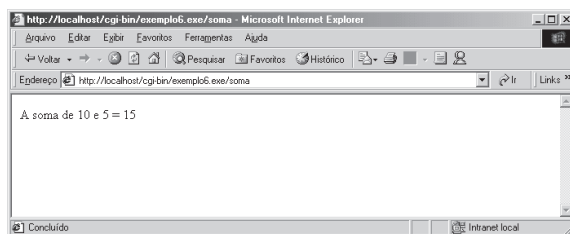
Tudo pronto. Agora devemos gravar o projeto com o nome de *exemplo6.dpr* e a unit como *un\_6.pas*. Compile o projeto, e digite a seguinte URL no browser: <http://localhost/cgi-bin/exemplo6.exe/formulario>

As figuras 6.40 e 6.41 ilustram o nosso exemplo.



**Figura 6.40** Action formulário





**Figura 6.41 Action SOMA**

Com isso concluímos este capítulo.

### Listagem 6.6

```
unit un_6;

interface

uses
  SysUtils, Classes, HTTPApp, HTTPProd;

type
  TWebModule1 = class(TWebModule)
    PageProducer1: TPageProducer;
    procedure WebModule1somaAction(Sender: TObject; Request: TWebRequest;
      Response: TWebResponse; var Handled: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  WebModule1: TWebModule1;

implementation

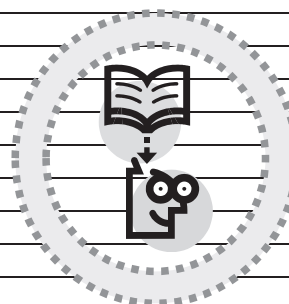
{$R *.DFM}

procedure TWebModule1.WebModule1somaAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  numero1, numero2: single;
begin
  numero1 := StrtoFloat(Request.ContentFields.Values['NUMERO1']);
  numero2 := StrtoFloat(Request.ContentFields.Values['NUMERO2']);

  Response.Content := 'A soma de ' +
    floattostr(numero1) + ' e ' +
    floattostr(numero2) + ' = ' +
    floattostr(numero1 + numero2);

end;

end.
```

**Anotações de Dúvidas****Preciso Revisar****Anotações Gerais**