#$KTCGI Component

Properties Methods

Unit
CGI

Description
The TCGI Component allows Delphi programs to link to information servers supporting Windows CGI version 1.1 and above. Used in conjunction with a compatible server, it allows collection, processing, and publication of data on the World Wide Web. Its core functionality is provided by the FormFields and Method properties, and by the Send method. It is intentionally lacking in bells and whistles, providing the basic properties and methods needed with as little overhead as possible.

The companion TCGIDlg Component makes visually debugging CGI applications quick and easy.

See Also
Acknowledgements

#hlp_tcgi
$TCGI Component
KTCGI

Properties

ContentType	ExternalFields	FormFields
HugeFields	Method	Profile
ResponseHeaders	StdOut	ServerStatus

tcgi_prop

#Methods

Send SendContent

#$ExternalFields Property

Unit
CGI

Applies to
TCGI

Declaration
```
property ExternalFields: TTupleList;
```

Description
The ExternalFields property contains a list of key/value pairs describing the external field data (field data between 255 and 65,534 bytes) entered into the form on the client end. This data should not be accessed directly. Use the GetExternalSize and GetExternalData methods.

#cgi_ExternalFields
$ExternalFields property

[#][$]$FormFields Property

Unit
CGI

Applies to
TCGI

Declaration
```
property FormFields: TTupleList;
```

Description
The FormFields property contains a list of key/value pairs describing the small field data (field data of 254 bytes or fewer) entered into the form on the client end.

[#]cgi_FormFields
[$]FormFields property

[#][$]$HugeFields Property

Unit
CGI

Applies to
TCGI

Declaration
```
property HugeFields: TTupleList;
```

Description
The HugeFields property contains a list of key/value pairs describing the huge field data (data larger than 65,535 bytes) sent by the client.

[#]cgi_HugeFields
[$]HugeFields property

[#][$]#$Method Property

Unit
CGI

Applies to
TCGI

Declaration
```
property Method: TRequestMethod;
```

Description
The Method property translates the RequestMethod field of the Profile record into an enumerated type. This is useful in that it allows you to use a **case Method of** construct to process CGI requests, since Delphi only allows the use of ordinal types in case statements.

If the TCGI component doesn't recognize the method type, Method will be set to `rmOTHER`, and you can test for the specific method by accessing the original, untranslated method string ('GET', 'POST', etc.) in Profile.RequestMethod.

Example
```
procedure CGIProcess;
begin
   case CGI.Method of
   rmGET: SendForm;
   rmPOST: ProcessForm;
   else SendMethodError;
end;
```

[#]cgi_Method
[$]Method property

[#][$]$Profile Property

Unit
CGI

Applies to
TCGI

Declaration
```
property Profile: TCGIProfile;
```

Description
The Profile property contains the CGI profile information for the current session.

[#]cgi_Profile
[$]Profile property

[#][$]$ResponseHeaders Property

Unit
CGI

Applies to
TCGI

Declaration
```
property ResponseHeaders: TStringList;
```

Description
Use the ResponseHeaders property to output any HTTP response headers needed in the return document. **Do not use this property to output Status, Content-Type, or Content-Length headers.** Status and Content-Type are output automatically as designated by the ServerStatus and ContentType properties, respectively. Content-Length is determined automatically when the SendContent method is called.

This property will usually not be needed. For detailed descriptions of the standard HTML response headers, see the HTTP 1.0 specification.

[#]cgi_ResponseHeaders
[$]ResponseHeaders property

[#][$]$ServerStatus Property

Unit
CGI

Applies to
TCGI

Declaration
```
property ServerStatus: TServerStatus;
```

Description
ServerStatus contains the HTTP document status, which gets translated into the proper HTTP response header when the SendContent method is called. All of the statuses described by the HTTP 1.0 specification are supported by the ServerStatus property. The default is stOK, which translates into the '200 OK' response header.

[#]cgi_ServerStatus
[$]ServerStatus property

#$ContentType Property

Unit
CGI

Applies to
TCGI

Declaration
```
property ContentType: String;
```

Description
The ContentType property contains the MIME content type/subtype of the data which will be returned by the CGI application. It is used by the SendContent method to generated the HTTP Content-Type response header. The default is 'text/html', but can be changed for other types of return data ('image/gif', for example). **This property must contain a valid MIME type/subtype, or you will most likely choke the browser!**

#cgi_ContentType
$ContentType property

[#][#][$]$StdOut Property

Unit
CGI

Applies to
TCGI

Declaration
```
property StdOut: TMemoryStream;
```

Description
The StdOut property acts as virtual standard output for the CGI application. For string-based data, you don't need to access StdOut directly; use the TCGI.Send method instead. For data stored in a memory buffer (or a PChar), use the StdOut.Write method. For stream-based data, use the StdOut.LoadFromStream method or the source stream's SaveToStream method.

Examples
```
procedure SendSomeStuff;
var
   Buffer: PChar;
   PicFile: TFileStream;
begin
   Buffer := StrNew('This is how you send a PChar.');
   PicFile := TFileStream.Create('picture.bmp',fmOpenRead);
   with CGI do begin
      { Use Send to output a string }
      Send('This works fine for strings.');
      try
         { Use Write to output a buffer }
         StdOut.Write(Buffer,StrLen(Buffer));

         { Use LoadFromStream to output stream content }
         StdOut.LoadFromStream(PicFile);
      finally
         StrDispose(Buffer);
         PicFile.Free;
      end;
   end;
end;
```

[#]cgi_StdOut
[$]StdOut property

[#][#][$]$Send Method

Unit
CGI

Applies to
TCGI

Declaration
```
procedure Send(Text: String);
```

Description
The Send method writes a string to the StdOut stream, which is later returned to the server via the SendContent method.

Example
```
CGI1.Send('All this talk about servers is making me <EM>hungry</EM>.');
```

[#]cgi_Send
[$]Send method

[#][$]$SendContent Method

Unit
CGI

Applies to
TCGI

Declaration
```
procedure SendContent;
```

Description
The SendContent method sends the buffered response data contained in StdOut to the server-specified output file. It also generates the Status, Content-Type, and Content-Length response headers based on the ServerStatus and ContentType properties and the length of the data in the StdOut stream. SendContent should be the last method your CGI application calls, since it finalizes the length of the data stream and reports the content information back to the server.

[#]cgi_SendContent
[$]SendContent method

#AcceptTypes Field

Applies to
TCGIProfile

Declaration
```
AcceptTypes: TTupleList;
```

Description
The AcceptTypes field contains the key/value pairs describing the MIME types that the client reports it can accept.

#AuthType Field

Applies to
TCGIProfile

Declaration
```
AuthType: String;
```

Description
If execution of the back-end is protected, AuthType is the protocol-specific authentication method used to validate the user.

#AuthUser Field

Applies to
TCGIProfile

Declaration
```
AuthUser: String;
```

Description
If execution of the back-end is protected, AuthUser is the username that the client used to authenticate for access to the back-end.

#ContentFile Field

Applies to
TCGIProfile

Declaration
```
ContentFile: String;
```

Description
The full name (including path) of the file containing the raw request content (for requests which have attached data).

#ContentLength Field

Applies to
TCGIProfile

Declaration
```
ContentLength: LongInt;
```

Description
The length (in bytes) of the data supplied with the request (for requests which have attached data).

#ContentType Field

Applies to
TCGIProfile

Declaration
```
ContentType: String;
```

Description
For requests which have attached data, ContentType is the MIME content type of the data in the format *type*/*subtype*. Example: "text/html"

#DebugMode Field

Applies to
TCGIProfile

Declaration
```
DebugMode: Boolean;
```

Description
DebugMode is True if the server's back-end debug flag is set.

#ExecutablePath Field

Applies to
TCGIProfile

Declaration
```
ExecutablePath: String;
```

Description
The logical path to the back-end executable, as needed for self-referencing URLs.

#prof_ExecutablePath

#ExtraHeaders Field

Applies to
TCGIProfile

Declaration
```
ExtraHeaders: TTupleList;
```

Description
The ExtraHeaders field contains a list of key/value pairs describing extra data reported by the client (e.g., browser name).

#prof_ExtraHeaders

#GMTOffset Field

Applies to
TCGIProfile

Declaration
```
ExtraHeaders: LongInt;
```

Description
The number of seconds to be added to GMT time to reach local time. For Pacific Standard Time, this number is -28,800. Useful for computing GMT times.

#prof_GMTOffset

#LogicalPath Field

Applies to
TCGIProfile

Declaration
```
LogicalPath: String;
```

Description
A request may specify a path to a resource needed to complete that request. This path may be in a logical pathname space. This item contain the pathname exactly as received by the server, without logical-to-physical translation.

Example
In the following URL, the LogicalPath info is in **boldface**:

http://www.fruit.org/cgi-win/compare**/apples/oranges**

See Also
PhysicalPath

#prof_LogicalPath

#OutputFile Field

Applies to
TCGIProfile

Declaration
```
OutputFile: String;
```

Description
The full path/name of the file in which the server expects to receive the back-end's results. There is usually no need to maintain the file yourself; it is handled by the Send method.

#PhysicalPath Field

Applies to
TCGIProfile

Declaration
```
PhysicalPath: String;
```

Description
If the request contained logical path information, the server provides the path in physical form, in the native object (e.g., file) access syntax of the operating system.

See Also
LogicalPath

#ProfileFile Field

Applies to
TCGIProfile

Declaration
```
ProfileFile: String;
```

Description
The full name (including path) of the file containing the CGI environment information and decoded form data.

#QueryString Field

Applies to
TCGIProfile

Declaration
```
QueryString: String;
```

Description
The information which follows the ? in the URL that generated the request is the "query" information. The server furnishes this to the back end whenever it is present on the request URL, without any decoding or translation.

Example
In the following URL, the QueryString is in **boldface**:

http://www.fruit.org/cgi-win/getprice?**grapes+kiwis**

#RemoteAddr Field

Applies to
TCGIProfile

Declaration
```
RemoteAddr: String;
```

Description
The network (IP) address of the client (requestor) system. This item is used for logging if the host name is not available.

#RemoteHost Field

Applies to
TCGIProfile

Declaration
```
RemoteHost: String;
```

Description
The network host name of the client (requestor) system, if available. This item is used for logging.

#RequestMethod Field

Applies to
TCGIProfile

Declaration
```
RequestMethod: String;
```

Description
The method with which the request was made. For HTTP, this is "GET", "HEAD", "POST", etc.

#prof_RequestMethod

#RequestProtocol Field

Applies to
TCGIProfile

Declaration
```
RequestProtocol: String;
```

Description
The name and revision of the information protocol this request came in with in the format *protocol*/*revision*. Example: "HTTP/1.0".

#ServerAdmin Field

Applies to
TCGIProfile

Declaration
```
ServerAdmin: String;
```

Description
The e-mail address of the server administrator.

#prof_ServerAdmin

#ServerName Field

Applies to
TCGIProfile

Declaration
```
ServerName: String;
```

Description
Hostname (or alias) of the information server. Needed for self-referencing URLs.

#ServerPort Field

Applies to
TCGIProfile

Declaration
```
ServerPort: Integer;
```

Description
The information server's network port number. Needed for self-referencing URLs.

#ServerSoftware Field

Applies to
TCGIProfile

Declaration
```
ServerSoftware: String;
```

Description
The name and version of the information server software.

#TAPUser Field

Applies to
TCGIProfile

Declaration
```
TAPUser: String;
```

Description
TAP identity of the authenticated client user.

#Version Field

Applies to
TCGIProfile

Declaration
```
Version: String;
```

Description
The revision of the CGI specification to which this information server complies. Format: CGI/*revision*. For this version, "CGI/1.1 WIN".

#TCGIProfile Type

Unit
CGI

Declaration
```
TCGIProfile = record
   AcceptTypes: TTupleList;
   AuthType: String;
   AuthUser: String;
   ContentFile: String;
   ContentLength: LongInt;
   ContentType: String;
   DebugMode: ByteBool;
   ExecutablePath: String;
   ExtraHeaders: TTupleList;
   GMTOffset: LongInt;
   LogicalPath: String;
   OutputFile: String;
   PhysicalPath: String;
   ProfileFile: String;
   QueryString: String;
   RemoteAddr: String;
   RemoteHost: String;
   RequestMethod: String;
   RequestProtocol: String;
   ServerAdmin: String;
   ServerName: String;
   ServerPort: Integer;
   ServerSoftware: String;
   TAPUser: String;
   Version: String;
end;
```

Description
The TCGIProfile type holds Common Gateway Interface (CGI) profile information.

See Also
TCGI Component

#hlp_tcgiprofile

#$TRequestMethod Type

Unit
CGI

Declaration
```
TRequestMethod = (rmGET, rmPOST, rmTEXTSEARCH, rmHEAD, rmLINK, rmUNLINK,
   rmPUT, rmOTHER);
```

Description
TRequestMethod defines the possible values of the Method property.

#hlp_trequestmethod
$TRequestMethod Type

#$TServerStatus Type

Unit
CGI

Declaration
```
TServerStatus = (stOK, stCreated, stAccepted, stPartialInfo, stNoResponse,
   stMoved, stNotModified, stBadRequest, stUnauthorized, stPaymentRequired,
   stForbidden, stNotFound, stInternalError, stNotImplemented, stOverloaded,
   stTimeout);
```

Description
TServerStatus defines the possible values of the ServerStatus property. All of the values defined by the HTML 1.0 specification are represented.

#hlp_tserverstatus
$TRequestMethod Type

#$TTupleList Object

Properties Methods

Unit
CGI

Description
The TTupleList object is descended from the TStringList object. It adds the Keys and IntValues properties and the IndexOfKey method.

#hlp_ttuplelist
$TTupleList Object

#**#Properties**

IntValues Keys

#ttuplelist_prop

#**#Methods**

GetExternalData GetExternalSize
IndexOfKey

#ttuplelist_meth

[#][$]$IntValues Property

Unit
CGI

Applies to
TTupleList

Declaration
```
property IntValues[const Key: String]: Integer;
```

Description
Returns the value half of the key/value pair identified by Key as an integer type. See the TStringList.Values property for more information on key/value pairs stored in string lists.

Example
If the key/value pair `'guava=30'` were stored in TupleList `FruitCount`, then `FruitCount.IntValues['guava']` would be equal to `30`.

See Also
Keys property

[#]tuple_IntValues
[$]IntValues property

[#][#][$]$Keys Property

Unit
CGI

Applies to
TTupleList

Declaration
```
property Keys[const Index: Integer]: String;
```

Description
Returns the key half of the key/value pair at the specified Index in the TupleList. See the TStringList.Values property for more information on key/value pairs stored in string lists.

Example
If the key/value pair `'guava=30'` were stored as the first item in TupleList `FruitCount`, then `FruitCount.Keys[0]` would be equal to `'guava'`.

See Also
IntValues property
IndexOfKey method

[#]tuple_Keys
[$]Keys property

#$IndexOfKey Method

Unit
CGI

Applies to
TTupleList

Declaration
```
function IndexOfKey(const Key: String): Integer;
```

Description
Returns the index of the key/value pair identified by Key. If the specified Key does not exist, IndexOfKey returns -1.

See Also
Keys property

#tuple_IndexOfKey
$IndexOfKey method

[#][#][$]$GetExternalData Method

Unit
CGI

Applies to
TTupleList

Declaration
```
function GetExternalData(const Key: String, var Buffer: PChar): Integer;
```

Description
Reads the external field specified by Key into Buffer. Buffer must be large enough to hold the external field data. Returns the number of bytes read.

Example
```
var
   Buffer: PChar;
   Size: Integer;
begin
   with CGI.ExternalFields do begin
      Size := GetExternalSize('kiwi');
      Buffer := StrAlloc(Size);
      GetExternalData('kiwi',Buffer);
   end;
end;
```

See Also
GetExternalSize method

[#]tuple_GetExternalData
[$]GetExternalData method

[#][$]$GetExternalSize Method

Unit
CGI

Applies to
TTupleList

Declaration
```
function GetExternalSize(const Key: String): Integer;
```

Description
Returns the number of bytes required to read the field data specified by Key into a buffer.

Example
```
var
   Buffer: PChar;
   Size: Integer;
begin
   with CGI.ExternalFields do begin
      Size := GetExternalSize('kiwi');
      Buffer := StrAlloc(Size);
      GetExternalData('kiwi',Buffer);
   end;
end;
```

See Also
GetExternalData method

[#]tuple_GetExternalSize
[$]GetExternalSize method

#$KTCGIDlg Component

Properties Methods

Unit
CGIDlg

Description
The TCGIDlg component provides feedback about the status and operation of the TCGI component. Calling the Execute method will pop up a modal dialog displaying the values included in the CGI Profile, any Form Fields, and a buffer containing the data to be sent back to the client.

#hlp_TCGIDebugDlg
$TCGIDlg Component
KTCGIDlg

#**#Properties**
CGI

#debug_prop

#Methods

Execute

#debug_meth

[#][$]$CGI Property

Unit
CGIDlg

Applies to
TCGIDlg

Declaration
```
property CGI: TCGI;
```

Description
The CGI Property points to the TCGI component to be dumped/debugged/displayed.

[#]debug_cgi
[$]CGI Property

[#][$]$Execute Method

Unit
CGIDlg

Applies to
TCGIDlg

Declaration
```
procedure Execute;
```

Description
The Execute method retrieves the relevant data from the CGI component and displays the debugging dialog.

[#]debug_exec
[$]Execute Method

#**#**The Common Gateway Interface (CGI) acts as a conduit between a web server and a back-end processing application (or script). CGI defines how the back-end script retrieves its data (such as query strings or filled out form fields), and where it should put the results it generates for transmission back to the client.

#def_CGI

#$About the TCGI Component

The TCGI component is being released to the public as postcard-ware. If you find it useful, entertaining, enlightening, or simply cool, just send a postcard letting me know what you think, and consider yourself registered! Postcards larger than 1 bit by 1 bit won't fit through my e-mail slot, so please forward them to:

Cool CGI Component
c/o Michael B. Klein
Washington Publishing Company
806 W. Diamond Ave., Suite 400
Gaithersburg, MD 20878

The design of the TCGI component was based largely on the CGI.BAS framework written in Visual Basic and provided along with the Windows httpd v1.4 Web Server for Windows 3.1. The WinHTTPD server, the Visual Basic framework, and a good deal of the definitions and explanations which appear in this help file were written by Robert B. Denny. His documentation and comments have been an invaluable aid to the creation of this component, so I feel he deserves a couple shameless plugs:

Windows httpd v1.4a for Windows 3.1 (shareware, $99 commercial licence fee) is available at `http://www.city.net/win-httpd`.

The brand-spankin'-new, 32-bit *WebSite for Windows NT 3.5 and Windows 95* (list price $499) is available from O'Reilly and Associates, Inc. Check out `http://website.ora.com/` for details and ordering information.

Please direct all inquiries about this component to:

Internet: *mbk@baldrick.com*
Compuserve: *74323,3555*

#hlp_aboutcgi
$About the TCGI component