NAME

ncgi − Procedures to manipulate CGI values.

SYNOPSIS

package require **Tcl 8.2**

package require **ncgi ?1.3?**

::ncgi::cookie *cookie*

::ncgi::decode *str*

::ncgi::empty *name*

::ncgi::exists *name*

::ncgi::encode *string*

::ncgi::header ?*type*? *args*

::ncgi::import *cginame* ?*tclname*?

::ncgi::importAll *args*

::ncgi::importFile *cmd cginame* ?*filename*?

::ncgi::input ?*fakeinput*? ?*fakecookie*?

::ncgi::multipart *type query*

::ncgi::nvlist

::ncgi::names

::ncgi::parse

::ncgi::parseMimeValue *value*

::ncgi::query

::ncgi::redirect *url*

::ncgi::reset *query type*

::ncgi::setCookie *args*

::ncgi::setDefaultValue *key defvalue*

::ncgi::setDefaultValueList *key defvaluelist*

::ncgi::setValue *key value*

::ncgi::setValueList *key valuelist*

::ncgi::type

::ncgi::urlStub ?*url*?

::ncgi::value *key* ?*default*?

::ncgi::valueList *key* ?*default*?

DESCRIPTION

The **ncgi** package provides commands that manipulate CGI values. These are values that come from Web forms and are processed either by CGI scripts or web pages with embedded Tcl code. Use the **ncgi** package to query these values, set and get cookies, and encode and decode www-url-encoded values.

In the simplest case, a CGI script first calls **::ncgi::parse** and then calls **::ncgi::value** to get different form values. If a CGI value is repeated, you should use **::ncgi::valueList** to get back the complete list of values.

An alternative to **::ncgi::parse** is **::ncgi::input**, which has semantics similar to Don Libes' **cgi_input** procedure. **::ncgi::input** restricts repeated CGI values to have names that end with "List". In this case, **::ncgi::value** will return the complete list of values, and **::ncgi::input** will raise errors if it find repeated form elements without the right name.

The **::ncgi::reset** procedure can be used in test suites and Web servers to initialize the source of the CGI values. Otherwise the values are read in from the CGI environment.

The complete set of procedures is described below.

::ncgi::cookie *cookie*
: Return a list of values for *cookie*, if any. It is possible that more than one cookie with the same name can be present, so this procedure returns a list.

::ncgi::decode *str*
: Decode strings in www-url-encoding, which represents special characters with a %xx sequence, where xx is the character code in hex.

::ncgi::empty *name*
: Returns 1 if the CGI variable *name* is not present or has the empty string as its value.

::ncgi::exists *name*
: The return value is a boolean. It returns **0** if the CGI variable *name* is not present, and **1** otherwise.

::ncgi::encode *string*
: Encode *string* into www-url-encoded format.

::ncgi::header ?*type*? *args*
: Output the CGI header to standard output. This emits a Content-Type: header and additional headers based on *args*, which is a list of header names and header values. The *type* defaults to "text/html".

::ncgi::import *cginame* ?*tclname*?
: This creates a variable in the current scope with the value of the CGI variable *cginame*. The name of the variable is *tclname*, or *cginame* if *tclname* is empty (default).

::ncgi::importAll *args*
: This imports several CGI variables as Tcl variables. If *args* is empty, then every CGI value is imported. Otherwise each CGI variable listed in *args* is imported.

::ncgi::importFile *cmd cginame* ?*filename*?
: This provides information about an uploaded file from a form input field of type **file** with name *cginame*. *cmd* can be one of **-server -client**, **-type** or **-data**.

  -client *cginame*
  : returns the filename as sent by the client.

-type *cginame*
returns the mime type of the uploaded file.

-data *cginame*
returns the contents of the file.

-server *cginame filename*
writes the file contents to a local temporary file (or *filename* if supplied) and returns the name of the file. The caller is responsible for deleting this file after use.

::ncgi::input ?*fakeinput*? ?*fakecookie*?
This reads and decodes the CGI values from the environment. It restricts repeated form values to have a trailing "List" in their name. The CGI values are obtained later with the **::ncgi::value** procedure.

::ncgi::multipart *type query*
This procedure parses a multipart/form-data *query*. This is used by **::ncgi::nvlist** and not normally called directly. It returns an alternating list of names and structured values. Each structure value is in turn a list of two elements. The first element is meta-data from the multipart/form-data structure. The second element is the form value. If you use **::ncgi::value** you just get the form value. If you use **::ncgi::valueList** you get the structured value with meta data and the value.

The *type* is the whole Content-Type, including the parameters like *boundary*. This returns a list of names and values that describe the multipart data. The values are a nested list structure that has some descriptive information first, and the actual form value second. The descriptive information is list of header names and values that describe the content.

::ncgi::nvlist
This returns all the query data as a name, value list. In the case of multipart/form-data, the values are structured as described in **::ncgi::multipart**.

::ncgi::names
This returns all names found in the query data, as a list. **::ncgi::multipart**.

::ncgi::parse
This reads and decodes the CGI values from the environment. The CGI values are obtained later with the **::ncgi::value** procedure. IF a CGI value is repeated, then you should use **::ncgi::valueList** to get the complete list of values.

::ncgi::parseMimeValue *value*
This decodes the Content-Type and other MIME headers that have the form of "primary value; param=val; p2=v2" It returns a list, where the first element is the primary value, and the second element is a list of parameter names and values.

::ncgi::query
This returns the raw query data.

::ncgi::redirect *url*
Generate a response that causes a 302 redirect by the Web server. The *url* is the new URL that is the target of the redirect. The URL will be qualified with the current server and current directory, if necessary, to convert it into a full URL.

::ncgi::reset *query type*
Set the query data and Content-Type for the current CGI session. This is used by the test suite and by Web servers to initialize the ncgi module so it does not try to read standard input or use environment variables to get its data. If neither *query* or *type* are specified, then the **ncgi** module will look in the standard CGI environment for its data.

::ncgi::setCookie *args*
Set a cookie value that will be returned as part of the reply. This must be done before **::ncgi::header** or **::ncgi::redirect** is called in order for the cookie to be returned properly. The *args* are a set of flags and values:

-name *name*

-value *value*

-expires *date*

-path *path restriction*

-domain *domain restriction*

::ncgi::setDefaultValue *key defvalue*
Set a CGI value if it does not already exists. This affects future calls to **::ncgi::value** (but not future calls to **::ncgi::nvlist**). If the CGI value already is present, then this procedure has no side effects.

::ncgi::setDefaultValueList *key defvaluelist*
Like **::ncgi::setDefaultValue** except that the value already has list structure to represent multiple checkboxes or a multi-selection.

::ncgi::setValue *key value*
Set a CGI value, overriding whatever was present in the CGI environment already. This affects future calls to **::ncgi::value** (but not future calls to **::ncgi::nvlist**).

::ncgi::setValueList *key valuelist*
Like **::ncgi::setValue** except that the value already has list structure to represent multiple checkboxes or a multi-selection.

::ncgi::type
Returns the Content-Type of the current CGI values.

::ncgi::urlStub ?*url*?
Returns the current URL, but without the protocol, server, and port. If *url* is specified, then it defines the URL for the current session. That value will be returned by future calls to **::ncgi::urlStub**

::ncgi::value *key* ?*default*?
Return the CGI value identified by *key*. If the CGI value is not present, then the *default* value is returned instead. This value defaults to the empty string.

If the form value *key* is repeated, then there are two cases: if **::ncgi::parse** was called, then **::ncgi::value** only returns the first value associated with *key*. If **::ncgi::input** was called, then **::ncgi::value** returns a Tcl list value and *key* must end in "List" (e.g., "skuList"). In the case of multipart/form-data, this procedure just returns the value of the form element. If you want the meta-data associated with each form value, then use **::ncgi::valueList**.

::ncgi::valueList *key* ?*default*?
Like **::ncgi::value**, but this always returns a list of values (even if there is only one value). In the case of multipart/form-data, this procedure returns a list of two elements. The first element is meta-data in the form of a parameter, value list. The second element is the form value.

EXAMPLES

Uploading a file

HTML:

```
<html>
<form action="/cgi-bin/upload.cgi" method="POST" enctype="multipart/form-data">
Path: <input type="file" name="filedata"><br>
Name: <input type="text" name="filedesc"><br>
<input type="submit">
</form>
</html>
```

TCL: upload.cgi

```
#!/usr/local/bin/tclsh
```

```
::ncgi::parse
set filedata [::ncgi::value filedata]
set filedesc [::ncgi::value filedesc]

puts "<html> File uploaded at <a href=\"/images/$filedesc\">$filedesc</a> </html>"

set filename /www/images/$filedesc

set fh [open $filename w]
puts -nonewline $fh $filedata
close $fh
```

BUGS, IDEAS, FEEDBACK

This document, and the package it describes, will undoubtedly contain bugs and other problems. Please report such in the category *ncgi* of the *Tcllib SF Trackers* [http://sourceforge.net/tracker/?group_id=12883]. Please also report any ideas for enhancements you may have for either package and/or documentation.

SEE ALSO

html

KEYWORDS

CGI, cookie, form, html

CATEGORY

CGI programming