Specification of Cgipp

1 Summary *class Cgipp*

Declaration:
class **Cgipp**;

Based on: cgi/CgiEnv.H and cgi/htmlTool.H

Properties: Cgipp is meant to be an easy-to-use class

2 Purpose

The class' purpose is to provide an easy-to-use interface to CGI-programming and let everybody with knowledge of C++ make CGI-scripts with ease. The class contains few public member-functions to retrieve the data from the web-pages and hides the more complex functions.

3 Analysis

3.1 Data Structure

class *Cgipp* {
FormEntry *itsFirstEntry;
CgiEnv *itsEnv;
};

4 Description

4.1 Types

The class contains a pointer to the CgiEnv class and a pointer to the FormEntry class.

4.2 Operations

4.2.1 Basic Methods

constructor **Cgipp** ()
Description: Constructs the class and initializes all variables and the CGI-Environment.

destructor **~Cgipp** ()
Description: Deletes any variables allocated by the class in it's lifetime.

4.2.2 Public Methods

CgiEnv* function **getCgiEnv** (void) **const**
Description: Returns a pointer to the CGI-Environment.

char * function **getvar** (char *name, int max, int &found, int newlines=1)
Description: Returns a char-pointer to the value of the cgi-variable with name *name* . If the variable is not found, the method returns NULL and sets *found* to 0. The max-argument states the maximum-size of the value.

char** function **getmultivar** (char *name, int &numFound)
Description: Returns a double char-pointer to the value of the cgi-variables with the name *name* . If the variable is not found, the method returns NULL and sets *numFound* to 0. The numFound-argument returns the number of variables found.

int function **getint** (char *name, int &found)
Description: Returns an integer with the value of the cgi-variable with name *name* . If the variable is not found, the method returns 0 and sets *found* to 0. the value.

double function **getdouble** (char *name, int &found)
Description: Returns a float-variable with the value of the cgi-variable with name *name* . If the variable is not found, the method returns 0.0 and sets *found* to 0.

4.2.3 Protected Methods

FormEntry* function **findEntry** (**const** char *name) **const**
Description: For internal use

FormEntry* function **findNextEntry** (**const** char *findTarget, FormEntry*findPos) **const**
Description: For internal use

int function **makeString** (FormEntry*e, char *result, int max, int newlines) **const**
Description: For internal use

int function **unEscapeChars** (char**sp, **const** char *cp, int len) **const**
Description: For internal use

int function **firstNonspaceChar** (**const** char *s) **const**
Description: For internal use

int function **getFormInput** (void)
Description: For internal use

int function **postFormInput** (void)
Description: For internal use

int function **parseFormInput** (**const** char *data, int length)
Description: For internal use

FormEntry* function **getFirstEntry** (void) **const**
Description: For internal use

5 Debugging

A major expansion from Cgicc v1.0 is the debugging-system, which consists of several easy-to-use macros. These are also described in the file **test.cpp** also included in the Cgipp v0.3 distribution but the main macros are (all debugging can be crippled by defining **NODEBUG**):
DEBUGPUT(arg) - which sends *arg* to the debug-file
_DEBUGPUT(myostream, arg) - which sends *arg* to the given ostream
TRACE(arg) - which sends a traced version of *arg* to the debug-file
_TRACE(myostream, arg) - which sends a traced version of *arg* to the given ostream
LINEPUT() - which sends a line to the debug-file

6 Bailing out

When you need to bail out and get a full error-report, it is recommended that you use the macro:
_CgiError(arg) - which will give a full error-message to the user, with *arg* as the main reason for the Error. This macro should be tested. It will also write output to the debug-file to let you view what really happened.

`Document created with SDS v0.20`