Ogród zoologiczny

Generated by Doxygen 1.8.14

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

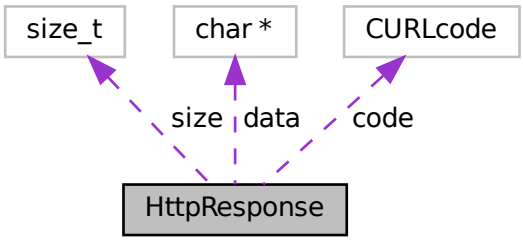
Chapter 3

Data Structure Documentation

3.1 HttpResponse Struct Reference

`#include <http.h>`

Collaboration diagram for HttpResponse:



Data Fields

- char * data
- size_t size
- CURLcode code

3.1.1 Detailed Description

Definition at line 8 of file http.h.

3.1.2 Field Documentation

3.1.2.1 code

```
CURLcode HttpResponse::code
```

Definition at line 11 of file http.h.

Referenced by http_get().

3.1.2.2 data

```
char* HttpResponse::data
```

Definition at line 9 of file http.h.

Referenced by http_get(), http_response_new(), and write_function().

3.1.2.3 size

```
size_t HttpResponse::size
```

Definition at line 10 of file http.h.

Referenced by http_response_new(), and write_function().

The documentation for this struct was generated from the following file:

- include/services/http.h

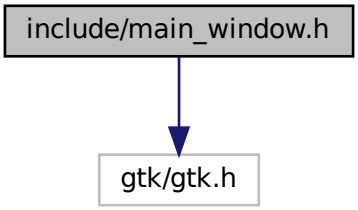
Chapter 4

File Documentation

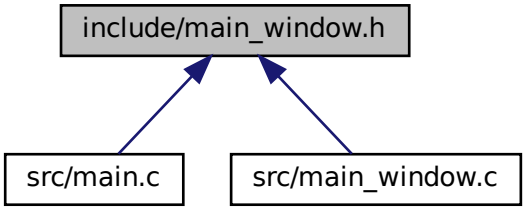
4.1 include/main_window.h File Reference

```
#include <gtk/gtk.h>
```

Include dependency graph for main_window.h:



This graph shows which files directly or indirectly include this file:



Functions

- GtkWidget * main_window_new (GtkApplication *app)

4.1.1 Function Documentation

4.1.1.1 main_window_new()

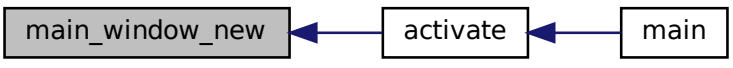
```
GtkWidget* main_window_new (
            GtkApplication * app )
```

Definition at line 7 of file main_window.c.

Referenced by activate().

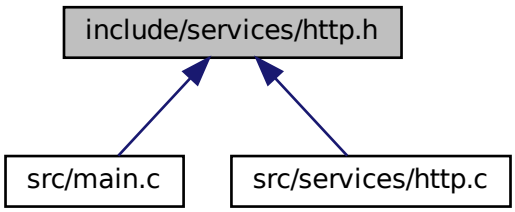
```
7                                                {
8     GtkWidget *window;
9     GtkWidget *mainContainer;
10
11     window = gtk_application_window_new (app);
12     gtk_window_set_title (GTK_WINDOW (window), "Ogród zoologiczny");
13     gtk_window_set_default_size (GTK_WINDOW (window), 500, 500);
14
15     mainContainer = gtk_grid_new();
16     gtk_container_add(GTK_CONTAINER (window), GTK_WIDGET (mainContainer));
17
18     GtkWidget *containerTable;
19     containerTable = gtk_scrolled_window_new(NULL, NULL);
20     gtk_widget_set_hexpand(containerTable, 1);
21     gtk_widget_set_vexpand(containerTable, 1);
22     GtkWidget *table;
23     table = gtk_grid_new();
24     gtk_container_add(GTK_CONTAINER(containerTable), GTK_WIDGET(table));
25     for(int i=0; i<200; ++i){
26         GtkWidget *label = gtk_label_new("Test");
27         gtk_label_set_xalign(GTK_LABEL (label), 0.0);
28         gtk_widget_set_margin_start(label, 5);
29         gtk_widget_set_margin_top(label, 5);
30         gtk_widget_set_hexpand(label, TRUE);
31         gtk_grid_attach(GTK_GRID (table), label, 0, i, 1, 1);
32     }
33
34     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (containerTable), 0, 1, 1, 1);
35
36     GtkWidget *buttonAddAnimal;
37     buttonAddAnimal = gtk_button_new();
38     gtk_button_set_label(GTK_BUTTON (buttonAddAnimal), "Dodaj zwierzę");
39     gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonAddAnimal), 0, 2, 1, 1);
40     gtk_widget_set_hexpand(buttonAddAnimal, 1);
41     gtk_widget_show_all (window);
42 }
```

Here is the caller graph for this function:



4.2 include/services/http.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct HttpResponse

Typedefs

- typedef struct HttpResponse HttpResponse

Functions

- size_t write_function (void *ptr, size_t size, size_t nmemb, HttpResponse *r)
- HttpResponse * http_get (char *url)

  Gets HTTP response for url.

4.2.1 Typedef Documentation

4.2.1.1 HttpResponse

```
typedef struct HttpResponse HttpResponse
```

4.2.2 Function Documentation

4.2.2.1 http_get()

```
HttpResponse* http_get (
            char * url )
```

Gets HTTP response for url.

Parameters

url	Site address

Returns

Filled HttpResponse

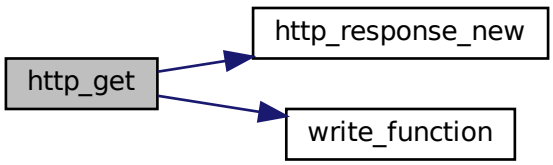
Definition at line 33 of file http.c.

References HttpResponse::code, HttpResponse::data, http_response_new(), and write_function().

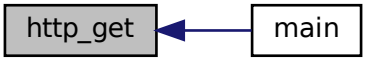
Referenced by main().

```
33                                      {
34     CURL *curl;
35     HttpResponse * res = http_response_new();
36
37     curl = curl_easy_init();
38     if(curl) {
39         puts(url);
40         curl_easy_setopt(curl, CURLOPT_URL, url);
41         /* example.com is redirected, so we tell libcurl to follow redirection */
42         curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
43         curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_function);
44         curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);
45
46         /* Perform the request, res will get the return code */
47         res->code = curl_easy_perform(curl);
48         /* Check for errors */
49         if(res->code != CURLE_OK)
50             fprintf(stderr, "curl_easy_perform() failed: %s\n",
51                     curl_easy_strerror(res->code));
52         else {
53             printf("Data: %s\n", res->data);
54         }
55
56         /* always cleanup */
57         curl_easy_cleanup(curl);
58     }
59     return res;
60 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.2 write_function()

```
size_t write_function (
            void * ptr,
            size_t size,
            size_t nmemb,
            HttpResponse * r )
```

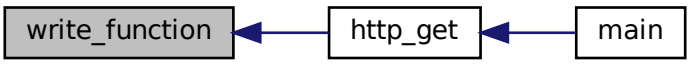
Definition at line 12 of file http.c.

References HttpResponse::data, and HttpResponse::size.

Referenced by http_get().

```
12                                                                      {
13     size_t new_len = r->size + size*nmemb;
14     r->data= realloc(r->data, new_len+1);
15     memcpy(r->data+r->size, ptr, size*nmemb);
16     r->data[new_len] = '\0';
17     return size*nmemb;
18 }
```

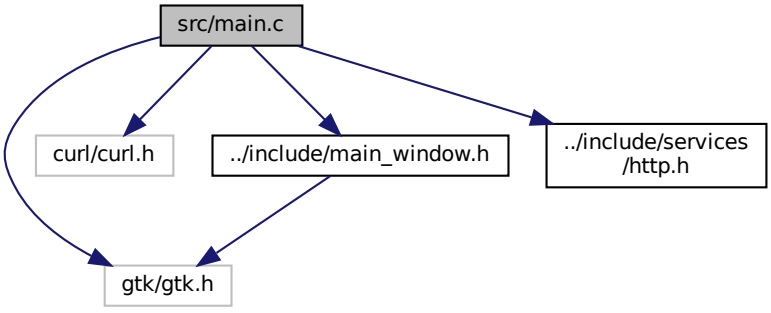
Here is the caller graph for this function:



4.3 src/main.c File Reference

```
#include <gtk/gtk.h>
#include <curl/curl.h>
#include "../include/main_window.h"
#include "../include/services/http.h"
```

Include dependency graph for main.c:



Functions

- static void activate (GtkApplication *app, gpointer user_data)

  Initialize UI.
- int main (int argc, char **argv)

4.3.1 Function Documentation

4.3.1.1 activate()

```
static void activate (
            GtkApplication * app,
            gpointer user_data )  [static]
```

Initialize UI.

Definition at line 10 of file main.c.

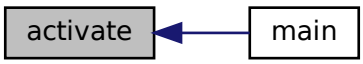
References main_window_new().

Referenced by main().

```
11 {
12     main_window_new(app);
13 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



4.3.1.2 main()

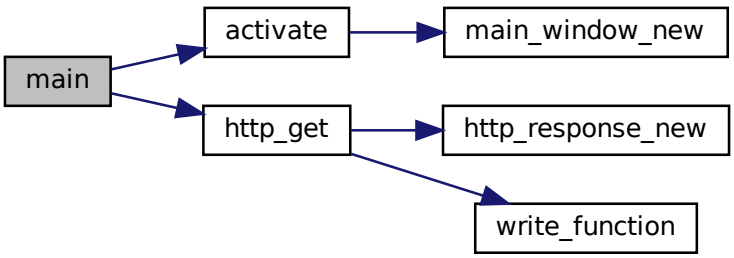
```
int main (
              int argc,
              char ** argv )
```

Definition at line 15 of file main.c.

References activate(), and http_get().

```
15                                {
16     GtkApplication *app;
17
18     http_get("https://mrokita.pl/");
19     int status;
20     app = gtk_application_new ("pl.mrokita.pri3", G_APPLICATION_FLAGS_NONE);
21     g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
22     status = g_application_run (G_APPLICATION (app), argc, argv);
23     g_object_unref (app);
24
25     return status;
26 }
```

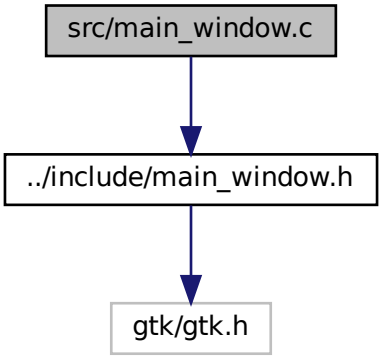
Here is the call graph for this function:



4.4 src/main_window.c File Reference

```
#include "../include/main_window.h"
```

Include dependency graph for main_window.c:



Functions

- GtkWidget * main_window_new (GtkApplication *app)

4.4.1 Function Documentation

4.4.1.1 main_window_new()

```
GtkWidget* main_window_new (
            GtkApplication * app )
```

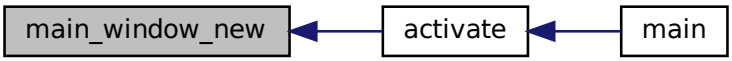
Definition at line 7 of file main_window.c.

Referenced by activate().

```
7                                                {
8     GtkWidget *window;
9     GtkWidget *mainContainer;
10
11     window = gtk_application_window_new (app);
12     gtk_window_set_title (GTK_WINDOW (window), "Ogród zoologiczny");
13     gtk_window_set_default_size (GTK_WINDOW (window), 500, 500);
14
15     mainContainer = gtk_grid_new();
16     gtk_container_add(GTK_CONTAINER (window), GTK_WIDGET (mainContainer));
17
18     GtkWidget *containerTable;
19     containerTable = gtk_scrolled_window_new(NULL, NULL);
20     gtk_widget_set_hexpand(containerTable, 1);
21     gtk_widget_set_vexpand(containerTable, 1);
22     GtkWidget *table;
23     table = gtk_grid_new();
24     gtk_container_add(GTK_CONTAINER(containerTable), GTK_WIDGET(table));
25     for(int i=0; i<200; ++i){
26         GtkWidget *label = gtk_label_new("Test");
27         gtk_label_set_xalign(GTK_LABEL (label), 0.0);
```

```
28          gtk_widget_set_margin_start(label, 5);
29          gtk_widget_set_margin_top(label, 5);
30          gtk_widget_set_hexpand(label, TRUE);
31          gtk_grid_attach(GTK_GRID (table), label, 0, i, 1, 1);
32      }
33
34      gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (containerTable), 0, 1, 1, 1);
35
36      GtkWidget *buttonAddAnimal;
37      buttonAddAnimal = gtk_button_new();
38      gtk_button_set_label(GTK_BUTTON (buttonAddAnimal), "Dodaj zwierzę");
39      gtk_grid_attach(GTK_GRID (mainContainer), GTK_WIDGET (buttonAddAnimal), 0, 2, 1, 1);
40      gtk_widget_set_hexpand(buttonAddAnimal, 1);
41      gtk_widget_show_all (window);
42  }
```

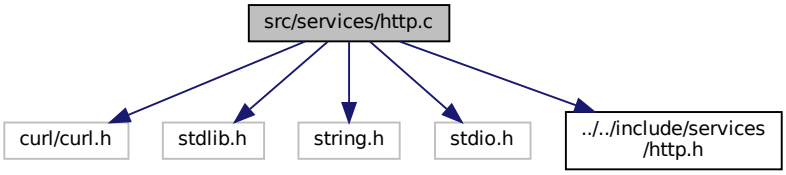
Here is the caller graph for this function:



4.5 src/services/http.c File Reference

```
#include <curl/curl.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "../../include/services/http.h"
```

Include dependency graph for http.c:



Functions

- size_t write_function (void *ptr, size_t size, size_t nmemb, HttpResponse *r)
- HttpResponse * http_response_new ()
- HttpResponse * http_get (char *url)

  Gets HTTP response for url.

4.5.1 Function Documentation

4.5.1.1 http_get()

```
HttpResponse* http_get (
            char * url )
```

Gets HTTP response for url.

Parameters

url	Site address

Returns

Filled HttpResponse

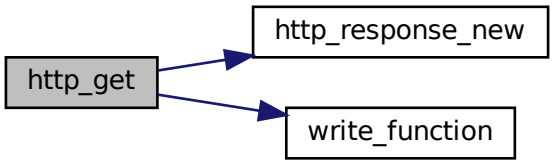
Definition at line 33 of file http.c.

References HttpResponse::code, HttpResponse::data, http_response_new(), and write_function().

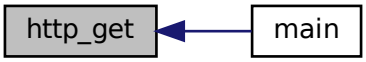
Referenced by main().

```
33                                  {
34     CURL *curl;
35     HttpResponse * res = http_response_new();
36
37     curl = curl_easy_init();
38     if(curl) {
39         puts(url);
40         curl_easy_setopt(curl, CURLOPT_URL, url);
41         /* example.com is redirected, so we tell libcurl to follow redirection */
42         curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
43         curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_function);
44         curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);
45
46         /* Perform the request, res will get the return code */
47         res->code = curl_easy_perform(curl);
48         /* Check for errors */
49         if(res->code != CURLE_OK)
50             fprintf(stderr, "curl_easy_perform() failed: %s\n",
51                     curl_easy_strerror(res->code));
52         else {
53             printf("Data: %s\n", res->data);
54         }
55
56         /* always cleanup */
57         curl_easy_cleanup(curl);
58     }
59     return res;
60 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.1.2 http_response_new()

```
HttpResponse* http_response_new ( )
```

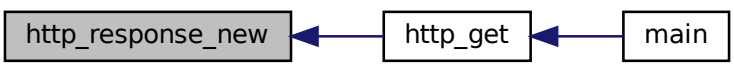
Definition at line 20 of file http.c.

References HttpResponse::data, and HttpResponse::size.

Referenced by http_get().

```
20                                  {
21     HttpResponse * res = malloc(sizeof(HttpResponse));
22     res->size = 0;
23     res->data = malloc(res->size+1);
24     res->data[res->size] = '\0';
25     return res;
26 };
```

Here is the caller graph for this function:



4.5.1.3 write_function()

```
size_t write_function (
             void * ptr,
             size_t size,
             size_t nmemb,
             HttpResponse * r )
```

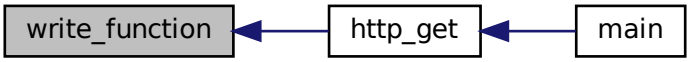
Definition at line 12 of file http.c.

References HttpResponse::data, and HttpResponse::size.

Referenced by http_get().

```
12                                                                      {
13     size_t new_len = r->size + size*nmemb;
14     r->data= realloc(r->data, new_len+1);
15     memcpy(r->data+r->size, ptr, size*nmemb);
16     r->data[new_len] = '\0';
17     return size*nmemb;
18 }
```

Here is the caller graph for this function:



Index