json_fastcgi_web_api

Generated by Doxygen 1.8.17

1 jQuery <--> C++ fastcgi web API

C++ Header-only event driven communication between jQuery in a web-browser via nginx.

This was developed because of a lack of a lightweight jQuery to C++ communication. It's a small helper which can easily be included in any C++ application which needs to talk to a web page where realtime data needs to be exchanged.

1.1 Prerequisites

```
apt-get install libfcgi-dev
apt-get install libcurl4-openssl-dev
```

1.2 Howto

The only file you need is:
`json_fastcgi_web_api.h`

Copy this header file into your project, include it and then overload the abstract callbacks in the class.

1.2.1 Implement the GET callback

This is the callback which sends JSON to the website:

```
class GETCallback {
public:
        /**
         * Needs to return the payload data sent to the web browser.
         * Use the JSON generator class or an external json generator.
         **/
        virtual std::string getJSONString() = 0;
};
```

Overload `getJSONString()` and return JSON. You can use the class `JSONGenerator` to generate the JSON data: Use the `add` methods to add key/value pairs and then get the json with the method `getJSON()`.

1.2.2 Implement the POST callback (optional)

This handler receives the JSON from jQuery POST command from the website for example a button press. Implement the callback:

```
class POSTCallback {
public:
        /**
         * Receives the data from the web browser in JSON format.
         * Use postDecoder() to decode the JSON or use an external
         * library.
         **/
        virtual void postString(std::string arg) = 0;
};
```

Overload `postString(std::string arg)` with a function which decodes the received POST data. You can use `postDecoder(std::string s)` which returns a `std::map` of key/value pairs.

1.2.3 Start the communication

The constructor takes as arguments the GET callback, the POST callback and the path to the fastCGI socket. As soon as the constructor is instantiated the communication starts.

```
/**
  * Constructor which inits it and starts the main thread.
  * Provide an instance of the callback handler which provides the
  * payload data in return. The optional socketpath variable
  * can be set to another path for the socket which talks to the
  * webserver. postCallback is the callback which returns
  * received json packets as a map.
  **/
 JSONCGIHandler(GETCallback* argGetCallback,
                POSTCallback* argPostCallback = nullptr,
                const char socketpath[] = "/tmp/fastcgisocket");
```

1.2.4 Stop the communication

This is done by deleting the instance.

1.3 Example code

The `demo_sensor_server` fakes a temperature sensor and this is plotted on the screen. The nginx config file and the website are in the `website` directory.

Start `demo_sensor_server` in the background with:

```
nohup ./demo_sensor_server &
```

which creates a socket under `/tmp/sensorsocket` to communicate with the fastcgi server.

1.3.1 Configuring the nginx for FastCGI

1. copy the the nginx config file `website/nginx-sites-enabled-default` to your nginx config directory `/etc/nginx/sites-enabled/default`.
2. copy `website/fakesensor.html` to `/var/www/html`.

Then point your web-browser to `fakesensor.html` on your website. You should see a fake temperatue reading on the screen and a plot with dygraph. The JSON packets can be viewed by appending `/sensor/` to the server URL.

The script sends also a JSON packet to the demo server which requests to clamp the temperature to 20C and prints out a string to stderr.

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 Class Documentation

3.1 JSONCGIHandler::GETCallback Class Reference

```
#include <json_fastcgi_web_api.h>
```

Public Member Functions

- virtual std::string getJSONString ()=0
- virtual std::string getContentType ()

3.1.1 Detailed Description

GET callback handler which needs to be implemented by the main program. This needs to provide the JSON payload as a string either by using the simple JSONGenerator below or by an external library.

3.1.2 Member Function Documentation

3.1.2.1 getContentType() `virtual std::string JSONCGIHandler::GETCallback::getContentType ( ) [inline], [virtual]`

The content type of the payload. That's by default "application/json" but can be overloaded if needed.

Returns

MIME type

3.1.2.2 getJSONString() `virtual std::string JSONCGIHandler::GETCallback::getJSONString ( ) [pure virtual]`

Needs to return the payload data sent to the web browser. Use the JSONGenerator to create the JSON or use an external json generator.

Returns

JSON data

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

- json_fastcgi_web_api.h

3.2 JSONCGIHandler Class Reference

```
#include <json_fastcgi_web_api.h>
```

Classes

- class GETCallback
- class JSONGenerator
- class POSTCallback

Public Member Functions

- JSONCGIHandler (GETCallback ∗argGetCallback, POSTCallback ∗argPostCallback=nullptr, const char socketpath[ ]="/tmp/fastcgisocket")
- ∼JSONCGIHandler ()

Static Public Member Functions

- static std::map< std::string, std::string > postDecoder (std::string s)

3.2.1 Detailed Description

C++ wrapper around fastCGI which sends and receives JSON in a jQuery friendly format.

Copyright (C) 2021 Bernd Porr `mail@berndporr.me.uk` Apache License 2.0

3.2.2 Constructor & Destructor Documentation

3.2.2.1 JSONCGIHandler()

```
JSONCGIHandler::JSONCGIHandler (
        GETCallback * argGetCallback,
        POSTCallback * argPostCallback = nullptr,
        const char socketpath[] = "/tmp/fastcgisocket" )  [inline]
```

Constructor which opens the connection and starts the main thread. Provide an instance of the callback handler which returns the payload data. argPostCallback is the callback which returns received json packets as a map. The optional socketpath variable can be set to another path for the socket which talks to the webserver.

Parameters

argGetCallback	Callback handler for sending JSON
argPostCallback	Callback handler for receiving JSON
socketpath	Path of the socket which communicates to the webserver

3.2.2.2 ∼JSONCGIHandler()

```
JSONCGIHandler::∼JSONCGIHandler ( )  [inline]
```

Destructor which shuts down the connection to the webserver and it also terminates the thread which is waiting for requests.

3.2.3 Member Function Documentation

3.2.3.1 postDecoder() `static std::map<std::string,std::string> JSONCGIHandler::postDecoder ( std::string s ) [inline], [static]`

Parses a POST string and returns a std::map with key/value pairs. It also converts back any xx style encoding back to chars using libcurl. Note this is a simple parser and it won't deal with nested JSON structures.

Parameters

s	The POST string to be decoded.

Returns

A std::map which conains the key/value pairs.

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

- json_fastcgi_web_api.h

3.3 JSONCGIHandler::JSONGenerator Class Reference

```
#include <json_fastcgi_web_api.h>
```

Public Member Functions

- void add (std::string key, std::string value)
- void add (std::string key, double value)
- void add (std::string key, float value)
- void add (std::string key, long value)
- void add (std::string key, int value)
- std::string getJSON ()

3.3.1 Detailed Description

Simple helper function to create a key/value json pairs for the callback function.

3.3.2 Member Function Documentation

3.3.2.1 add() [1/5] `void JSONCGIHandler::JSONGenerator::add ( std::string key, double value ) [inline]`

Adds a JSON entry: double

Parameters

key	The JSON key
value	The JSON value as a double

3.3.2.2 add() [2/5]

```
void JSONCGIHandler::JSONGenerator::add (
        std::string key,
        float value ) [inline]
```

Adds a JSON entry: float

Parameters

key	The JSON key
value	The JSON value as a float

3.3.2.3 add() [3/5]

```
void JSONCGIHandler::JSONGenerator::add (
        std::string key,
        int value ) [inline]
```

Adds a JSON entry: int

Parameters

key	The JSON key
value	The JSON value as an int

3.3.2.4 add() [4/5]

```
void JSONCGIHandler::JSONGenerator::add (
        std::string key,
        long value ) [inline]
```

Adds a JSON entry: long int

Parameters

key	The JSON key
value	The JSON value as a long int

3.3.2.5 add() [5/5]

```
void JSONCGIHandler::JSONGenerator::add (
            std::string key,
            std::string value )  [inline]
```

Adds a JSON entry: string

Parameters

key	The JSON key
value	The JSON value as a string

3.3.2.6 getJSON() `std::string JSONCGIHandler::JSONGenerator::getJSON ( )  [inline]`

Gets the json string

Returns

The JSON data ready to be sent

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

- json_fastcgi_web_api.h

3.4 JSONCGIHandler::POSTCallback Class Reference

`#include <json_fastcgi_web_api.h>`

Public Member Functions

- virtual void postString (std::string postArg)=0

3.4.1 Detailed Description

Callback handler which needs to be implemented by the main program.

3.4.2 Member Function Documentation

3.4.2.1 postString() `virtual void JSONCGIHandler::POSTCallback::postString (`

```
            std::string postArg )  [pure virtual]
```

Receives the POST data from the web browser. Use postDecoder() to decode the postArg string.

Parameters

postArg	POST data received from jQuery

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

- json_fastcgi_web_api.h

Index