Cjango-Unchained

Generated by Doxygen 1.8.13

# **Contents**

1	Cjan	igo-Und	hained													1
	1.1	Introdu	uction							 	 	 	 	 	 	1
2	Bug	List														3
3	Hier	archica	l Index													5
	3.1	Class	Hierarchy							 	 	 	 	 	 	5
4	Clas	s Index	(													7
	4.1	Class	List							 	 	 	 	 	 	7
5	File	Index														9
	5.1	File Lis	st							 	 	 	 	 	 	9
6	Clas	s Docu	mentatior	n												11
	6.1	App C	lass Refere	ence						 	 	 	 	 	 	11
		6.1.1	Member	Func	tion D	ocum	entati	ion .		 	 	 	 	 	 	11
			6.1.1.1	add	d_mon	itored	l_dir()			 	 	 	 	 	 	11
			6.1.1.2	han	ndle_re	eques	t()			 	 	 	 	 	 	12
			6.1.1.3	run	()					 	 	 	 	 	 	12
			6.1.1.4	wor	rker()					 	 	 	 	 	 	13
	6.2	http_se	ession_ge	t_exc	eption	Clas	s Refe	erenc	е.	 	 	 	 	 	 	13
		6.2.1	Detailed	Desc	ription	۱				 	 	 	 	 	 	13
	6.3	http::H	ttpReques	st Cla	ss Ref	ferenc	е			 	 	 	 	 	 	14
		6.3.1	Construc	ctor &	Destr	uctor	Docu	menta	ation	 	 	 	 	 	 	14
			6.3.1.1	Http	oRequ	est()				 	 	 	 	 	 	15

ii CONTENTS

	6.3.2	Member	Function Documentation	 15
		6.3.2.1	get_cookie()	 15
		6.3.2.2	get_meta()	 15
		6.3.2.3	get_method()	 16
		6.3.2.4	get_parameters()	 16
		6.3.2.5	get_path()	 16
		6.3.2.6	get_scheme()	 16
		6.3.2.7	get_session()	 17
		6.3.2.8	xorshf96()	 17
6.4	http::H	ttpReques	stBodyParser Class Reference	 17
	6.4.1	Construc	ctor & Destructor Documentation	 17
		6.4.1.1	HttpRequestBodyParser()	 18
	6.4.2	Member	Function Documentation	 18
		6.4.2.1	parse()	 18
	6.4.3	Member	Data Documentation	 18
		6.4.3.1	url_encoded_form_parsers	 18
6.5	http::H	ttpReques	stLine Class Reference	 19
	6.5.1	Construc	ctor & Destructor Documentation	 19
		6.5.1.1	HttpRequestLine() [1/2]	 19
		6.5.1.2	HttpRequestLine() [2/2]	 19
6.6	http::H	ttpReques	stParser Class Reference	 20
	6.6.1	Member	Function Documentation	 20
		6.6.1.1	parse()	 20
		6.6.1.2	parse_body()	 21
		6.6.1.3	parse_request_line_and_headers()	 21
6.7	http::H	ttpRespon	nse Class Reference	 22
	6.7.1	Construc	ctor & Destructor Documentation	 23
		6.7.1.1	HttpResponse() [1/4]	 23
		6.7.1.2	HttpResponse() [2/4]	 23
		6.7.1.3	HttpResponse() [3/4]	 23

CONTENTS

		6.7.1.4	HttpResponse() [4/4]	24
	6.7.2	Member	Function Documentation	24
		6.7.2.1	get_template()	24
		6.7.2.2	render_to_response() [1/4]	25
		6.7.2.3	render_to_response() [2/4]	25
		6.7.2.4	render_to_response() [3/4]	25
		6.7.2.5	render_to_response() [4/4]	26
		6.7.2.6	set_cookie()	26
6.8	http::H	ttpSession	Class Reference	27
	6.8.1	Member	Function Documentation	27
		6.8.1.1	get()	27
6.9	http::H	ttpStreamF	Reader Class Reference	27
	6.9.1	Member	Function Documentation	28
		6.9.1.1	eat_white_space()	28
		6.9.1.2	get_next_line()	28
		6.9.1.3	read()	29
		6.9.1.4	to_string()	29
6.10	MSock	et Class R	eference	29
6.11	Router	Class Ref	erence	30
	6.11.1	Member	Function Documentation	30
		6.11.1.1	add_route()	30
		6.11.1.2	get_http_response()	31
		6.11.1.3	resolve()	31
6.12	Selecto	or Struct R	eference	31
6.13	http::U	rlEncoded	FormParser Class Reference	32
	6.13.1	Member	Function Documentation	32
		6.13.1.1	can_parse_content_type()	32
		6.13.1.2	get_parameter()	33
		6.13.1.3	split()	33

iv CONTENTS

7	File I	Docume	ntation	35
	7.1	http_pa	rser/http_request.cpp File Reference	35
		7.1.1	Detailed Description	35
	7.2	http_pa	rser/http_request.hpp File Reference	35
		7.2.1	Detailed Description	36
		7.2.2	Function Documentation	36
			7.2.2.1 operator<<()	36
	7.3	http_pa	rser/http_request_body_parser.cpp File Reference	36
		7.3.1	Detailed Description	36
	7.4	http_pa	rser/http_request_body_parser.hpp File Reference	37
		7.4.1	Detailed Description	37
	7.5	http_pa	rser/http_request_line.cpp File Reference	37
		7.5.1	Detailed Description	37
	7.6	http_pa	rser/http_request_line.hpp File Reference	37
		7.6.1	Detailed Description	37
	7.7	http_pa	rser/http_request_parser.cpp File Reference	38
		7.7.1		38
	7.8	http pa		38
		7.8.1		38
	7.9	http pa		38
		7.9.1		39
		7.9.2		39
				39
	7 10	httn na	· - · · · · · · · · · · · · · · · · · ·	39
	7.10	. —	·- ·	40
	7 11		·	40
				40
	7 10			40
	7.12			40
	7 12		•	41
	7.10			41
	711			41
	7.14			41
	7 4 5		and the property of the proper	
	7.15		record Zer errechZer (Zer errecht)	41
	7.10		·	41
	7.16			42
	7 4 7		•	42
	7.17			42
		7.17.1	Detailed Description	43
Inc	dex			45

## **Chapter 1**

# Cjango-Unchained

## 1.1 Introduction

The Cjango-Unchained is a lightweight C++ web app framework. This documentation describes Cjango's internal programming interfaces, including ones which you will use when you write your original callback functions.

2 Cjango-Unchained

## **Chapter 2**

# **Bug List**

## Member App::run (int port)

if the number of opened sockets exceeds SOMAXCONN, Cjango gets "apr\_socket\_recv: Connection reset by peer (54)" error and halts immediately. On Mac OS X (default: 128), check by "sysctl -a | grep somax" and change it by "sudo sysctl -w kern.ipc.somaxconn=2048"

4 Bug List

# **Chapter 3**

# **Hierarchical Index**

## 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

App	1
exception	
http_session_get_exception	3
http::HttpRequest	4
http::HttpRequestBodyParser	7
http::HttpRequestLine	9
http::HttpRequestParser	
http::HttpResponse	
http::HttpSession	7
http::HttpStreamReader	
MSocket	
Router	
Selector	1
http::/urlEncodedFormParser 3	2

6 Hierarchical Index

# **Chapter 4**

# **Class Index**

## 4.1 Class List

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

App
http_session_get_exception
Custom exception class in case HttpSession::get wants to hard-fail when key is not present 13
http::HttpRequest
http::HttpRequestBodyParser
http::HttpRequestLine
http::HttpRequestParser
http::HttpResponse
http::HttpSession
http::HttpStreamReader
MSocket
Router
Selector
http::UrlEncodedFormParser

8 Class Index

# **Chapter 5**

# File Index

## 5.1 File List

Here is a list of all documented files wit	h brief descriptio	ns
--	--------------------	----

арр/ <b>арр.hpp</b>	??
app/ <b>externs.hpp</b>	??
app/ <b>msocket.hpp</b>	??
app/ <b>selector.hpp</b>	??
http_parser/http_request.cpp	
HttpRequest class implementation	35
http_parser/http_request.hpp	
HttpRequest class declaration	35
http_parser/http_request_body_parser.cpp	
HttpRequestBodyParser class implementation	36
http_parser/http_request_body_parser.hpp	
HttpRequestBodyParser class declaration	37
http_parser/http_request_line.cpp	
HttpRequestLine class implementation	37
http_parser/http_request_line.hpp	
HttpRequestLine class declaration	37
http parser/http request parser.cpp	
HttpRequestParser class implementation	38
http_parser/http_request_parser.hpp	
HttpRequestParser class declaration	38
http_parser/http_response.cpp	
HttpResponse class implementation	38
http_parser/http_response.hpp	
HttpResponse class declaration	39
http_parser/http_session.cpp	
HttpSession class implementation	40
http parser/http session.hpp	
HttpSession class declaration. This class is thread-safe	40
http_parser/http_stream_reader.cpp	
HttpStreamReader class implementation	41
http parser/http stream reader.hpp	
HttpStreamReader class declaration	41
http_parser/url_encoded_form_parser.cpp	
UrlEncodedFormParser class implementation	41
http parser/url encoded form parser.hpp	
UrlEncodedFormParser class declaration	42
routing/router.hpp	
Router class definition and related consts/aliases	42

10 File Index

## **Chapter 6**

## **Class Documentation**

## 6.1 App Class Reference

#### **Public Member Functions**

- void add\_route (std::string url\_pattern, functor f)
  - add a <url, function> mapping to the Router instance
- void worker (int clntSock, string strRequest)

Runs in the work thread spawned to handle given httpRequest.

- void reload\_url\_mappings ()
- void add monitored dir (const std::string dir)

Add a directory to monitor for dynamic loading feature.

- void set\_urls\_json\_dir (std::string dir)
- std::string get\_urls\_json\_dir () const
- void run (int port)

starts Http server socket. Runs forever on the given port, setting up listening socket that polls for client connections. If there is a client connection, calls <a href="https://handle.request(">handle\_request()</a>) to handle the request in a spawned thread.

int handle\_request (int socket)

handles request for given socket, it calls recv() on the socket and spawns a new worker thread with worker() to handle the received request

## **Public Attributes**

Router router

## 6.1.1 Member Function Documentation

## 6.1.1.1 add\_monitored\_dir()

Add a directory to monitor for dynamic loading feature.

#### **Parameters**

```
std::string dir
```

#### Returns

void

#### 6.1.1.2 handle\_request()

handles request for given socket, it calls recv() on the socket and spawns a new worker thread with worker() to handle the received request

## **Parameters**

```
int clntSock
```

#### Returns

void

#### 6.1.1.3 run()

starts Http server socket. Runs forever on the given port, setting up listening socket that polls for client connections. If there is a client connection, calls handle\_request() to handle the request in a spawned thread.

#### **Parameters**

int port, port to start the server

## Returns

void

Bug if the number of opened sockets exceeds SOMAXCONN, Cjango gets "apr\_socket\_recv: Connection reset by peer (54)" error and halts immediately. On Mac OS X (default: 128), check by "sysctl -a | grep somax" and change it by "sudo sysctl -w kern.ipc.somaxconn=2048"

#### 6.1.1.4 worker()

Runs in the work thread spawned to handle given httpRequest.

#### **Parameters**

int	clntSock, socket to send the response
std::string	strRequest, request in string

## Returns

void

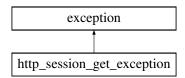
The documentation for this class was generated from the following files:

- · app/app.hpp
- · app/app.cpp

## 6.2 http\_session\_get\_exception Class Reference

custom exception class in case HttpSession::get wants to hard-fail when key is not present

Inheritance diagram for http\_session\_get\_exception:



## 6.2.1 Detailed Description

custom exception class in case HttpSession::get wants to hard-fail when key is not present

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

http\_parser/http\_session.cpp

## 6.3 http::HttpRequest Class Reference

#### **Public Member Functions**

- HttpRequest (std::string path)
- HttpRequest (std::string, std::string, std::string, std::unordered\_map< std::string, std::string >, std::unordered\_map< std::string, std::string >)

HttpRequest Constructor that represents one HTTP request that can be received by a server.

· std::string get method () const

getter for http request's method. Ex: GET

• std::string get\_path () const

getter for http request's path. Ex: /abc.html

• std::string get\_scheme () const

getter for http request's scheme. Ex: HTTP/1.0

• unsigned long get\_session\_id ()

getter for http request's session\_id, persisted via cookie mechanism

bool has\_session\_id ()

check if http request has a session\_id associated with it

• std::unordered map< std::string, std::string > const & get meta () const

getter for http request's meta variables in the request header.

• std::unordered\_map< std::string, std::string > const & get\_parameters () const

getter for http request's get/post parameters

• std::unordered\_map< std::string, std::string > const & get\_cookie () const

getter for http request's cookie map

std::shared\_ptr< HttpSession > get\_session ()

getter for a session object pointer associated with the current http request If there is no HttpSession object associated with the http request, a new HttpSession object will be created for the request.

#### Static Public Member Functions

• static unsigned long xorshf96 ()

Marsaglia's xorshf generator.

## **Public Attributes**

- · std::string method
- · std::string path
- std::string scheme

## **Static Public Attributes**

static std::string session\_cookie\_key ="session"

#### 6.3.1 Constructor & Destructor Documentation

#### 6.3.1.1 HttpRequest()

```
http::HttpRequest::HttpRequest (
    std::string method,
    std::string path,
    std::string scheme,
    std::unordered_map< std::string, std::string > meta,
    std::unordered_map< std::string, std::string > params,
    std::unordered_map< std::string, std::string > cookie )
```

HttpRequest Constructor that represents one HTTP request that can be received by a server.

#### **Parameters**

method	http request method
path	http request path
scheme	http request version
meta	a map containing the key value pairs of http headers
params	a map containing the key value pairs of http parameters
cookie	a map containing the key value pairs of http header's cookie entries

#### Returns

HttpRequest

## 6.3.2 Member Function Documentation

## 6.3.2.1 get\_cookie()

```
std::unordered_map< std::string, std::string > const & http::HttpRequest::get_cookie ( ) const
getter for http request's cookie map
```

## Returns

a map of key value pairs for http request cookie

#### 6.3.2.2 get\_meta()

```
std::unordered_map< std::string, std::string > const & http::HttpRequest::get_meta ( ) const
getter for http request's meta variables in the request header.
```

#### Returns

a map of key value pairs for http request header

```
6.3.2.3 get_method()
std::string http::HttpRequest::get_method ( ) const
getter for http request's method. Ex: GET
Returns
     a string indicating http request's method
6.3.2.4 get_parameters()
\verb|std::unordered_map| < \verb|std::string| > \verb|const| & | \verb|http::HttpRequest::get_parameters| ( ) \\
const
getter for http request's get/post parameters
Returns
     a map of key value pairs for http request parameters
6.3.2.5 get_path()
std::string http::HttpRequest::get_path ( ) const
getter for http request's path. Ex: /abc.html
Returns
     a path string
6.3.2.6 get_scheme()
std::string http::HttpRequest::get_scheme ( ) const
getter for http request's scheme. Ex: HTTP/1.0
Returns
     a string indicating the current http protocol version of the request
```

#### 6.3.2.7 get\_session()

```
std::shared_ptr< http::HttpSession > http::HttpRequest::get_session ( )
```

getter for a session object pointer associated with the current http request If there is no HttpSession object associated with the http request, a new HttpSession object will be created for the request.

#### Returns

a shared\_ptr of HttpSession object

#### 6.3.2.8 xorshf96()

```
unsigned long http::HttpRequest::xorshf96 ( ) [static]
```

Marsaglia's xorshf generator.

#### Returns

a pseudo random unsigned long

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

- http\_parser/http\_request.hpp
- http\_parser/http\_request.cpp

## 6.4 http::HttpRequestBodyParser Class Reference

#### **Public Member Functions**

- HttpRequestBodyParser ()
  - HttpRequestBodyParser constructor that contains a default UrlEncodedFormParser for form parsing.
- HttpRequestBodyParser (std::vector< UrlEncodedFormParser >)
- std::unordered\_map< std::string, std::string > parse (std::istream &, std::string, int)

given a http request body in istream containing http request parameters, it will parse the content of istream and loads the http request parameters to an unordered\_map

## **Public Attributes**

• std::vector< UrlEncodedFormParser > url\_encoded\_form\_parsers

## 6.4.1 Constructor & Destructor Documentation

#### 6.4.1.1 HttpRequestBodyParser()

```
http::HttpRequestBodyParser::HttpRequestBodyParser ( )
```

HttpRequestBodyParser constructor that contains a default UrlEncodedFormParser for form parsing.

Returns

HttpRequestBodyParser

#### 6.4.2 Member Function Documentation

```
6.4.2.1 parse()
```

given a http request body in istream containing http request parameters, it will parse the content of istream and loads the http request parameters to an unordered\_map

Returns

an unordered\_map containing the parameters for HttpRequest

## 6.4.3 Member Data Documentation

## 6.4.3.1 url\_encoded\_form\_parsers

```
\verb|std::vector<| UrlEncodedFormParser>| http::HttpRequestBodyParser::url\_encoded\_form\_parsers| | true | Leading | true |
```

parser that parses application/x-www-form-urlencoded

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

- http\_parser/http\_request\_body\_parser.hpp
- http\_parser/http\_request\_body\_parser.cpp

## 6.5 http::HttpRequestLine Class Reference

## **Public Member Functions**

• HttpRequestLine (std::string action, std::string uri, std::string protocolVersion, std::unordered\_map< std
::string, std::string > parameters)

HttpRequestLine constructor that represents the first line of a http request as well as the paramters of a get Http← Request.

• HttpRequestLine (std::string action, std::string uri, std::string protocolVersion)

HttpRequestLine constructor that represents the first line of a http request.

#### **Public Attributes**

- · std::string action
- · std::string uri
- std::unordered\_map< std::string, std::string > parameters
- std::string protocolVersion

## 6.5.1 Constructor & Destructor Documentation

## **6.5.1.1** HttpRequestLine() [1/2]

HttpRequestLine constructor that represents the first line of a http request as well as the paramters of a get Http↔ Request.

## Parameters

action	method of a http request
uri	path of a http request
parameters	a map containing the parameters of a http get request
protocolVersion	version of the http protocol for the request

#### **6.5.1.2** HttpRequestLine() [2/2]

```
std::string uri,
std::string protocolVersion )
```

HttpRequestLine constructor that represents the first line of a http request.

#### **Parameters**

action	method of a http request
uri	path of a http request
protocolVersion	version of the http protocol for the request

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

- http\_parser/http\_request\_line.hpp
- http\_parser/http\_request\_line.cpp

## 6.6 http::HttpRequestParser Class Reference

#### **Public Member Functions**

- HttpRequestParser (HttpRequestBodyParser)
- HttpRequest parse\_request\_line\_and\_headers (std::istream &input\_stream)

given an input\_stream containing a http request, parses the request line and headers

- std::unordered\_map< std::string, std::string > parse\_body (std::istream &, std::string, int)
   helper method that parses a http request's body
- HttpRequest parse (std::istream &input\_stream)

given an input\_stream containing a http request, parses the request line, headers, and body

## **Public Attributes**

- HttpRequestBodyParser body\_parser
- HttpStreamReader reader
- UrlEncodedFormParser url encoded form parser

#### 6.6.1 Member Function Documentation

#### 6.6.1.1 parse()

given an input\_stream containing a http request, parses the request line, headers, and body

#### **Parameters**

ing a http request string	input_stream	
---------------------------	--------------	--

## Returns

a http request object representing the data from the input\_stream

#### 6.6.1.2 parse\_body()

helper method that parses a http request's body

#### **Parameters**

input_stream	a stream containing http request's body
content_type	the content type of the http request as specified in its headers
content_leng	the content length of the http request as specified in its headers

#### Returns

a map containing key value pairs of the http request's cookie

#### 6.6.1.3 parse\_request\_line\_and\_headers()

given an input\_stream containing a http request, parses the request line and headers

#### **Parameters**

#### Returns

HttpRequest with populated http request headers and an empty body

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

- http\_parser/http\_request\_parser.hpp
- http\_parser/http\_request\_parser.cpp

## 6.7 http::HttpResponse Class Reference

#### **Public Member Functions**

HttpResponse (std::string)

HttpResponse constructor with a 200 status code.

HttpResponse (std::string, HttpRequest &)

HttpResponse constructor with a 200 status code and content type text/html.

• HttpResponse (std::string, std::string)

HttpResponse constructor with a 200 status code.

HttpResponse (int)

HttpResponse constructor with content type text/html.

• std::string to\_string ()

returns a well-formated string version of http response compliant with http/1.0 protocol

void set\_cookie (std::string, std::string)

given a key value pair, inserts the pair to request's cookie

#### **Static Public Member Functions**

static HttpResponse render\_to\_response (std::string)

given a file path, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.

static HttpResponse render\_to\_response (std::string, std::string)

given a file path and content type of the file, generate a http response The generated HttpResponse will have text/html as its content-type and a status code of 200.

• static HttpResponse render to response (std::string, HttpRequest &)

given a file path and a http request, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.

• static HttpResponse render\_to\_response (std::string, std::string, HttpRequest &)

given a file path, content type of the file, and a http request, create a http response

• static std::string get\_template (std::string path)

given a file path, generate a string containing file's data

#### **Public Attributes**

- · std::string content
- int **status\_code** = 200
- std::string reason\_phrase = "OK"
- std::string http\_version = "HTTP/1.0"
- std::string content\_type = "text/html"
- std::unordered\_map< std::string, std::string > headers

#### **Static Public Attributes**

- static std::unordered map< int, std::string > code to reason
- · static std::string templates\_root

## 6.7.1 Constructor & Destructor Documentation

HttpResponse constructor with a 200 status code.

#### **Parameters**

content body of the http response
-----------------------------------

## Returns

HttpResponse

## **6.7.1.2** HttpResponse() [2/4]

HttpResponse constructor with a 200 status code and content type text/html.

## **Parameters**

content	body of the http response
request	a HttpRequest object which corresponds to the current http response

#### Returns

HttpResponse

## **6.7.1.3** HttpResponse() [3/4]

HttpResponse constructor with a 200 status code.

#### **Parameters**

content	body of the http response
content_type	content type of the http response

#### Returns

HttpResponse

## **6.7.1.4** HttpResponse() [4/4]

```
\label{eq:http:httpResponse:HttpResponse} \mbox{ (} \\ \mbox{int } status\_code \mbox{ )}
```

HttpResponse constructor with content type text/html.

## **Parameters**

code of the http response	status_code
---------------------------	-------------

#### Returns

HttpResponse

## 6.7.2 Member Function Documentation

## 6.7.2.1 get\_template()

given a file path, generate a string containing file's data

## **Parameters**

path	path to a file whose content will be returned as a string path should be relative to
	HttpResponse::templates_root

#### Returns

a string representing the file's content

given a file path, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.

#### **Parameters**

path	path to a file whose content will be in the http response's body. path should be relative to
	HttpResponse::templates_root

#### Returns

HttpResponse

std::string content\_type ) [static]

given a file path and content type of the file, generate a http response The generated HttpResponse will have text/html as its content-type and a status code of 200.

#### **Parameters**

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root
content_type	content type of the http response whose value is set in http response's header

#### Returns

HttpResponse

given a file path and a http request, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.

## **Parameters**

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root
request	a HttpRequest object which corresponds to the current http response

#### Returns

HttpResponse

```
6.7.2.5 render_to_response() [4/4]
```

given a file path, content type of the file, and a http request, create a http response

#### **Parameters**

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root
content_type	content type of the http response whose value is set in http response's header
request	a HttpRequest object which corresponds to the current http response

## Returns

HttpResponse

## 6.7.2.6 set\_cookie()

given a key value pair, inserts the pair to request's cookie

#### **Parameters**

key	cookie key
value	cookie value

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

- http\_parser/http\_response.hpp
- http\_parser/http\_response.cpp

## 6.8 http::HttpSession Class Reference

#### **Public Member Functions**

• HttpSession ()

HttpSession constructor that initializes a reader-writer lock.

• std::string get (std::string)

given a key, return its value from the session map. This method is thread safe

• void set (std::string, std::string)

given a (key, value pair), insert it to the session map. This method is thread safe

#### 6.8.1 Member Function Documentation

```
6.8.1.1 get()
```

given a key, return its value from the session map. This method is thread safe

#### **Parameters**

key key used to retrieve value stored in the session map

#### Returns

if key is found, the value corresponding to the key. If key is not found, an empty string

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

- http\_parser/http\_session.hpp
- http\_parser/http\_session.cpp

## 6.9 http::HttpStreamReader Class Reference

#### **Public Member Functions**

std::string get\_next\_line (std::istream &input\_stream)

given an input stream, returns the current line deliminted by '

• std::string to\_string (std::istream &input\_stream)

given an input stream, converts it to string

- std::string read\_util (std::istream &input\_stream, int character)
- std::string read (std::istream &input\_stream, int length)

given an input stream and number of length bytes we want to read from the input stream, converts number of length bytes + 1 from input stream to a string

void eat\_white\_space (std::istream &input\_stream)

removes white spaces from input stream

## **Static Public Attributes**

- static const int carraige\_return = '\r'
- static const int line\_feed = '\n'

#### 6.9.1 Member Function Documentation

#### 6.9.1.1 eat\_white\_space()

removes white spaces from input stream

#### **Parameters**

## 6.9.1.2 get\_next\_line()

given an input stream, returns the current line deliminted by '

#### **Parameters**

input_stream	is the input stream we are currently processing
--------------	---

#### Returns

a string representing the next line in the input\_stream

#### 6.9.1.3 read()

given an input stream and number of length bytes we want to read from the input stream, converts number of length bytes + 1 from input stream to a string

#### **Parameters**

input_stream	is the input stream we are currently processing
length	number of bytes we would like to read from the input stream

## 6.9.1.4 to\_string()

given an input stream, converts it to string

#### **Parameters**

input_stream	is the input stream we are currently processing
--------------	---

## Returns

the entire input stream as a string

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

- http\_parser/http\_stream\_reader.hpp
- http\_parser/http\_stream\_reader.cpp

## 6.10 MSocket Class Reference

**Public Member Functions** 

• MSocket (int sock)

- · int socket ()
- · void close ()
- int set\_reusable ()
- int set\_nonblocking ()
- int **bind** (int port)
- int listen (int maxconn)
- std::shared\_ptr< MSocket > accept ()
- int recv (void \*buf, size t len)
- int send (void \*buf, size t len)
- int **send** (std::string str)

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

- · app/msocket.hpp
- app/msocket.cpp

## 6.11 Router Class Reference

#### **Public Member Functions**

- Router (URLmap routes)
- int nr\_patterns () const
- void erase\_all\_patterns ()
- void add\_route (std::string url\_pattern, functor f)

Add a mapping from url\_pattern to a callback into patterns\_list and pattern\_to\_callback. if the given url pattern is already set, this will overwrite the pattern by the new callback.

- void set\_static\_dir (const std::string dir)
- std::string get\_static\_dir () const
- std::string resolve (http::HttpRequest) const

Called in get\_http\_response() for mapping a given request's path to an url pattern. If no pattern matches the given request's path, RouterException::INVALID\_URL will be returned. If starting from "/static/", RouterException::STAT← IC\_FILE\_SERVED will be returned. Both enums are handled in get\_http\_response().

http::HttpResponse get\_http\_response (http::HttpRequest)

find and execute a callback corresponding to the given request

#### 6.11.1 Member Function Documentation

#### 6.11.1.1 add\_route()

Add a mapping from url\_pattern to a callback into patterns\_list and pattern\_to\_callback. if the given url pattern is already set, this will overwrite the pattern by the new callback.

#### **Parameters**

```
url_pattern (e.g. "/diary/[0-9]{4}")
```

#### Returns

void

### 6.11.1.2 get\_http\_response()

```
http::HttpResponse Router::get_http_response (
    http::HttpRequest request )
```

find and execute a callback corresponding to the given request

#### Returns

an HttpResponse object built by the callback

### 6.11.1.3 resolve()

Called in get\_http\_response() for mapping a given request's path to an url pattern. If no pattern matches the given request's path, RouterException::INVALID\_URL will be returned. If starting from "/static/", RouterException::STA TIC\_FILE\_SERVED will be returned. Both enums are handled in get\_http\_response().

#### Returns

a corresponding url pattern such as "diary/[0-9]{4}/[0-9]{2}/"

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

- routing/router.hpp
- · routing/router.cpp

### 6.12 Selector Struct Reference

#### **Public Member Functions**

- · Selector (int sock)
- void set\_server (int sock)
- void fd\_zero ()
- void fd\_set2 (int sock)
- bool fd\_isset (int sock, int m=0)
- void fd\_clr (int sock)
- bool server\_isset (int m=0)
- int select ()
- void report (int m=0)

32 Class Documentation

#### **Public Attributes**

- int servSock = -1
- fd set allSocks
- · fd set retSocks
- · struct timeval timeout
- int maxSock = -1

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

· app/selector.hpp

### 6.13 http::UrlEncodedFormParser Class Reference

#### **Public Member Functions**

- bool can\_parse\_content\_type (std::string content\_type)
   checks if cotent\_type in question can be parsed by the current parser
- · char charToInt (char)
- char strToBin (char \*pString)
- std::string urlDecode (const std::string &str)
- std::unordered\_map< std::string, std::string > get\_parameter (std::istream &input\_stream, int content\_leng) given current input stream which contains get or post params encoded as a single string, parses the stream into a map
- std::vector < std::string > split (std::string str, char delimiter)
   splits a string by a character delimiter and returns the result as a vector

#### **Public Attributes**

- HttpStreamReader input\_stream\_reader
- std::string supported\_content\_type = "application/x-www-form-urlencoded"

#### 6.13.1 Member Function Documentation

#### 6.13.1.1 can\_parse\_content\_type()

checks if cotent\_type in question can be parsed by the current parser

#### **Parameters**

content type	a string representing the content type of a http request

#### Returns

: a boolean indicating whether content\_type is supported

### 6.13.1.2 get\_parameter()

given current input stream which contains get or post params encoded as a single string, parses the stream into a map

#### **Parameters**

input_stream	is the input stream we are currently processing which contains get or post params
content_leng	length of the input_stream

#### Returns

a map containing http request's parameters

### 6.13.1.3 split()

```
std::vector< std::string > http::UrlEncodedFormParser::split ( std::string str, char delimiter)
```

splits a string by a character delimiter and returns the result as a vector

### Parameters

str	string to be split
delimiter	a character delimiter to split the string by

### Returns

the split string in a vector

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

- http\_parser/url\_encoded\_form\_parser.hpp
- http\_parser/url\_encoded\_form\_parser.cpp

34 Class Documentation

## **Chapter 7**

## **File Documentation**

### 7.1 http\_parser/http\_request.cpp File Reference

HttpRequest class implementation.

```
#include "http_request.hpp"
#include <stdio.h>
#include <stdlib.h>
#include "../app/externs.hpp"
```

### **Variables**

• std::string http\_logger\_name = "http\_request"

### 7.1.1 Detailed Description

HttpRequest class implementation.

### 7.2 http\_parser/http\_request.hpp File Reference

HttpRequest class declaration.

```
#include <unordered_map>
#include <iostream>
#include <fstream>
#include <string>
#include <memory>
#include "http_session.hpp"
#include <pthread.h>
```

### Classes

· class http::HttpRequest

### **Functions**

std::ostream & http::operator<< (std::ostream &Str, HttpRequest const &v)</li>
 overloading << for HttpRequest object that enables HttpRequest to be written to a output stream</li>

### 7.2.1 Detailed Description

HttpRequest class declaration.

#### 7.2.2 Function Documentation

### 7.2.2.1 operator << ()

```
std::ostream & http::operator<< (  std::ostream \& Str, \\  HttpRequest const \& v )
```

overloading << for HttpRequest object that enables HttpRequest to be written to a output stream

### Returns

std::ostream& enabling HttpRequest to be written to a output stream

### 7.3 http\_parser/http\_request\_body\_parser.cpp File Reference

HttpRequestBodyParser class implementation.

```
#include "http_request_body_parser.hpp"
#include "../app/externs.hpp"
```

### 7.3.1 Detailed Description

HttpRequestBodyParser class implementation.

### 7.4 http\_parser/http\_request\_body\_parser.hpp File Reference

HttpRequestBodyParser class declaration.

```
#include <vector>
#include <unordered_map>
#include <iostream>
#include <fstream>
#include "url encoded form parser.hpp"
```

#### **Classes**

• class http::HttpRequestBodyParser

### 7.4.1 Detailed Description

HttpRequestBodyParser class declaration.

### 7.5 http\_parser/http\_request\_line.cpp File Reference

HttpRequestLine class implementation.

```
#include "http_request_line.hpp"
```

### 7.5.1 Detailed Description

HttpRequestLine class implementation.

### 7.6 http\_parser/http\_request\_line.hpp File Reference

HttpRequestLine class declaration.

```
#include <unordered_map>
#include <string>
```

### Classes

· class http::HttpRequestLine

### 7.6.1 Detailed Description

HttpRequestLine class declaration.

### 7.7 http\_parser/http\_request\_parser.cpp File Reference

HttpRequestParser class implementation.

```
#include "http_request_parser.hpp"
#include "../app/externs.hpp"
#include <algorithm>
#include <string>
#include <iostream>
```

### 7.7.1 Detailed Description

HttpRequestParser class implementation.

### 7.8 http\_parser/http\_request\_parser.hpp File Reference

HttpRequestParser class declaration.

```
#include "url_encoded_form_parser.hpp"
#include "http_request_body_parser.hpp"
#include "http_request.hpp"
#include "http_request_line.hpp"
#include <unordered_map>
#include <iostream>
#include <fstream>
```

### Classes

· class http::HttpRequestParser

### 7.8.1 Detailed Description

HttpRequestParser class declaration.

### 7.9 http\_parser/http\_response.cpp File Reference

HttpResponse class implementation.

```
#include "http_response.hpp"
#include <fstream>
#include <streambuf>
#include "../app/externs.hpp"
```

### **Functions**

std::string get\_reason\_phrase (int status\_code)
 helper method that translates http status code to status message

### 7.9.1 Detailed Description

HttpResponse class implementation.

#### 7.9.2 Function Documentation

### 7.9.2.1 get\_reason\_phrase()

```
std::string get_reason_phrase (
    int status_code )
```

helper method that translates http status code to status message

#### **Parameters**

### Returns

the corresponding status message

### 7.10 http\_parser/http\_response.hpp File Reference

HttpResponse class declaration.

```
#include <unordered_map>
#include <string>
#include "http_request.hpp"
```

### Classes

· class http::HttpResponse

### **Functions**

std::ostream & http::operator<< (std::ostream &Str, HttpResponse const &v)</li>
 overloads << function for debugging purposes: convenience of printing out http response</li>

### 7.10.1 Detailed Description

HttpResponse class declaration.

### 7.11 http\_parser/http\_session.cpp File Reference

HttpSession class implementation.

```
#include "http_session.hpp"
```

#### **Classes**

class http\_session\_get\_exception
 custom exception class in case HttpSession::get wants to hard-fail when key is not present

### **Variables**

• http\_session\_get\_exception sessionex

### 7.11.1 Detailed Description

HttpSession class implementation.

### 7.12 http\_parser/http\_session.hpp File Reference

HttpSession class declaration. This class is thread-safe.

```
#include <unordered_map>
#include <iostream>
#include <fstream>
#include <string>
#include <mutex>
#include <exception>
#include <pthread.h>
```

#### **Classes**

• class http::HttpSession

### 7.12.1 Detailed Description

HttpSession class declaration. This class is thread-safe.

### 7.13 http\_parser/http\_stream\_reader.cpp File Reference

HttpStreamReader class implementation.

```
#include "http_stream_reader.hpp"
#include "../app/externs.hpp"
```

### 7.13.1 Detailed Description

HttpStreamReader class implementation.

### 7.14 http\_parser/http\_stream\_reader.hpp File Reference

HttpStreamReader class declaration.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
```

#### Classes

• class http::HttpStreamReader

### 7.14.1 Detailed Description

HttpStreamReader class declaration.

### 7.15 http\_parser/url\_encoded\_form\_parser.cpp File Reference

UrlEncodedFormParser class implementation.

```
#include "url_encoded_form_parser.hpp"
#include "../app/externs.hpp"
#include <ctype.h>
#include <stdlib.h>
```

### 7.15.1 Detailed Description

UrlEncodedFormParser class implementation.

### 7.16 http\_parser/url\_encoded\_form\_parser.hpp File Reference

UrlEncodedFormParser class declaration.

```
#include "http_stream_reader.hpp"
#include <string>
#include <unordered_map>
#include <iostream>
#include <fstream>
#include <exception>
#include <vector>
#include <sstream>
```

#### Classes

· class http::UrlEncodedFormParser

### 7.16.1 Detailed Description

UrlEncodedFormParser class declaration.

### 7.17 routing/router.hpp File Reference

Router class definition and related consts/aliases.

```
#include <stdio.h>
#include <functional>
#include <unordered_map>
#include <vector>
#include <string>
#include <stdexcept>
#include <algorithm>
#include <cjango>
```

#### Classes

class Router

### **Typedefs**

- using **functor** = std::function< http::HttpResponse(http::HttpRequest)>
- using URLmap = std::unordered\_map< std::string, functor >

#### **Enumerations**

enum RouterException { INVALID\_URL, STATIC\_FILE\_SERVED }

### **Variables**

• const std::string route\_logger\_name = "route"

### 7.17.1 Detailed Description

Router class definition and related consts/aliases.

Author

caprice-j

Date

2017.04.17

Version

0.1

### Warning

this example may include some wrong descriptions

# Index

add_monitored_dir	get_path, 16
App, 11	get_scheme, 16
add_route	get_session, 16
Router, 30	HttpRequest, 14
App, 11	xorshf96, 17
add_monitored_dir, 11	http::HttpRequestBodyParser, 17
handle_request, 12	HttpRequestBodyParser, 17
run, 12	parse, 18
worker, 12	url_encoded_form_parsers, 18
	http::HttpRequestLine, 19
can_parse_content_type	HttpRequestLine, 19
http::UrlEncodedFormParser, 32	http::HttpRequestParser, 20
	parse, 20
eat_white_space	parse_body, 21
http::HttpStreamReader, 28	parse_request_line_and_headers, 21
	http::HttpResponse, 22
get	get_template, 24
http::HttpSession, 27	HttpResponse, 23, 24
get_cookie	render_to_response, 24–26
http::HttpRequest, 15	— — ·
get_http_response	set_cookie, 26
Router, 31	http::HttpSession, 27
get_meta	get, 27
http::HttpRequest, 15	http::HttpStreamReader, 27
get_method	eat_white_space, 28
http::HttpRequest, 15	get_next_line, 28
get_next_line	read, 29
http::HttpStreamReader, 28	to_string, 29
get_parameter	http::UrlEncodedFormParser, 32
http::UrlEncodedFormParser, 33	can_parse_content_type, 32
get_parameters	get_parameter, 33
http::HttpRequest, 16	split, 33
get_path	http_parser/http_request.cpp, 35
http::HttpRequest, 16	http_parser/http_request.hpp, 35
get_reason_phrase	http_parser/http_request_body_parser.cpp, 36
http_response.cpp, 39	http_parser/http_request_body_parser.hpp, 37
get_scheme	http_parser/http_request_line.cpp, 37
http::HttpRequest, 16	http_parser/http_request_line.hpp, 37
get_session	http_parser/http_request_parser.cpp, 38
http::HttpRequest, 16	http_parser/http_request_parser.hpp, 38
get_template	http_parser/http_response.cpp, 38
http::HttpResponse, 24	http_parser/http_response.hpp, 39
Titipiii titipi tooponoo, 2 T	http_parser/http_session.cpp, 40
handle_request	http_parser/http_session.hpp, 40
App, 12	http_parser/http_stream_reader.cpp, 41
http::HttpRequest, 14	http parser/http stream reader.hpp, 41
get_cookie, 15	http_parser/url_encoded_form_parser.cpp, 41
get_meta, 15	http_parser/url_encoded_form_parser.hpp, 42
get_method, 15	http_request.hpp
get_parameters, 16	operator<<, 36
got_paramotors, 10	operator < <, 00

46 INDEX

```
http_response.cpp
     get_reason_phrase, 39
http_session_get_exception, 13
HttpRequest
    http::HttpRequest, 14
HttpRequestBodyParser
    http::HttpRequestBodyParser, 17
HttpRequestLine
     http::HttpRequestLine, 19
HttpResponse
    http::HttpResponse, 23, 24
MSocket, 29
operator<<
    http_request.hpp, 36
parse
     http::HttpRequestBodyParser, 18
    http::HttpRequestParser, 20
parse_body
    http::HttpRequestParser, 21
parse_request_line_and_headers
    http::HttpRequestParser, 21
read
    http::HttpStreamReader, 29
render to response
    http::HttpResponse, 24-26
resolve
     Router, 31
Router, 30
    add_route, 30
    get_http_response, 31
     resolve, 31
routing/router.hpp, 42
run
     App, 12
Selector, 31
set_cookie
    http::HttpResponse, 26
split
    http::UrlEncodedFormParser, 33
to_string
    http::HttpStreamReader, 29
url_encoded_form_parsers
    http::HttpRequestBodyParser, 18
worker
     App, 12
xorshf96
    http::HttpRequest, 17
```