trustyRC

0.1.4

Generated by Doxygen 1.5.8

Sun Aug 16 15:28:26 2009

Contents

Chapter 1

Directory Hierarchy

1.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Directory Documentation

5.1 src/plugins/ Directory Reference

Files

- file admin.cpp

  Admin implementation file.

- file admin.h

  Admin header file.

- file advertising.cpp

  Advertising implementation file.

- file advertising.h

  Advertising header file.

- file antiflood.cpp

  AntiFlood implementation file.

- file antiflood.h

  AntiFlood header file.

- file bzrh.cpp

  BZRH implementation file.

- file bzrh.h

  BZRH header file.

- file ctcp.cpp

  CTCP implementation file.

- file ctcp.h

  CTCP header file.

- file danstonchat.cpp
  DansTonChat implementation file.
- file danstonchat.h
  DansTonChat header file.
- file fedorafr.cpp
  Fedorafr implementation file.
- file fedorafr.h
  Fedorafr header file.
- file fedoraproject.cpp
  FedoraProject implementation file.
- file fedoraproject.h
  FedoraProject header file.
- file gameserver.cpp
  GameServer implementation file.
- file gameserver.h
  GameServer header file.
- file ignore.cpp
  Ignore implementation file.
- file ignore.h
  Ignore header file.
- file infos.cpp
  Infos implementation file.
- file infos.h
  Infos header file.
- file ipconverting.cpp
  IpConverting implementation file.
- file ipconverting.h
  IpConverting header file.
- file lamoule.cpp
  Lamoule implementation file.
- file lamoule.h
  Lamoule header file.
- file logfactory.cpp
  LogFactory implementation file.

- file logfactory.h

  LogFactory header file.

- file magic8ball.cpp

  Magic8Ball implementation file.

- file magic8ball.h

  Magic8Ball header file.

- file moderation.cpp

  Moderation implementation file.

- file moderation.h

  Moderation header file.

- file module.cpp

  Module implementation file.

- file module.h

  Module header file.

- file ping.cpp

  Ping implementation file.

- file ping.h

  Ping header file.

- file pluginsample.cpp

  PluginSample implementation file.

- file pluginsample.h

  PluginSample header file.

- file postconnect.cpp

  PostConnect implementation file.

- file postconnect.h

  PostConnect header file.

- file quotes.cpp

  Quotes implementation file.

- file quotes.h

  Quotes header file.

- file remotecontrol.cpp

  RemoteControl implementation file.

- file remotecontrol.h

RemoteControl header file.

- file slapme.cpp

  Slapme implementation file.

- file slapme.h

  Slapme header file.

- file survey.cpp

  Survey implementation file.

- file survey.h

  Survey header file.

- file tele.cpp

  Tele implementation file.

- file tele.h

  Tele header file.

- file trad.cpp

  Trad implementation file.

- file trad.h

  Trad header file.

- file usersinfos.cpp

  UsersInfos implementation file.

- file usersinfos.h

  UsersInfos header file.

5.2 src/ Directory Reference

Directories

- directory plugins

Files

- file botkernel.cpp

  BotKernel implementation file.

- file botkernel.h

  BotKernel header file.

- file channel.cpp

  Channel implementation file.

- file channel.h

  Channel header file.

- file configurationfile.cpp

  ConfigurationFile implementation file.

- file configurationfile.h

  ConfigurationFile header file.

- file cppthread.cpp

  CPPThread implementation file.

- file cppthread.h

  CPPThread header file.

- file ircprotocol.cpp

  IRCProtocol implementation file.

- file ircprotocol.h

  IRCProtocol header file.

- file logfile.cpp

  LogFile implementation file.

- file logfile.h

  LogFile header file.

- file main.cpp

  Main program.

- file message.cpp

  Message implementation file.

- file message.h

  Message header file.

- file plugin.cpp

  Plugin implementation file.

- file plugin.h

  Plugin header file.

- file socket.cpp

  Socket implementation file.

- file socket.h

  Socket header file.

- file tools.cpp

  Tools implementation file.

- file tools.h

  Tools header file.

Chapter 6

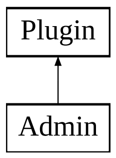
Class Documentation

6.1 Admin Class Reference

Bot access management.

`#include <admin.h>`

Inheritance diagram for Admin::



Public Member Functions

- Admin (BotKernel ∗)

  Constructor.

- bool addChannel (string)

  Add a channel managed by the bot.

- bool delChannel (string)

  Remove a channel managed by the bot.

- bool channelExists (string)

  Tell if a channel is registred.

- vector< string > getChannelsList ()

  Return channel list.

- bool addSuperAdmin (string)

  Add a super admin.

- bool addTempSuperAdmin (string, unsigned int)

Add a temporary super admin.

- void clearTempAdmins ()

  Clear timed out temporary super admins.
- bool delSuperAdmin (unsigned int)

  Del a super admin.
- bool isSuperAdmin (string)

  Tell if a user is a super admin.
- bool maskIsSuperAdmin (string)

  Tell if a mask is super admin.
- vector< string > superAdminList ()

  Give the super admin hosts list.
- bool addUser (string, string, unsigned int)

  Add a user to a channel.
- bool delUser (string, string)

  Del a user from a channel.
- bool userExists (string, string)

  Tell if a user exists for a channel.
- bool updateUserLevel (string, string, unsigned int)

  Update a user level.
- unsigned int getUserLevel (string, string)

  Get a user level for a channel.
- unsigned int getMaskLevel (string, string)

  Get mask level for a channel.
- vector< string > chanLevels (string)

  Give all the access for a channel.
- void enableCommand (string, string)

  Enable a command on a channel.
- void disableCommand (string, string)

  Disable a command on a channel.
- void addOnlyonCommand (string, string)

  Allow a command on a specific channel (disable for others).
- void delOnlyonCommand (string, string)

  Remove a Onlyon command.

- bool commandOK (string, string)

  Tell if a command is can be executed on a channel.

- vector< string > commandsStatus ()

  Give commands status (onlyon or disabled).

Private Member Functions

- void initFile ()

  Initialize the XML file.

Private Attributes

- TiXmlDocument ∗ doc

  Represent the xml document.

- TiXmlNode ∗ root

  Represent documents's root.

6.1.1 Detailed Description

Bot access management.

This plugin stores (with an xml file) bot accounts (by host) and provides stuff for administration

Definition at line 50 of file admin.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Admin::Admin (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file admin.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, doc, BotKernel::getDatasDir(), IN_BEFORE_TREATMENT, IN_COMMAND_HANDLER, IN_FIRST_WORD, IN_LOOP, IN_TYPE_-HANDLER, initFile(), Plugin::name, root, and Plugin::version.

6.1.3 Member Function Documentation

6.1.3.1 bool Admin::addChannel (string *channel*)

Add a channel managed by the bot.

Add a channel managed by the bot

Parameters:

channel Channel name that you want to add

Returns:

true if the channel has been added, else false

Definition at line 111 of file admin.cpp.

References channelExists(), doc, root, and Tools::to_lower().

Referenced by addUser().

6.1.3.2 void Admin::addOnlyonCommand (string *command*, string *channel*)

Allow a command on a specific channel (disable for others).

Allow a command on a specific channel (disable for others)

Parameters:

command Onlyon command

channel Channel where to enable only the command

Returns:

true if operation OK, else false

Definition at line 625 of file admin.cpp.

References doc, root, and Tools::to_lower().

Referenced by addOnlyon().

6.1.3.3 bool Admin::addSuperAdmin (string *mask*)

Add a super admin.

Add a super admin for the bot

Parameters:

mask Super admin's mask

Returns:

True if the admin has been added, else false

Definition at line 171 of file admin.cpp.

References doc, isSuperAdmin(), root, and Tools::to_lower().

Referenced by addsuperadmin().

6.1.3.4 bool Admin::addTempSuperAdmin (string *mask*, unsigned int *duration*)

Add a temporary super admin.

Add a temporary super admin for the bot

Parameters:

mask Super admin's mask

duration Super admin time (in seconds)

Returns:

True if the admin has been added, else false

Definition at line 192 of file admin.cpp.

References doc, isSuperAdmin(), root, and Tools::to_lower().

Referenced by addtempsuperadmin().

6.1.3.5 bool Admin::addUser (string *channel*, string *mask*, unsigned int *level*)

Add a user to a channel.

Add a user to a channel

Parameters:

channel Channel where to add the user

mask User's mask

level User's level

Returns:

True if the user has benn added, else false

Definition at line 333 of file admin.cpp.

References addChannel(), channelExists(), doc, root, Tools::to_lower(), and userExists().

Referenced by updateUserLevel().

6.1.3.6 vector< string > Admin::chanLevels (string *channel*)

Give all the access for a channel.

Give all the access of a channel

Parameters:

channel Channel for the one you want the access list

Returns:

A vector containing access for the given channel

Definition at line 576 of file admin.cpp.

References root, and Tools::to_lower().

Referenced by chanlev().

6.1.3.7 bool Admin::channelExists (string *channel*)

Tell if a channel is registred.

Tell if a channel is registred

Parameters:

channel Channel name that you want to test existance

Returns:

true if the channel exists, else false

Definition at line 153 of file admin.cpp.

References root, and Tools::to_lower().

Referenced by addChannel(), and addUser().

6.1.3.8 void Admin::clearTempAdmins ()

Clear timed out temporary super admins.

Return Clear outdated temporary super admins

Definition at line 308 of file admin.cpp.

References doc, and Tools::strToInt().

6.1.3.9 bool Admin::commandOK (string *command*, string *channel*)

Tell if a command is can be executed on a channel.

Tell if a command can be executed on a channel Check if the command is disabled or is not a "only on" one.

Parameters:

command Command to check

channel Channel to check

Returns:

True if the command is ok, else false

Definition at line 695 of file admin.cpp.

References Tools::isInVector(), root, and Tools::to_lower().

Referenced by allowedCommandCheck().

6.1.3.10 vector< string > Admin::commandsStatus ()

Give commands status (onlyon or disabled).

Give commands status (onlyon or disabled)

Returns:

A vector containing commands status

Definition at line 725 of file admin.cpp.

References root.

Referenced by commandsStatus().

6.1.3.11 bool Admin::delChannel (string *channel*)

Remove a channel managed by the bot.

Remove a channel managed by the bot

Parameters:

channel Channel name that you want to delete

Returns:

true if the channel has been deleted, else false

Definition at line 130 of file admin.cpp.

References doc, root, and Tools::to_lower().

Referenced by delUser().

6.1.3.12 void Admin::delOnlyonCommand (string *command*, string *channel*)

Remove a Onlyon command.

Remove a Onlyon command

Parameters:

command Command on unonlyon
channel Channel where the command is no more onlyon

Definition at line 639 of file admin.cpp.

References doc, root, and Tools::to_lower().

Referenced by delOnlyon().

6.1.3.13 bool Admin::delSuperAdmin (unsigned int *index*)

Del a super admin.

Del a super admin

Parameters:

index Super admin index

Returns:

True is the users has been deleted, else false

Definition at line 215 of file admin.cpp.

References doc.

Referenced by delsuperadmin().

6.1.3.14 bool Admin::delUser (string *channel*, string *mask*)

Del a user from a channel.

Del a user from a channel

Parameters:

> ***channel*** Channel where to del the user
> ***mask*** User's mask to delete

Returns:

> True if the user has been added, else false

Definition at line 405 of file admin.cpp.

References delChannel(), doc, root, and Tools::to_lower().

Referenced by updateUserLevel().

6.1.3.15 void Admin::disableCommand (string *command*, string *channel*)

Disable a command on a channel.

Disable a command on a channel

Parameters:

> ***command*** Command to disable
> ***channel*** Channel where to disable the command

Definition at line 679 of file admin.cpp.

References doc, root, and Tools::to_lower().

Referenced by disable().

6.1.3.16 void Admin::enableCommand (string *command*, string *channel*)

Enable a command on a channel.

Enable a command on a channel

Parameters:

> ***command*** Command to enable
> ***channel*** Channel where to enable the command

Definition at line 659 of file admin.cpp.

References doc, root, and Tools::to_lower().

Referenced by enable().

6.1.3.17 vector< string > Admin::getChannelsList ()

Return channel list.

Return channel list

Returns:

A vector containing

Definition at line 605 of file admin.cpp.

References root.

Referenced by whoami().

6.1.3.18 unsigned int Admin::getMaskLevel (string *channel*, string *mask*)

Get mask level for a channel.

Get mask level for a channel 0 for nothing 1 for voice 2 for op 3 for master 4 for owner

Parameters:

channel User's channel

mask User's mask

Returns:

User's level (0 if non existing)

Definition at line 536 of file admin.cpp.

References Tools::masksMatch(), root, Tools::strToInt(), and Tools::to_lower().

Referenced by modeHandler().

6.1.3.19 unsigned int Admin::getUserLevel (string *channel*, string *mask*)

Get a user level for a channel.

Get a user level for a channel 0 for nothing 1 for voice 2 for op 3 for master 4 for owner

Parameters:

channel User's channel

mask User's mask

Returns:

User's level (0 if non existing)

Definition at line 499 of file admin.cpp.

References Tools::ircMaskMatch(), root, Tools::strToInt(), and Tools::to_lower().

Referenced by chanlev(), Moderation::checkAccess(), Moderation::hasOpPrivileges(), invite(), joinHandler(), kickHandler(), modeHandler(), and whoami().

6.1.3.20 void Admin::initFile () `[private]`

Initialize the XML file.

Initilaize the XML file by creating root and first childs (file empty structure)

Definition at line 90 of file admin.cpp.

References doc, and root.

Referenced by Admin().

6.1.3.21 bool Admin::isSuperAdmin (string *mask*)

Tell if a user is a super admin.

Tell if a user is a super admin

Parameters:

mask mask for witch you want to check the access

Returns:

true if the user is a super admin, else false

Definition at line 235 of file admin.cpp.

References Tools::ircMaskMatch(), root, and Tools::to_lower().

Referenced by addad(), addIgnore(), addOnlyon(), addSuperAdmin(), addTempSuperAdmin(), adinfos(), chanlev(), clearCountDowns(), commandsStatus(), cycleChannel(), delad(), deletekey(), deleteplayer(), delIgnore(), delOnlyon(), delQuote(), disable(), disconnect(), enable(), flushconffile(), getconfvalue(), getnbcountdowns(), Moderation::hasOpPrivileges(), ignoreList(), increase(), invite(), isIgnored(), joinChannel(), kickHandler(), leaveChannel(), listads(), listlibs(), listmodules(), load(), loadconffile(), loadnocheck(), modeHandler(), modeHandlerProtect(), moduleinfos(), nextscore(), notice(), onInvite(), protectmodes(), protecttopic(), quoteInfos(), raw(), reauth(), reloadfas(), reset(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), stopSurvey(), superadminlist(), tell(), testMsgTimestamp(), topicHandler(), unload(), unloadnocheck(), unprotectmodes(), unprotecttopic(), and whoami().

6.1.3.22 bool Admin::maskIsSuperAdmin (string *mask*)

Tell if a mask is super admin.

Tell is a mask is a super admin

Parameters:

mask Mask to test

Returns:

true is it's a superadmin else false

Definition at line 253 of file admin.cpp.

References Tools::masksMatch(), root, and Tools::to_lower().

Referenced by modeHandler().

6.1.3.23 vector< string > Admin::superAdminList ()

Give the super admin hosts list.

Return the list of the super admins

Returns:

a vector of string containing super admins' hosts

Definition at line 280 of file admin.cpp.

References doc, Tools::intToStr(), and Tools::strToInt().

Referenced by superadminlist().

6.1.3.24 bool Admin::updateUserLevel (string *channel*, string *mask*, unsigned int *level*)

Update a user level.

Update a User's level

Parameters:

channel user's channel

mask user's mask

level New user's level

Definition at line 444 of file admin.cpp.

References addUser(), delUser(), doc, root, Tools::to_lower(), and userExists().

Referenced by chanlev().

6.1.3.25 bool Admin::userExists (string *channel*, string *mask*)

Tell if a user exists for a channel.

Tell if a user exists for a channel The user is recognized by a host

Parameters:

channel Channel where we want to make the test

mask User's mask

Returns:

True if the user exists for the given channel, else false

Definition at line 373 of file admin.cpp.

References root, and Tools::to_lower().

Referenced by addUser(), and updateUserLevel().

6.1.4 Member Data Documentation

6.1.4.1 TiXmlDocument∗ Admin::doc `[private]`

Represent the xml document.

Definition at line 54 of file admin.h.

Referenced by addChannel(), addOnlyonCommand(), addSuperAdmin(), addTempSuperAdmin(), addUser(), Admin(), clearTempAdmins(), delChannel(), delOnlyonCommand(), delSuperAdmin(), delUser(), disableCommand(), enableCommand(), initFile(), superAdminList(), and updateUserLevel().

6.1.4.2 TiXmlNode∗ Admin::root `[private]`

Represent documents's root.

Definition at line 56 of file admin.h.

Referenced by addChannel(), addOnlyonCommand(), addSuperAdmin(), addTempSuperAdmin(), addUser(), Admin(), chanLevels(), channelExists(), commandOK(), commandsStatus(), delChannel(), delOnlyonCommand(), delUser(), disableCommand(), enableCommand(), getChannelsList(), getMaskLevel(), getUserLevel(), initFile(), isSuperAdmin(), maskIsSuperAdmin(), updateUserLevel(), and userExists().

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

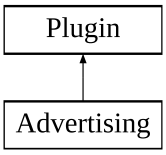
- src/plugins/admin.h
- src/plugins/admin.cpp

6.2 Advertising Class Reference

Plugin managing ads.

```
#include <advertising.h>
```

Inheritance diagram for Advertising::



Public Member Functions

- Advertising (BotKernel $*$)

  Constructor.

- time_t addAdvertise (string, unsigned int, unsigned int, string, string)

  Add an advertise to the advertise list.

- bool delAdvertise (string)

  Delete an advertise from the file.

- bool adExists (string)

  Tell if an advertise exists.

- vector< string > getAdvertiseInfos (string)

  Get advertise infos.

- vector< string > getAdvertisesList ()

  Get advertises list.

- void deleteOutdatedAds ()

  Delete outdated ads.

- void launchAdvertise (BotKernel $*$, string, unsigned int)

  launch an add

Private Member Functions

- void initFile ()

  Initialize the XML file.

Private Attributes

- TiXmlDocument ∗ doc

  Represent the xml document.

- TiXmlNode ∗ root

  Represent documents's root.

6.2.1 Detailed Description

Plugin managing ads.

This plugin makes the bot display ads on channels

Definition at line 51 of file advertising.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Advertising::Advertising (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 36 of file advertising.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, doc, BotKernel::getDatasDir(), IN_COMMAND_HANDLER, IN_LOOP, initFile(), launchAdvertise(), Plugin::name, root, Tools::strToInt(), and Plugin::version.

6.2.3 Member Function Documentation

6.2.3.1 time_t Advertising::addAdvertise (string *channel*, unsigned int *frequency*, unsigned int *until*, string *by*, string *text*)

Add an advertise to the advertise list.

Add an advertising to a channel, and save it int the XML file

Parameters:

channel Ad's channel

frequency Ad's frequency (in seconds)

until Number (in seconds) after witch the ad is outdated

by User who add the ad

text Ad's text

Returns:

ad's timestamp used to access to it

Definition at line 97 of file advertising.cpp.

References adExists(), doc, Tools::intToStr(), and root.

Referenced by addad().

6.2.3.2 bool Advertising::adExists (string *id*)

Tell if an advertise exists.

Tell if an ad exists

Parameters:

id Ad's id

Returns:

true if exists, else false

Definition at line 148 of file advertising.cpp.

References doc.

Referenced by addAdvertise().

6.2.3.3 bool Advertising::delAdvertise (string *id*)

Delete an advertise from the file.

Delete an advertise from the XML file

Parameters:

id Ad's id

Returns:

true is ad deleted, else false

Definition at line 128 of file advertising.cpp.

References doc.

Referenced by delad().

6.2.3.4 void Advertising::deleteOutdatedAds ()

Delete outdated ads.

Delete outdated ads

Definition at line 208 of file advertising.cpp.

References doc, and Tools::strToInt().

Referenced by cleanList().

6.2.3.5 vector< string > Advertising::getAdvertiseInfos (string *id*)

Get advertise infos.

Get advertise infos. Infos are :

- channel
- frequency
- until
- date
- sender
- text

Parameters:

id Ad's id

Definition at line 169 of file advertising.cpp.

References doc.

Referenced by adinfos(), and displayAdvertise().

6.2.3.6 vector< string > Advertising::getAdvertisesList ()

Get advertises list.

Get advertises list

Returns:

A vector containing id's and texts

Definition at line 190 of file advertising.cpp.

References doc, and Tools::strToInt().

Referenced by listads().

6.2.3.7 void Advertising::initFile () `[private]`

Initialize the XML file.

Initilaize the XML file by creating root and first childs (file empty structure)

Definition at line 68 of file advertising.cpp.

References doc, and root.

Referenced by Advertising().

6.2.3.8 void Advertising::launchAdvertise (BotKernel * *b*, string *id*, unsigned int *freq*)

launch an add

Launch an add

Parameters:

b Kernel pointer

id Ad's id

freq Ad's display frequency

Definition at line 82 of file advertising.cpp.

References BotKernel::addCountDown(), and displayAdvertise().

Referenced by Advertising().

6.2.4 Member Data Documentation

6.2.4.1 TiXmlDocument* Advertising::doc `[private]`

Represent the xml document.

Definition at line 55 of file advertising.h.

Referenced by addAdvertise(), adExists(), Advertising(), delAdvertise(), deleteOutdatedAds(), getAdvertiseInfos(), getAdvertisesList(), and initFile().

6.2.4.2 TiXmlNode* Advertising::root `[private]`

Represent documents's root.

Definition at line 57 of file advertising.h.

Referenced by addAdvertise(), Advertising(), and initFile().

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

- src/plugins/advertising.h
- src/plugins/advertising.cpp

6.3 AntiExcessFlood Struct Reference

Anti excess-flood variables.

```
#include <botkernel.h>
```

Public Attributes

- time_t last_decrease
- int penality

6.3.1 Detailed Description

Anti excess-flood variables.

Definition at line 56 of file botkernel.h.

6.3.2 Member Data Documentation

6.3.2.1 time_t AntiExcessFlood::last_decrease

Definition at line 58 of file botkernel.h.

Referenced by BotKernel::BotKernel(), and BotKernel::send().

6.3.2.2 int AntiExcessFlood::penality

Definition at line 59 of file botkernel.h.

Referenced by BotKernel::BotKernel(), and BotKernel::send().

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

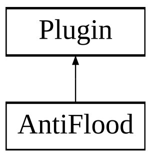
- src/botkernel.h

6.4 AntiFlood Class Reference

Plugin that able the bot to detect flood.

`#include <antiflood.h>`

Inheritance diagram for AntiFlood::



Public Member Functions

- AntiFlood (BotKernel ∗)

  Constructor.

6.4.1 Detailed Description

Plugin that able the bot to detect flood.

trustyRC use an anti-excess-flood system build in the kernel that controls output messages flow to make sure that the bot won't be killed by the server for "excess flood". This system make that the bot can sometimes take some time to answer. The main case is when a lot of users use bot's commands : the kernel will store answers and people will think that the bot lags. It can be considered as a flood attack from users. To avoid this problem, this plugin will watch messages' timestamps and if a message to treat is too old, it will drop it. Of course, only PRIVMSG are concerned, to don't drop a "PING" command for example. A configuration parameters exists to don't drop messages if they come from a super admin, even if there timestamp is too old.

Definition at line 51 of file antiflood.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AntiFlood::AntiFlood (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file antiflood.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, IN_-BEFORE_TREATMENT, Plugin::name, and Plugin::version.

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

- src/plugins/antiflood.h
- src/plugins/antiflood.cpp

6.5 BotKernel Class Reference

Bot kernel class.

`#include <botkernel.h>`

Public Member Functions

- BotKernel (string)

  Constructor.

- ∼BotKernel ()

  Destructor.

- plugin_function registerFunction (string, Plugin ∗, func_type, string, plugin_function, time_t, unsigned int)

  Register a function.

- void unregisterFunction (plugin_function)

  Unregister a finction.

- plugin_function addCountDown (Plugin ∗, plugin_function, Message ∗, unsigned int, unsigned int)

  Add a countdown.

- void run ()

  Launch kernel process.

- void stop ()

  Stop kernel process.

- void send (string)

  Send a message.

- void send (vector< string >)

  Send a list (in a vector) of messages.

- string getVersion ()

  Return the kernel version.

- string getDescription ()

  Return the kernel description.

- string getAuthor ()

  Return the kernel author.

- void setConnected (bool)

  set the bot connection state

- bool getConnected ()

  get the bot connection state

- LogFile ∗ getSysLog ()

  Return the SysLog object pointer.
- string getNick ()

  get bot's nick
- void setNick (string)

  set bot's nick
- bool loadPlugin (string, bool)

  Load a plugin.
- bool unloadPlugin (string, bool)

  Unload a plugin.
- pPlugin ∗ getPlugin (string)

  Send a plugin informations.
- vector< string > getPluginsList ()

  Send plugin list.
- time_t getStartTime ()

  Return power on timestamp.
- time_t getStartOnline ()

  Return connection timestamp.
- vector< CountDownFunction > ∗ getCountDowns ()

  get a pointer on countdowns
- ConfigurationFile ∗ getCONFF ()

  Return the configuration file object pointer.
- string getDatasDir ()

  Return datasdir.

Private Member Functions

- void displayLicenceHeader ()

  Display licence header.
- void initDirs ()

  Create needed directories.
- bool executeFunction (Message ∗, StructFunctionStorage)

  Execute a plugin function.

- plugin_function storeFunction (StructFunctionStorage ∗)

  Store a plugin function.

- void msgTreatment (Message ∗)

  Treat a received message.

- void connect ()

  Connect the bot.

- void reconnect ()

  Reconnect the bot.

- void loadPlugins (bool)

  Load all the plugins mentionned in the configuration file.

- void unloadMyPlugins (bool)

  Unload all the loaded plugins.

- bool pluginLoaded (string)

  Tell if a plugin is loaded (based on the plugin's name).

- bool pluginLoaded (void ∗)

  Tell if a plugin is loaded (based on the plugin's handle).

Private Attributes

- bool connected

  Bot connection state.

- string version

  Bot version.

- string description

  Bot description.

- string author

  Bot author.

- list< string > sendQueue

  Contains messages to send.

- bool turn

  True if the bot must process.

- string datasDir

  bot's datas directory

- bool verbose

True if received messages must be displayed.

- time_t startTime

  Timestamp representing the bot launch time.

- time_t startOnline

  Timestamp representing the bot connection time.

- LogFile ∗ myLog

  SysLog pointer.

- ConfigurationFile ∗ conff

  ConfigurationFile pointer.

- Socket ∗ sock

  Socket pointer.

- vector< StructFunctionStorage > in_loop_plugins

  "loop" plugins functions storage

- vector< StructFunctionStorage > in_free_command_handler_plugins

  "command handler" plugins functions storage

- vector< StructFunctionStorage > in_command_handler_plugins

  "free command handler" plugins functions storage

- vector< StructFunctionStorage > in_type_handler_plugins

  "type" plugins functions storage

- vector< StructFunctionStorage > in_before_treatment_plugins

  "before traitement" plugins functions storage

- vector< StructFunctionStorage > in_all_msgs_plugins

  "all messages" plugins functions storage

- vector< StructFunctionStorage > in_first_word_plugins

  "First word check" plugins functions storage

- vector< StructFunctionStorage > out_all_msgs_plugins

  "all outgoing messages" plugins functions storage

- vector< pPlugin > myPlugins

  plugins object en headers

- vector< CountDownFunction > countDowns

  Countdowns storage.

- string nick

  Bot's nick.

- AntiExcessFlood AEX

  Anti excess flood variables.

6.5.1 Detailed Description

Bot kernel class.

This class manages the bot kernel. It can load plugins, and uses all project class

Definition at line 83 of file botkernel.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BotKernel::BotKernel (string *confFile*)

Constructor.

The class constructor. Initialize private attributes. Parse the configuration file and initialize bot parameters and objects. Then load plugins

Parameters:

confFile Configuration file path (including name)

Postcondition:

An object is contructed

Definition at line 41 of file botkernel.cpp.

References ConfigurationFile::addProtectedKey(), AEX, author, conff, connected, countDowns, datasDir, description, displayLicenceHeader(), ConfigurationFile::getValue(), in_all_msgs_plugins, in_before_-treatment_plugins, in_command_handler_plugins, in_free_command_handler_plugins, in_loop_plugins, in_type_handler_plugins, initDirs(), AntiExcessFlood::last_decrease, ConfigurationFile::load(), loadPlugins(), myLog, LogFile::open(), AntiExcessFlood::penality, sendQueue, sock, startTime, turn, verbose, and version.

6.5.2.2 BotKernel::∼BotKernel ()

Destructor.

The class destructor Unload plugins, stop threads, and delete objects

Definition at line 89 of file botkernel.cpp.

References conff, INFO, LogFile::log(), myLog, sock, and unloadMyPlugins().

6.5.3 Member Function Documentation

6.5.3.1 plugin_function BotKernel::addCountDown (Plugin ∗ *p*, plugin_function *f*, Message ∗ *m*, unsigned int *count*, unsigned int *timeout*)

Add a countdown.

Add a count down to the kernel. Functions stored will be executed at the end of their count down A countdown function return false if it must not be deleted after execution (re launch countdown)

Parameters:

p Plugin's object

f Function to execute

m Message

count countdown

timeout Function timeout

Returns:

function adress

Definition at line 639 of file botkernel.cpp.

References conff, CountDownFunction::count, COUNTDOWN, countDowns, CountDownFunction::function, StructFunctionStorage::function, Plugin::getHandle(), Message::getMessage(), ConfigurationFile::getValue(), StructFunctionStorage::handle, StructFunctionStorage::highlightedWord, Tools::intToStr(), StructFunctionStorage::lastExec, LogFile::log(), CountDownFunction::msg, myLog, StructFunctionStorage::object, Tools::strToUnsignedInt(), StructFunctionStorage::symbole, StructFunctionStorage::timeout, CountDownFunction::timestamp, StructFunctionStorage::type, and WARNING.

Referenced by addad(), bannedHandler(), Advertising::launchAdvertise(), launchSurvey(), secondaryNick(), and slapme().

6.5.3.2 void BotKernel::connect () `[private]`

Connect the bot.

Connect the bot to the IRC server. Connection parameters are picked in the configuration file.

Definition at line 102 of file botkernel.cpp.

References conff, Socket::connectSock(), ERROR, ConfigurationFile::getValue(), IRCProtocol::identify(), INFO, LogFile::log(), myLog, send(), setConnected(), setNick(), sock, startOnline, Tools::strToInt(), and turn.

Referenced by reconnect(), and run().

6.5.3.3 void BotKernel::displayLicenceHeader () `[private]`

Display licence header.

Display licence header

Definition at line 828 of file botkernel.cpp.

References author, and version.

Referenced by BotKernel().

6.5.3.4 bool BotKernel::executeFunction (Message ∗ *m*, StructFunctionStorage *pfs*) `[private]`

Execute a plugin function.

Execute a function, managing the timeout

Parameters:

m Message pointer

pfs Function storage

Returns:

True if the message treatment must continue, else false

Definition at line 719 of file botkernel.cpp.

References ThreadParams::b, StructFunctionStorage::back, CPPThread::exec(), ThreadParams::function, Message::getNickSender(), CPPThread::join(), LogFile::log(), ThreadParams::msg, myLog, OUT_-ALL_MSGS, ThreadParams::sem, send(), IRCProtocol::sendNotice(), StructFunctionStorage::symbole, CPPThread::terminate(), threadFunc(), StructFunctionStorage::timeout, StructFunctionStorage::type, and WARNING.

Referenced by msgTreatment(), run(), and send().

6.5.3.5 string BotKernel::getAuthor ()

Return the kernel author.

Get kernel author

Returns:

Kernel's author

Definition at line 898 of file botkernel.cpp.

References author.

6.5.3.6 ConfigurationFile ∗ BotKernel::getCONFF ()

Return the configuration file object pointer.

Gives A pointer to the ConfigurationFile's kernel objet

Returns:

A pointer to the ConfigurationFile's kernel objet

Definition at line 889 of file botkernel.cpp.

References conff.

Referenced by addsuperadmin(), addtempsuperadmin(), allowedCommandCheck(), autoop(), autovoice(), ban(), banmask(), bannedHandler(), bzsearch(), checkBug(), danstonchat(), deletekey(), delsuperadmin(), flushconffile(), getconfvalue(), LogFactory::getLoggedChannels(), getMyFirstNick(), LogFactory::hasToBeLogged(), help(), joinHandler(), kickHandler(), lamoule(), launchSurvey(), loadconffile(), modeHandler(), modeHandlerProtect(), onEndOfMOTD(), planet(), player(), prefix(), protectmodes(), protecttopic(), purifyFile(), randomKick(), rejoinChan(), RemoteControl::RemoteControl(), secondaryNick(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), testMsgTimestamp(), top5(), topicHandler(), toptotal(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), vote(), and wiki().

6.5.3.7 bool BotKernel::getConnected ()

get the bot connection state

get the bot connexion state

Returns:

true for connected, false for disconnected

Definition at line 953 of file botkernel.cpp.

References connected.

6.5.3.8 vector< CountDownFunction > * BotKernel::getCountDowns ()

get a pointer on countdowns

Get a pointer on countdowns

Returns:

The pointer

Definition at line 1031 of file botkernel.cpp.

References countDowns.

Referenced by clearCountDowns(), and getnbcountdowns().

6.5.3.9 string BotKernel::getDatasDir ()

Return datasdir.

Get bot's datas directory

Returns:

bot's datas directory

Definition at line 1040 of file botkernel.cpp.

References datasDir.

Referenced by Admin::Admin(), Advertising::Advertising(), FedoraProject::FedoraProject(), greplog(), Ignore::Ignore(), Lamoule::Lamoule(), lastseen(), listlibs(), LogFactory::LogFactory(), Moderation::Moderation(), LogFactory::newLog(), Quotes::Quotes(), reloadfas(), run(), and sysinfos().

6.5.3.10 string BotKernel::getDescription ()

Return the kernel description.

Get kernel description

Returns:

Kernel description

Definition at line 916 of file botkernel.cpp.

References description.

6.5.3.11 string BotKernel::getNick ()

get bot's nick

Gives bot's nick

Returns:

bot's nick

Definition at line 934 of file botkernel.cpp.

References nick.

Referenced by banmask(), checkConnection(), clearOutBans(), joinHandler(), kick(), kickall(), kickHandler(), masskick(), modeHandler(), modeHandlerProtect(), nick_changed(), nickHandler(), onEndOfMOTD(), onJoin(), onKick(), onPart(), partHandler(), quitHandler(), randomKick(), sendHandler(), topicHandler(), unop(), and unopall().

6.5.3.12 pPlugin $*$ BotKernel::getPlugin (string *name*)

Send a plugin informations.

Return a plugin informations Informations are name, handle, constructor, destructor ... witch name is given by the "name" parameter

Parameters:

name Plugin name

Returns:

Plugin storage pointer. NULL if the plugin is not loaded

Definition at line 975 of file botkernel.cpp.

References myPlugins, and pluginLoaded().

Referenced by addad(), addIgnore(), adinfos(), ban(), Moderation::checkAccess(), Moderation::checkMode(), LogFactory::cleanLogs(), clearOutBans(), delad(), deleteplayer(), delIgnore(), delQuote(), Moderation::getChanUsersList(), LogFactory::getLoggedChannels(), Moderation::hasOpPrivileges(), ignoreList(), increase(), invite(), isIgnored(), joinHandler(), kickHandler(), lamoule(), listads(), listlibs(), listmodules(), load(), loadnocheck(), modeHandler(), modeHandlerProtect(), moduleinfos(), myThread(), nextscore(), partHandler(), protectmodes(), protecttopic(), quitHandler(), quoteInfos(), reauth(), reloadfas(), stopSurvey(), testMsgTimestamp(), topicHandler(), topicJoin(), unload(), unloadnocheck(), unprotectmodes(), and unprotecttopic().

6.5.3.13 vector$<$ string $>$ BotKernel::getPluginsList ()

Send plugin list.

Return the plugins list

Returns:

A vector containing plugins names

Definition at line 999 of file botkernel.cpp.

References myPlugins.

Referenced by listmodules().

6.5.3.14 time_t BotKernel::getStartOnline ()

Return connection timestamp.

Return connection timestamp

Returns:

connection timestamp

Definition at line 1022 of file botkernel.cpp.

References startOnline.

Referenced by online().

6.5.3.15 time_t BotKernel::getStartTime ()

Return power on timestamp.

Return power on timestamp

Returns:

Power on timestamp

Definition at line 1013 of file botkernel.cpp.

References startTime.

Referenced by uptime().

6.5.3.16 LogFile ∗ BotKernel::getSysLog ()

Return the SysLog object pointer.

Gives A pointer to the LogFile's kernel objet

Returns:

A pointer to the LogFile's kernel objet

Definition at line 925 of file botkernel.cpp.

References myLog.

Referenced by addIgnore(), addOnlyon(), addsuperadmin(), addtempsuperadmin(), bannedHandler(), checkConnection(), clearCountDowns(), deletekey(), delIgnore(), delOnlyon(), delsuperadmin(), disable(), disconnect(), enable(), error(), FedoraProject::FedoraProject(), flushconffile(), joinChannel(), kickall(), kickHandler(), leaveChannel(), load(), loadconffile(), loadnocheck(), LogFactory::LogFactory(), masskick(), onEndOfMOTD(), onInvite(), opall(), randomKick(), rejoinChan(), reloadfas(), reset(), secondaryNick(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), RemoteControl::tcpServer(), unload(), unloadnocheck(), unopall(), unvoiceall(), and voiceall().

6.5.3.17 string BotKernel::getVersion ()

Return the kernel version.

Get kernel version

Returns:

Kernel version

Definition at line 907 of file botkernel.cpp.

References version.

Referenced by ctcp_version(), and version().

6.5.3.18 void BotKernel::initDirs () `[private]`

Create needed directories.

Create needed directories

Definition at line 839 of file botkernel.cpp.

References conff, datasDir, and ConfigurationFile::getFilePath().

Referenced by BotKernel().

6.5.3.19 bool BotKernel::loadPlugin (string *fileName*, bool *checkDependancy*)

Load a plugin.

Load a plugin, to make all its functions usable by the bot

Postcondition:

The kernel can use new functions

Parameters:

fileName Dynamic librairy containing the plugins. Just the name, not the path

checkDependancy Set this to true to check for plugin requirements

Returns:

true if the plugin has correctly been loaded, else false

Definition at line 291 of file botkernel.cpp.

References Plugin::checkMembers(), pPlugin::creator, datasDir, pPlugin::destructor, ERROR, error(), StructFunctionStorage::function, Plugin::getFunctions(), Plugin::getName(), Plugin::getRequirements(), StructFunctionStorage::handle, pPlugin::handle, INFO, LogFile::log(), myLog, myPlugins, pPlugin::name, pPlugin::object, pluginLoaded(), Plugin::setHandle(), storeFunction(), StructFunctionStorage::symbole, and WARNING.

Referenced by load(), loadnocheck(), and loadPlugins().

6.5.3.20 void BotKernel::loadPlugins (bool *checkDependancy*) `[private]`

Load all the plugins mentionned in the configuration file.

Load all the plugins mentionned in the configuration file.

Parameters:

checkDependancy Set this to true to check for plugin requirements

Definition at line 268 of file botkernel.cpp.

References conff, ConfigurationFile::getValue(), INFO, Tools::intToStr(), loadPlugin(), LogFile::log(), myLog, and Tools::stringToVector().

Referenced by BotKernel().

6.5.3.21 void BotKernel::msgTreatment (Message ∗ *msg*) `[private]`

Treat a received message.

Treat a message according to its type. Each type of plugin is tested to watch if the message is concerned

Parameters:

msg Pointer to the message object

Definition at line 674 of file botkernel.cpp.

References conff, executeFunction(), Message::getPart(), Message::getSplit(), ConfigurationFile::getValue(), in_all_msgs_plugins, in_command_handler_plugins, in_first_word_plugins, in_free_command_handler_plugins, in_type_handler_plugins, Message::nbParts(), and Tools::to_lower().

Referenced by run().

6.5.3.22 bool BotKernel::pluginLoaded (void ∗ *handle*) `[private]`

Tell if a plugin is loaded (based on the plugin's handle).

Check if a plugin is loaded. The result is based on the plugin handle

Parameters:

handle Plugin handle

Returns:

True if the given plugin is loaded, else false

Definition at line 504 of file botkernel.cpp.

References myPlugins.

6.5.3.23 bool BotKernel::pluginLoaded (string *name*) `[private]`

Tell if a plugin is loaded (based on the plugin's name).

Check if a plugin is loaded. The result is based on the plugin NAME (object attribute), not on the file name containing the plugin

Parameters:

name Plugin name

Returns:

True if the given plugin is loaded, else false

Definition at line 488 of file botkernel.cpp.

References myPlugins.

Referenced by getPlugin(), loadPlugin(), and registerFunction().

6.5.3.24 void BotKernel::reconnect () `[private]`

Reconnect the bot.

Makes the bot reload the connection All connections objects a re-initialised

Definition at line 122 of file botkernel.cpp.

References Socket::closeSock(), connect(), INFO, LogFile::log(), myLog, setConnected(), and sock.

Referenced by run().

6.5.3.25 plugin_function BotKernel::registerFunction (string *hlword*, Plugin ∗ *object*, func_type *type*, string *symbole*, plugin_function *function*, time_t *lastExec*, unsigned int *timeout*)

Register a function.

Register a new function. The function is added in the plugins queues and can be triggered

Parameters:

hlword higlited word

object plugin's object

type function type

symbole symbole to execute

function function to execute

lastExec last exec

timeout fuction timeout

Returns:

a pointer on the function, NULL if the plugin does no exists

Definition at line 556 of file botkernel.cpp.

References StructFunctionStorage::function, Plugin::getHandle(), StructFunctionStorage::handle, StructFunctionStorage::highlightedWord, StructFunctionStorage::lastExec, StructFunctionStorage::object, pluginLoaded(), storeFunction(), StructFunctionStorage::symbole, StructFunctionStorage::timeout, and StructFunctionStorage::type.

Referenced by launchSurvey().

6.5.3.26 void BotKernel::run ()

Launch kernel process.

Kernel process Manage objects to receive, send and treat messages

Definition at line 136 of file botkernel.cpp.

References Socket::closeSock(), connect(), connected, countDowns, executeFunction(), getDatasDir(), Socket::getState(), in_before_treatment_plugins, in_loop_plugins, INFO, Tools::intToStr(), LogFile::log(), Tools::log(), msgTreatment(), myLog, Socket::receive(), reconnect(), setConnected(), Message::setMessage(), sock, Tools::stringToVector(), Tools::strToInt(), turn, verbose, and WARNING.

6.5.3.27 void BotKernel::send (vector< string > *vec*)

Send a list (in a vector) of messages.

Send strings to the server

Parameters:

vec Vector containing strings to send

Definition at line 818 of file botkernel.cpp.

References send().

6.5.3.28 void BotKernel::send (string *str*)

Send a message.

Send a string to the server

Parameters:

str String to send

Definition at line 756 of file botkernel.cpp.

References AEX, conff, executeFunction(), Message::getMessage(), ConfigurationFile::getValue(), AntiExcessFlood::last_decrease, LogFile::log(), myLog, out_all_msgs_plugins, AntiExcessFlood::penality, Socket::sendStr(), setConnected(), Message::setMessage(), sock, and WARNING.

Referenced by addad(), addIgnore(), addOnlyon(), addQuote(), addsuperadmin(), addtempsuperadmin(), adinfos(), autoop(), autovoice(), ball(), ban(), bandel(), baninfos(), banlist(), banmask(), bug(), bzsearch(), chanlev(), checkBug(), checkConnection(), clearCountDowns(), clearOutBans(), commandsStatus(), connect(), ctcp_ping(), ctcp_version(), cycleChannel(), danstonchat(), delad(), deletekey(), deleteplayer(), delIgnore(), delOnlyon(), delQuote(), delsuperadmin(), disable(), disconnect(), displayAdvertise(), displayPaste(), enable(), endSurvey(), executeFunction(), fas(), flushconffile(), getconfvalue(), getMyFirstNick(), getnbcountdowns(), greplog(), help(), hl(), host2ip(), ignoreList(), increase(), invite(), ip2host(), isIgnored(), joinChannel(), joinHandler(), kick(), kickall(), kickHandler(), lamoule(), lastQuote(), lastseen(), launchSurvey(), leaveChannel(), listads(), listlibs(), listmodules(), load(), loadconffile(), loadnocheck(), masskick(), modeHandler(), modeHandlerProtect(), moduleinfos(), myFunction(), nextscore(), notice(), onEndOfMOTD(), onInvite(), onJoin(), online(), op(), opall(), partHandler(), pinged(), planet(), player(), prefix(), protectmodes(), protecttopic(), q3(), quitHandler(), quote(), quoteInfos(), randomKick(), raw(), rejoinChan(), reloadfas(), reloadUsers(), searchQuote(), secondaryNick(), send(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), slapme(), slapUser(), stopSurvey(), superadminlist(), sysinfos(), tele(), tell(), top5(), topic(), topicHandler(), topshot(), toptotal(), trad(), unautoop(), unautovoice(), unbanall(), unload(), unloadnocheck(), unop(), unopall(), unprotect-modes(), unprotecttopic(), unvoice(), unvoiceall(), uptime(), version(), voice(), voiceall(), vote(), warsow(), whoami(), whoowns(), and wiki().

6.5.3.29 void BotKernel::setConnected (bool *state*)

set the bot connection state

Set the bot connexion state This state is important to make the bot reset if disconnected

Parameters:

state true for connected, false for disconnected

Definition at line 963 of file botkernel.cpp.

References connected.

Referenced by checkConnection(), connect(), reconnect(), reset(), run(), and send().

6.5.3.30 void BotKernel::setNick (string *nick*)

set bot's nick

Set bot's nick

Parameters:

nick new nick

Definition at line 943 of file botkernel.cpp.

Referenced by connect(), getMyFirstNick(), secondaryNick(), and setNick().

6.5.3.31 void BotKernel::stop ()

Stop kernel process.

Stop the bot

Definition at line 258 of file botkernel.cpp.

References INFO, LogFile::log(), myLog, and turn.

Referenced by disconnect().

6.5.3.32 plugin_function BotKernel::storeFunction (StructFunctionStorage ∗ *func*) `[private]`

Store a plugin function.

Store a function and return it's id

Parameters:

func Function to store

Returns:

function'address. NULL if nothing inserted

Definition at line 519 of file botkernel.cpp.

References StructFunctionStorage::function, IN_ALL_MSGS, in_all_msgs_plugins, IN_BEFORE_-TREATMENT, in_before_treatment_plugins, IN_COMMAND_HANDLER, in_command_handler_-plugins, IN_FIRST_WORD, in_first_word_plugins, IN_FREE_COMMAND_HANDLER, in_free_-command_handler_plugins, IN_LOOP, in_loop_plugins, IN_TYPE_HANDLER, in_type_handler_-plugins, LogFile::log(), myLog, OUT_ALL_MSGS, out_all_msgs_plugins, StructFunctionStorage::symbole, StructFunctionStorage::type, and WARNING.

Referenced by loadPlugin(), and registerFunction().

6.5.3.33 void BotKernel::unloadMyPlugins (bool *checkDependancy*) `[private]`

Unload all the loaded plugins.

Unload all the loaded plugins. Makes the bot run only with the kernel (no functionnalities)

Parameters:

checkDependancy Set this to true to check for plugin requirements

Definition at line 473 of file botkernel.cpp.

References myPlugins, and unloadPlugin().

Referenced by ∼BotKernel().

6.5.3.34 bool BotKernel::unloadPlugin (string *name*, bool *checkDependancy*)

Unload a plugin.

Unload a plugin. Functions provided by the plugin can no more be used by the kernel.

Parameters:

name Plugin name
checkDependancy Set this to true to check for plugin requirements

Returns:

true if the plugin has correctly been unloaded, else false

Definition at line 382 of file botkernel.cpp.

References ERROR, in_all_msgs_plugins, in_before_treatment_plugins, in_command_handler_plugins, in_first_word_plugins, in_free_command_handler_plugins, in_loop_plugins, in_type_handler_plugins, INFO, LogFile::log(), myLog, myPlugins, and out_all_msgs_plugins.

Referenced by unload(), unloadMyPlugins(), and unloadnocheck().

6.5.3.35 void BotKernel::unregisterFunction (plugin_function *func*)

Unregister a finction.

Unregister a function

Parameters:

func Function pointer

Definition at line 577 of file botkernel.cpp.

References countDowns, in_all_msgs_plugins, in_before_treatment_plugins, in_command_handler_-plugins, in_first_word_plugins, in_free_command_handler_plugins, in_loop_plugins, in_type_handler_-plugins, and out_all_msgs_plugins.

Referenced by endSurvey(), myUselessFunction(), and stopSurvey().

6.5.4 Member Data Documentation

6.5.4.1 AntiExcessFlood BotKernel::AEX `[private]`

Anti excess flood variables.

Definition at line 197 of file botkernel.h.

Referenced by BotKernel(), and send().

6.5.4.2 string BotKernel::author `[private]`

Bot author.

Definition at line 147 of file botkernel.h.

Referenced by BotKernel(), displayLicenceHeader(), and getAuthor().

6.5.4.3 ConfigurationFile* BotKernel::conff `[private]`

ConfigurationFile pointer.

Definition at line 163 of file botkernel.h.

Referenced by addCountDown(), BotKernel(), connect(), getCONFF(), initDirs(), loadPlugins(), msgTreatment(), send(), and ~BotKernel().

6.5.4.4 bool BotKernel::connected `[private]`

Bot connection state.

Definition at line 141 of file botkernel.h.

Referenced by BotKernel(), getConnected(), run(), and setConnected().

6.5.4.5 vector<CountDownFunction> BotKernel::countDowns `[private]`

Countdowns storage.

Definition at line 185 of file botkernel.h.

Referenced by addCountDown(), BotKernel(), getCountDowns(), run(), and unregisterFunction().

6.5.4.6 string BotKernel::datasDir `[private]`

bot's datas directory

Definition at line 153 of file botkernel.h.

Referenced by BotKernel(), getDatasDir(), initDirs(), and loadPlugin().

6.5.4.7 string BotKernel::description `[private]`

Bot description.

Definition at line 145 of file botkernel.h.

Referenced by BotKernel(), and getDescription().

6.5.4.8 vector<StructFunctionStorage> BotKernel::in_all_msgs_plugins `[private]`

"all messages" plugins functions storage

Definition at line 177 of file botkernel.h.

Referenced by BotKernel(), msgTreatment(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.9 vector<StructFunctionStorage> BotKernel::in_before_treatment_plugins `[private]`

"before traitement" plugins functions storage

Definition at line 175 of file botkernel.h.

Referenced by BotKernel(), run(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.10 vector<StructFunctionStorage> BotKernel::in_command_handler_plugins `[private]`

"free command handler" plugins functions storage

Definition at line 171 of file botkernel.h.

Referenced by BotKernel(), msgTreatment(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.11 vector<StructFunctionStorage> BotKernel::in_first_word_plugins `[private]`

"First word check" plugins functions storage

Definition at line 179 of file botkernel.h.

Referenced by msgTreatment(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.12 vector<StructFunctionStorage> BotKernel::in_free_command_handler_plugins `[private]`

"command handler" plugins functions storage

Definition at line 169 of file botkernel.h.

Referenced by BotKernel(), msgTreatment(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.13 vector<StructFunctionStorage> BotKernel::in_loop_plugins `[private]`

"loop" plugins functions storage

Definition at line 167 of file botkernel.h.

Referenced by BotKernel(), run(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.14 vector<StructFunctionStorage> BotKernel::in_type_handler_plugins [private]

"type" plugins functions storage

Definition at line 173 of file botkernel.h.

Referenced by BotKernel(), msgTreatment(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.15 LogFile* BotKernel::myLog [private]

SysLog pointer.

Definition at line 161 of file botkernel.h.

Referenced by addCountDown(), BotKernel(), connect(), executeFunction(), getSysLog(), loadPlugin(), loadPlugins(), reconnect(), run(), send(), stop(), storeFunction(), unloadPlugin(), and ~BotKernel().

6.5.4.16 vector<pPlugin> BotKernel::myPlugins [private]

plugins object en headers

Definition at line 183 of file botkernel.h.

Referenced by getPlugin(), getPluginsList(), loadPlugin(), pluginLoaded(), unloadMyPlugins(), and unloadPlugin().

6.5.4.17 string BotKernel::nick [private]

Bot's nick.

Definition at line 187 of file botkernel.h.

Referenced by getNick().

6.5.4.18 vector<StructFunctionStorage> BotKernel::out_all_msgs_plugins [private]

"all outgoing messages" plugins functions storage

Definition at line 181 of file botkernel.h.

Referenced by send(), storeFunction(), unloadPlugin(), and unregisterFunction().

6.5.4.19 list<string> BotKernel::sendQueue [private]

Contains messages to send.

Definition at line 149 of file botkernel.h.

Referenced by BotKernel().

6.5.4.20 Socket∗ BotKernel::sock `[private]`

Socket pointer.

Definition at line 165 of file botkernel.h.

Referenced by BotKernel(), connect(), reconnect(), run(), send(), and ∼BotKernel().

6.5.4.21 time_t BotKernel::startOnline `[private]`

Timestamp representing the bot connection time.

Definition at line 159 of file botkernel.h.

Referenced by connect(), and getStartOnline().

6.5.4.22 time_t BotKernel::startTime `[private]`

Timestamp representing the bot launch time.

Definition at line 157 of file botkernel.h.

Referenced by BotKernel(), and getStartTime().

6.5.4.23 bool BotKernel::turn `[private]`

True if the bot must process.

Definition at line 151 of file botkernel.h.

Referenced by BotKernel(), connect(), run(), and stop().

6.5.4.24 bool BotKernel::verbose `[private]`

True if received messages must be displayed.

Definition at line 155 of file botkernel.h.

Referenced by BotKernel(), and run().

6.5.4.25 string BotKernel::version `[private]`

Bot version.

Definition at line 143 of file botkernel.h.

Referenced by BotKernel(), displayLicenceHeader(), and getVersion().

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

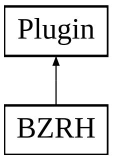
- src/botkernel.h
- src/botkernel.cpp

6.6 BZRH Class Reference

BZRH provides commands t query bugzilla.redhat.com.

`#include <bzrh.h>`

Inheritance diagram for BZRH::



Public Member Functions

- BZRH (BotKernel ∗)

  Constructor.

- vector< string > searchBugs (string, string)

  Search for bugs on bugzilla.

- string getBugInfos (string, bool)

  Retrieve informations about a bug.

Static Public Member Functions

- static int writer (char ∗, size_t, size_t, string ∗)

  writer call back function used by curl

6.6.1 Detailed Description

BZRH provides commands t query bugzilla.redhat.com.

BZRH (Bugzilla RedHat) is a plugin that allow users to query bugzilla.redhat.com to retrieve informations about bugs. This plugin uses libcurl to access to the website threw HTTPS protocol

Definition at line 44 of file bzrh.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 BZRH::BZRH (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file bzrh.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_ALL_MSGS, IN_-COMMAND_HANDLER, Plugin::name, and Plugin::version.

6.6.3 Member Function Documentation

6.6.3.1 string BZRH::getBugInfos (string *bug*, bool *displayNotFound*)

Retrieve informations about a bug.

Retrieve informations about a bug

Parameters:

bug	Bug number
displayNotFound	Tell if "bug not found must be displayed

Returns:

Bug's infos

Definition at line 140 of file bzrh.cpp.

References Tools::cleanHTML(), Tools::urlencode(), and writer().

Referenced by bug(), and checkBug().

6.6.3.2 vector$<$ string $>$ BZRH::searchBugs (string *pattern*, string *max*)

Search for bugs on bugzilla.

Search for bugs on bugzilla

Parameters:

pattern	Search pattern
max	max results

Returns:

vector containing results

Definition at line 51 of file bzrh.cpp.

References Tools::cleanHTML(), Tools::intToStr(), Tools::strToUnsignedInt(), Tools::urlencode(), and writer().

Referenced by bzsearch().

6.6.3.3 int BZRH::writer (char $*$ *data*, size_t *size*, size_t *nmemb*, string $*$ *buffer*) `[static]`

writer call back function used by curl

Definition at line 228 of file bzrh.cpp.

Referenced by getBugInfos(), and searchBugs().

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

- src/plugins/bzrh.h
- src/plugins/bzrh.cpp

6.7 Channel Class Reference

Channel management class.

`#include <channel.h>`

Public Member Functions

- Channel (string)

  Constructor.

- ∼Channel ()

  Destructor.

- string getName ()

  return the channel name

- bool addUser (string, string, string, string)

  Add a user to the chan.

- bool delUserByNick (string)

  Del a user from the chan.

- bool delUserByHost (string)

  Del a user from the chan.

- string ∗ getInfosByNick (string)

  Return infos about a user.

- string getNickByHost (string)

  Return the nick associated to a given host.

- string getStatusByNick (string)

  Return the status associated to a given nick.

- string getStatusByHost (string)

  Return the status associated to a given host.

- string getHostByNick (string)

  Return the host associated to a given nick.

- string getIdentByNick (string)

  Return the ident associated to a nick.

- string getIdentByHost (string)

  Return the ident associated to a host.

- bool setNickByNick (string, string)

  Change a nick.

- bool setNickByHost (string, string)

  Change the nick associated to a host.

- bool updateStatusByNick (string, char, char)

  Update the status associated to a nick.

- bool checkNickAccess (string, char)

  Check if a nick owns a given access.

- void truncateUsersList ()

  Erase all users from the channel.

- vector< string ∗ > getUsers ()

  Gives users vector.

- time_t getLastWhoUpdate ()

  Get last update.

- void notifyWho ()

  Notify a WHO update.

- string getTopic ()

  get topic

- void setTopic (string)

  set topic

- string ∗ getLastPartInfos ()

  get informations about the last user who left channel

- bool isOnChannel (string)

  Tell if a nick is on the channel.

Private Member Functions

- vector< string ∗ >::iterator getIterator (string, unsigned int)

  Return a nick iterator in a vector.

Private Attributes

- vector< string ∗ > users

  Contain all the chan users.

- string name

  Channel name.

- time_t lastWhoUpdate

Keep last WHO update.

- string topic

  stores channel's topic

- string lastPart [4]

  Stores informations about the last user who left channel.

6.7.1 Detailed Description

Channel management class.

This class stores and manage all informations about a channel

Definition at line 44 of file channel.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Channel::Channel (string *name*)

Constructor.

Class constructor Initialize private attributes

Parameters:

name Channel name

Definition at line 36 of file channel.cpp.

References lastPart, lastWhoUpdate, topic, and users.

6.7.2.2 Channel::∼Channel ()

Destructor.

Destructor Clear private attributes

Definition at line 51 of file channel.cpp.

References truncateUsersList().

6.7.3 Member Function Documentation

6.7.3.1 bool Channel::addUser (string *nick*, string *host*, string *ident*, string *status*)

Add a user to the chan.

Add a user to the channel It's then possible to follow status

Parameters:

nick User nick

host User host

ident User ident
status User status

Returns:

true if operation ok, else false (already added ?)

Definition at line 74 of file channel.cpp.

References getHostByNick(), and users.

6.7.3.2 bool Channel::checkNickAccess (string *nick*, char *access*)

Check if a nick owns a given access.

Check if a given nick owns a given access

Parameters:

nick The nick witch you want to check access
access Access to test (o,v,etc ...)

Returns:

true if the nick owns the access, else false

Definition at line 347 of file channel.cpp.

References getIterator(), and users.

6.7.3.3 bool Channel::delUserByHost (string *host*)

Del a user from the chan.

Del a user from the chan according to a given host

Parameters:

host host's user to delete

Postcondition:

lastPart has been updated

Returns:

true if the user ahs been deleted, else false

Definition at line 140 of file channel.cpp.

References getIterator(), lastPart, and users.

6.7.3.4 bool Channel::delUserByNick (string *nick*)

Del a user from the chan.

Del a user from the chan according to a given nick

Parameters:

nick Nick to delete

Postcondition:

lastPart has been updated

Returns:

true if the user ahs been deleted, else false

Definition at line 118 of file channel.cpp.

References getIterator(), lastPart, and users.

6.7.3.5 string Channel::getHostByNick (string *nick*)

Return the host associated to a given nick.

Give the host corresponding to a given nick

Parameters:

nick nick from witch you want to get the host

Returns:

Host corresponding to the given nick (empty string if no match found)

Definition at line 224 of file channel.cpp.

References getIterator(), and users.

Referenced by addUser().

6.7.3.6 string Channel::getIdentByHost (string *host*)

Return the ident associated to a host.

Give the host corresponding to a given host

Parameters:

host host from witch you want to get the ident

Returns:

Ident corresponding to the given host (empty string if no match found)

Definition at line 254 of file channel.cpp.

References getIterator(), and users.

6.7.3.7 string Channel::getIdentByNick (string *nick*)

Return the ident associated to a nick.

Give the ident corresponding to a given nick

Parameters:

nick nick from witch you want to get the ident

Returns:

Ident corresponding to the given nick (empty string if no match found)

Definition at line 239 of file channel.cpp.

References getIterator(), and users.

6.7.3.8 string * Channel::getInfosByNick (string *nick*)

Return infos about a user.

Return informations about a nick, in this order : tab[0]=nick; tab[1]=host; tab[2]=ident; tab[3]=status;

Parameters:

nick User nick

Returns:

array with informations

Definition at line 178 of file channel.cpp.

References getIterator(), and users.

Referenced by isOnChannel().

6.7.3.9 vector< string * >::iterator Channel::getIterator (string *comparator*, unsigned int *index*) `[private]`

Return a nick iterator in a vector.

Return a vector iterator according to arguments

Parameters:

comparator String to compair with the vector

index User's infos index

Returns:

Iterator usable for a vector

Definition at line 96 of file channel.cpp.

References users.

Referenced by checkNickAccess(), delUserByHost(), delUserByNick(), getHostByNick(), getIdentByHost(), getIdentByNick(), getInfosByNick(), getNickByHost(), getStatusByHost(), getStatusByNick(), setNickByHost(), setNickByNick(), and updateStatusByNick().

6.7.3.10 string ∗ Channel::getLastPartInfos ()

get informations about the last user who left channel

Returns informations about the last user who left the channel

Returns:

Informations about the last user who left the channel

Definition at line 434 of file channel.cpp.

References lastPart.

6.7.3.11 time_t Channel::getLastWhoUpdate ()

Get last update.

Get last WHO update timestamp

Returns:

last update timestamp

Definition at line 399 of file channel.cpp.

References lastWhoUpdate.

6.7.3.12 string Channel::getName ()

return the channel name

Get channel name

Returns:

Channel name

Definition at line 60 of file channel.cpp.

References name.

6.7.3.13 string Channel::getNickByHost (string *host*)

Return the nick associated to a given host.

Give the nick corresponding to a given host

Parameters:

host host from witch you want to get the nick

Returns:

Nick corresponding to the given host (empty string if no match found)

Definition at line 160 of file channel.cpp.

References getIterator(), and users.

6.7.3.14 string Channel::getStatusByHost (string *host*)

Return the status associated to a given host.

Give the status corresponding to a given host

Parameters:

host host from witch you want to get the status

Returns:

Status corresponding to the given host (empty string if no match found)

Definition at line 209 of file channel.cpp.

References getIterator(), and users.

6.7.3.15 string Channel::getStatusByNick (string *nick*)

Return the status associated to a given nick.

Give the status corresponding to a given nick

Parameters:

nick Nick from witch you want to get the status

Returns:

Status corresponding to the given nick (empty string if no match found)

Definition at line 194 of file channel.cpp.

References getIterator(), and users.

6.7.3.16 string Channel::getTopic ()

get topic

get topic

Returns:

topic

Definition at line 416 of file channel.cpp.

References topic.

6.7.3.17 vector< string ∗ > Channel::getUsers ()

Gives users vector.

Return users vector informations Users a stored in a string tab like that : tab[0]=nick; tab[1]=host; tab[2]=ident; tab[3]=status;

Returns:

users vector informations

Definition at line 390 of file channel.cpp.

References users.

6.7.3.18 bool Channel::isOnChannel (string *nick*)

Tell if a nick is on the channel.

Tell if a nick is on the channel

Parameters:

nick Nick to check

Returns:

True if the nick is on the channel, else false

Definition at line 444 of file channel.cpp.

References getInfosByNick().

6.7.3.19 void Channel::notifyWho ()

Notify a WHO update.

Notify that a a WHO command has been sent for this channel

Definition at line 407 of file channel.cpp.

References lastWhoUpdate.

6.7.3.20 bool Channel::setNickByHost (string *host*, string *newnick*)

Change the nick associated to a host.

Change a user nick to an other one This one is found by the host

Parameters:

host User host for witch you want to change the nick

newnick New nick (instead of old one)

Returns:

True if the nick has been changed, else false (not found)

Definition at line 289 of file channel.cpp.

References getIterator(), and users.

6.7.3.21 bool Channel::setNickByNick (string *old*, string *newnick*)

Change a nick.

Change a user nick to an other one This one is found by the nick

Parameters:

old Nick to change

newnick New nick (instead of old one)

Returns:

True if the nick has been changed, else false (not found)

Definition at line 271 of file channel.cpp.

References getIterator(), and users.

6.7.3.22 void Channel::setTopic (string *topic*)

set topic

set topic

Parameters:

topic channel topic

Definition at line 425 of file channel.cpp.

6.7.3.23 void Channel::truncateUsersList ()

Erase all users from the channel.

Clear users list

Definition at line 371 of file channel.cpp.

References users.

Referenced by ∼Channel().

6.7.3.24 bool Channel::updateStatusByNick (string *nick*, char *sign*, char *mode*)

Update the status associated to a nick.

Change a user status to an other one This one is found by the nick

Parameters:

nick User nick for witch you want to change the status

sign + or - (+ for grant level, - for remove level)

mode user mode (v,o etc ...)

Returns:

True if the nick has been changed, else false (not found)

Definition at line 308 of file channel.cpp.

References getIterator(), and users.

6.7.4 Member Data Documentation

6.7.4.1 string Channel::lastPart[4] `[private]`

Stores informations about the last user who left channel.

Definition at line 109 of file channel.h.

Referenced by Channel(), delUserByHost(), delUserByNick(), and getLastPartInfos().

6.7.4.2 time_t Channel::lastWhoUpdate `[private]`

Keep last WHO update.

Definition at line 105 of file channel.h.

Referenced by Channel(), getLastWhoUpdate(), and notifyWho().

6.7.4.3 string Channel::name `[private]`

Channel name.

Definition at line 103 of file channel.h.

Referenced by getName().

6.7.4.4 string Channel::topic `[private]`

stores channel's topic

Definition at line 107 of file channel.h.

Referenced by Channel(), and getTopic().

6.7.4.5 vector<string*> Channel::users `[private]`

Contain all the chan users.

Definition at line 99 of file channel.h.

Referenced by addUser(), Channel(), checkNickAccess(), delUserByHost(), delUserByNick(), getHostByNick(), getIdentByHost(), getIdentByNick(), getInfosByNick(), getIterator(), getNickByHost(), getStatusByHost(), getStatusByNick(), getUsers(), setNickByHost(), setNickByNick(), truncateUsersList(), and updateStatusByNick().

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

- src/channel.h
- src/channel.cpp

6.8 ConfigurationFile Class Reference

Configuration file class.

```
#include <configurationfile.h>
```

Public Member Functions

- ConfigurationFile (string)

  Class constructor.

- ∼ConfigurationFile ()

  Class destructor.

- bool load ()

  Parse and load the configuration file.

- bool flush ()

  Flush the settings in the file.

- map< string, string > getConfig ()

  Return the configuration (keys and values, in a MAP container).

- void addProtectedKey (string)

  Add a key to protected list.

- string getValue (string, bool displayProtected=true)

  Return a value associated to a key.

- void setValue (string, string)

  Set a value to a key.

- bool delKey (string)

  Delete a key.

- string getFilePath ()

  Returns file path.

Private Attributes

- map< string, string > config

  Configuration container.

- string file

  Configuration file path (and name).

- vector< string > protectedKeys

  Stores protected keys.

6.8.1 Detailed Description

Configuration file class.

Class that manage a configuration file. It uses a MAP containing values associated to keys

Definition at line 44 of file configurationfile.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 ConfigurationFile::ConfigurationFile (string *fname*)

Class constructor.

The class constructor. Initialize private attributes

Parameters:

fname Configuration file path (including name)

Postcondition:

An object is contructed

Definition at line 35 of file configurationfile.cpp.

References config, file, and protectedKeys.

6.8.2.2 ConfigurationFile::∼ConfigurationFile ()

Class destructor.

The class destructor

Definition at line 45 of file configurationfile.cpp.

6.8.3 Member Function Documentation

6.8.3.1 void ConfigurationFile::addProtectedKey (string *key*)

Add a key to protected list.

Add a key to the protected list

Parameters:

key Key to protect

Definition at line 135 of file configurationfile.cpp.

References protectedKeys.

Referenced by BotKernel::BotKernel().

6.8.3.2 bool ConfigurationFile::delKey (string *key*)

Delete a key.

Delete a key from the configuration file

Parameters:

key The key you want delete

Definition at line 176 of file configurationfile.cpp.

References config.

Referenced by deletekey().

6.8.3.3 bool ConfigurationFile::flush ()

Flush the settings in the file.

Flush all the configuration keys (with values) in the configuration file File format : key=value

Precondition:

The file must exist and be writeable

Postcondition:

All the configuration values are saved into the configuration file

Returns:

true if the configuration file could be open, else erturn false

Definition at line 104 of file configurationfile.cpp.

References config, and file.

Referenced by flushconffile().

6.8.3.4 map< string, string > ConfigurationFile::getConfig ()

Return the configuration (keys and values, in a MAP container).

Returns the MAP containing the configuration

Precondition:

The file must have been parsed

Returns:

a map container, containing the configuration value (with keys)

Definition at line 126 of file configurationfile.cpp.

References config.

6.8.3.5 string ConfigurationFile::getFilePath ()

Returns file path.

Get file path

Returns:

file path

Definition at line 193 of file configurationfile.cpp.

References file.

Referenced by BotKernel::initDirs().

6.8.3.6 string ConfigurationFile::getValue (string *key*, bool *displayProtected* = `true`)

Return a value associated to a key.

Return the value associated to a given key

Precondition:

The file must have been parsed

Parameters:

key The key from with you want to get the value

displayProtected If true, protected key/values will be displayed

Returns:

The value associated to the given key. Empty string if the key doesn't exists

Definition at line 147 of file configurationfile.cpp.

References config, Tools::isInVector(), and protectedKeys.

Referenced by BotKernel::addCountDown(), addsuperadmin(), addtempsuperadmin(), allowedCommandCheck(), autoop(), autovoice(), ban(), banmask(), bannedHandler(), BotKernel::BotKernel(), bzsearch(), checkBug(), BotKernel::connect(), danstonchat(), delsuperadmin(), getconfvalue(), LogFactory::getLoggedChannels(), getMyFirstNick(), LogFactory::hasToBeLogged(), help(), joinHandler(), kickHandler(), lamoule(), launchSurvey(), load(), BotKernel::loadPlugins(), modeHandler(), modeHandlerProtect(), BotKernel::msgTreatment(), onEndOfMOTD(), planet(), player(), prefix(), protectmodes(), protecttopic(), purifyFile(), randomKick(), rejoinChan(), RemoteControl::RemoteControl(), secondaryNick(), BotKernel::send(), setSuperAdminPass(), testMsgTimestamp(), top5(), topicHandler(), toptotal(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), vote(), and wiki().

6.8.3.7 bool ConfigurationFile::load ()

Parse and load the configuration file.

Read the configuration file and load it in a MAP containe File format : key=value # is the comment char

Precondition:

The file must exist and be readable

Postcondition:

The file is parsed, and closed. The MAP contains keys and values

Returns:

true if no errors appears, else false

Definition at line 58 of file configurationfile.cpp.

References config, file, getValue(), and Tools::stringToVector().

Referenced by BotKernel::BotKernel(), and loadconffile().

6.8.3.8 void ConfigurationFile::setValue (string *key*, string *value*)

Set a value to a key.

Set a value to a given key. If the key exists, the value is replaced. If the key does'nt exists, it's added to the map

Postcondition:

The value associated to the given key i saved in the configuration. To save it in the file, use flush() method

Parameters:

key The key for with you want to set a value

value The value that you want to give to the given key

Definition at line 168 of file configurationfile.cpp.

References config.

Referenced by autoop(), autovoice(), protectmodes(), protecttopic(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), unautoop(), unautovoice(), unprotectmodes(), and unprotecttopic().

6.8.4 Member Data Documentation

6.8.4.1 map<string,string> ConfigurationFile::config `[private]`

Configuration container.

Definition at line 69 of file configurationfile.h.

Referenced by ConfigurationFile(), delKey(), flush(), getConfig(), getValue(), load(), and setValue().

6.8.4.2 string ConfigurationFile::file `[private]`

Configuration file path (and name).

Definition at line 71 of file configurationfile.h.

Referenced by ConfigurationFile(), flush(), getFilePath(), and load().

6.8.4.3 vector<string> ConfigurationFile::protectedKeys `[private]`

Stores protected keys.

Definition at line 73 of file configurationfile.h.

Referenced by addProtectedKey(), ConfigurationFile(), and getValue().

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

- src/configurationfile.h
- src/configurationfile.cpp

6.9 CountDownFunction Struct Reference

Countdown information storage.

```
#include <botkernel.h>
```

Public Attributes

- StructFunctionStorage function
- Message msg
- time_t timestamp
- unsigned int count

6.9.1 Detailed Description

Countdown information storage.

Definition at line 63 of file botkernel.h.

6.9.2 Member Data Documentation

6.9.2.1 unsigned int CountDownFunction::count

Definition at line 67 of file botkernel.h.

Referenced by BotKernel::addCountDown().

6.9.2.2 StructFunctionStorage CountDownFunction::function

Definition at line 64 of file botkernel.h.

Referenced by BotKernel::addCountDown().

6.9.2.3 Message CountDownFunction::msg

Definition at line 65 of file botkernel.h.

Referenced by BotKernel::addCountDown().

6.9.2.4 time_t CountDownFunction::timestamp

Definition at line 66 of file botkernel.h.

Referenced by BotKernel::addCountDown().

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

- src/botkernel.h

6.10 CPPThread Class Reference

pthread C++ wrapper

`#include <cppthread.h>`

Public Member Functions

- CPPThread ()

  Constructor.

- ∼CPPThread ()

  Destructor.

- bool exec (threadProcess, void ∗)

  Prepare and launch a thread.

- bool terminate ()

  Terminate (cancel) the thread.

- bool isRunning ()

  Check if the thread is running.

- bool isFinished ()

  Check if the thread is finished.

- void ∗ join ()

  Join thread.

- pthread_t ∗ getHandle ()

  Get thread's handle.

Static Private Member Functions

- static void ∗ threadStartup (void ∗)

  threaded function

Private Attributes

- pthread_t ∗ handle

  pthread handle

- threadInfos ti

  Threads informations.

6.10.1 Detailed Description

pthread C++ wrapper

This class stores pthread management function to use a pthread as an object

Definition at line 48 of file cppthread.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 CPPThread::CPPThread ()

Constructor.

The class constructor. Initialize private attributes

Postcondition:

An object is contructed

Definition at line 37 of file cppthread.cpp.

References threadInfos::finished, handle, threadInfos::running, and ti.

6.10.2.2 CPPThread::∼CPPThread ()

Destructor.

The class destructor

Definition at line 46 of file cppthread.cpp.

References handle, and terminate().

6.10.3 Member Function Documentation

6.10.3.1 bool CPPThread::exec (threadProcess *myThreadProcess*, void ∗ *args*)

Prepare and launch a thread.

Prepare and launch a thread. Stores a function to execute, and arguments, then create a thread, and launch the threadStartup

Parameters:

myThreadProcess Function to execute

args Arguments to send to the threaded function

Returns:

true if the thread has been launched, else false

Definition at line 60 of file cppthread.cpp.

References threadInfos::args, handle, isRunning(), threadInfos::process, threadStartup(), and ti.

Referenced by BotKernel::executeFunction(), and RemoteControl::RemoteControl().

6.10.3.2 pthread_t * CPPThread::getHandle ()

Get thread's handle.

Return pthread handle. Use this handle to use pthread functions not provided by this class

Returns:

Handle's pointer

Definition at line 137 of file cppthread.cpp.

References handle.

6.10.3.3 bool CPPThread::isFinished ()

Check if the thread is finished.

Tell if the thread is finished. The thread is concidered has finished if the exectuion has finished itself (not canceled)

Returns:

True if the thread is finished, else false

Definition at line 118 of file cppthread.cpp.

References threadInfos::finished, and ti.

6.10.3.4 bool CPPThread::isRunning ()

Check if the thread is running.

Give running state

Returns:

True if the thread is running, else false

Definition at line 108 of file cppthread.cpp.

References threadInfos::running, and ti.

Referenced by exec(), and terminate().

6.10.3.5 void * CPPThread::join ()

Join thread.

Wait for thread end, and then free memory

Definition at line 125 of file cppthread.cpp.

References handle.

Referenced by BotKernel::executeFunction().

6.10.3.6 bool CPPThread::terminate ()

Terminate (cancel) the thread.

Cancel the thread

Returns:

true if the thread has been canceled, else false

Definition at line 93 of file cppthread.cpp.

References handle, isRunning(), threadInfos::running, and ti.

Referenced by BotKernel::executeFunction(), and ∼CPPThread().

6.10.3.7 void ∗ CPPThread::threadStartup (void ∗ *arg*) `[static, private]`

threaded function

Threaded function. In this function, the thread is prepared, then the function to execute is launched.

Parameters:

arg thread information structure pointer

Postcondition:

running and finished flags are updated

Definition at line 78 of file cppthread.cpp.

References threadInfos::args, threadInfos::finished, threadInfos::process, threadInfos::running, and ti.

Referenced by exec().

6.10.4 Member Data Documentation

6.10.4.1 pthread_t∗ CPPThread::handle `[private]`

pthread handle

Definition at line 52 of file cppthread.h.

Referenced by CPPThread(), exec(), getHandle(), join(), terminate(), and ∼CPPThread().

6.10.4.2 threadInfos CPPThread::ti `[private]`

Threads informations.

Definition at line 56 of file cppthread.h.

Referenced by CPPThread(), exec(), isFinished(), isRunning(), terminate(), and threadStartup().

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

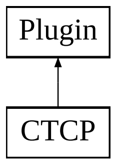
- src/cppthread.h
- src/cppthread.cpp

6.11 CTCP Class Reference

Provide CTCP Answers.

`#include <ctcp.h>`

Inheritance diagram for CTCP::



Public Member Functions

- CTCP (BotKernel $*$)

  Constructor.

6.11.1 Detailed Description

Provide CTCP Answers.

Provide CTCP Answers

Definition at line 41 of file ctcp.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 CTCP::CTCP (BotKernel $*$ *b*)

Constructor.

Constructor

Definition at line 34 of file ctcp.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_FREE_COMMAND_-HANDLER, Plugin::name, and Plugin::version.

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

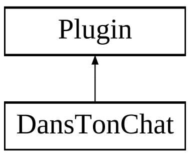
- src/plugins/ctcp.h
- src/plugins/ctcp.cpp

6.12 DansTonChat Class Reference

Display quotes from danstonchat.com.

```
#include <danstonchat.h>
```

Inheritance diagram for DansTonChat::



Public Member Functions

- DansTonChat (BotKernel ∗)

  Constructor.

6.12.1 Detailed Description

Display quotes from danstonchat.com.

Display quotes from danstonchat.com

Definition at line 41 of file danstonchat.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 DansTonChat::DansTonChat (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file danstonchat.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

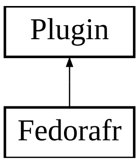
- src/plugins/danstonchat.h
- src/plugins/danstonchat.cpp

6.13 Fedorafr Class Reference

Class that provides stuff to search on Fedora-fr.org wiki or planet.

`#include <fedorafr.h>`

Inheritance diagram for Fedorafr::



Public Member Functions

- Fedorafr (BotKernel ∗)

  Constructor.

- vector< string > getWikiLinks (string)

  Extract links in a wiki result page.

6.13.1 Detailed Description

Class that provides stuff to search on Fedora-fr.org wiki or planet.

Class that provides stuff to search on Fedora-fr.org wiki or planet. It connects the bot to Fedora-fr.org website, execute queries and parse result

Definition at line 42 of file fedorafr.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Fedorafr::Fedorafr (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file fedorafr.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

6.13.3 Member Function Documentation

6.13.3.1 vector< string > Fedorafr::getWikiLinks (string *datas*)

Extract links in a wiki result page.

Extract links in a wiki result page

Parameters:

datas HTML code from wiki page

Returns:

Wiki's URLs

Definition at line 52 of file fedorafr.cpp.

References Tools::stringToVector().

Referenced by wiki().

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

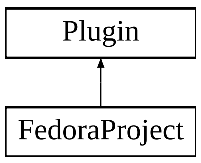
- src/plugins/fedorafr.h
- src/plugins/fedorafr.cpp

6.14 FedoraProject Class Reference

Plugin in connection with fedora project.

`#include <fedoraproject.h>`

Inheritance diagram for FedoraProject::



Public Member Functions

- FedoraProject (BotKernel ∗)

  Constructor.

- string whoowns (string)

  Tells who owns a package.

- bool loadFasFile (string)

  Load FAS file.

- vector< string > getFasUserInfos (string)

  get FAS user infos

Static Public Member Functions

- static int writer (char ∗, size_t, size_t, string ∗)

  writer call back function used by curl

Private Attributes

- map< string, vector< string > > usersInfos

  FAS users infos.

6.14.1 Detailed Description

Plugin in connection with fedora project.

Plugin that provides tools to retrieve informations about fedora project

Definition at line 45 of file fedoraproject.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 FedoraProject::FedoraProject (BotKernel $*$ *b*)

Constructor.

Constructor

Definition at line 34 of file fedoraproject.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, BotKernel::getDatasDir(), BotKernel::getSysLog(), IN_COMMAND_HANDLER, loadFasFile(), LogFile::log(), Plugin::name, Plugin::version, and WARNING.

6.14.3 Member Function Documentation

6.14.3.1 vector< string > FedoraProject::getFasUserInfos (string *nick*)

get FAS user infos

Get FAS user infos Infos are (see `https://admin.fedoraproject.org/accounts/group/dump/cla_-done`) :

- mail
- real name
- status

Returns:

FAS user infos

Definition at line 127 of file fedoraproject.cpp.

References usersInfos.

Referenced by fas().

6.14.3.2 bool FedoraProject::loadFasFile (string *file*)

Load FAS file.

Load FAS file that contains FAS users informations

Returns:

true is load is OK, else false

Definition at line 97 of file fedoraproject.cpp.

References Tools::stringToVector(), and usersInfos.

Referenced by FedoraProject(), and reloadfas().

6.14.3.3 string FedoraProject::whoowns (string *name*)

Tells who owns a package.

Tells who owns a package using `https://admin.fedoraproject.org/pkgdb/packages/name/` webpage

Parameters:

name Package name

Returns:

Package's owner

Definition at line 55 of file fedoraproject.cpp.

References Tools::urlencode(), and writer().

Referenced by whoowns().

6.14.3.4 int FedoraProject::writer (char ∗ *data*, size_t *size*, size_t *nmemb*, string ∗ *buffer*) `[static]`

writer call back function used by curl

Definition at line 139 of file fedoraproject.cpp.

Referenced by whoowns().

6.14.4 Member Data Documentation

6.14.4.1 map<string,vector<string> > FedoraProject::usersInfos `[private]`

FAS users infos.

Definition at line 49 of file fedoraproject.h.

Referenced by getFasUserInfos(), and loadFasFile().

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

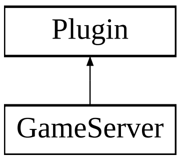
- src/plugins/fedoraproject.h
- src/plugins/fedoraproject.cpp

6.15 GameServer Class Reference

Provides tools to query game servers.

`#include <gameserver.h>`

Inheritance diagram for GameServer::



Public Member Functions

- GameServer (BotKernel ∗)

  Constructor.

- long strToLong (string)

  Convert a string to a long (seems fucked).

- string getHLstring (unsigned int ∗, char ∗)

  Get a string in HL1 protocol.

- char getHLbyte (unsigned int ∗, char ∗)

  Get a byte in HL1 protocol.

- string getHLlong (unsigned int ∗, char ∗)

  Get a long in HL1 protocol (as a string).

- bool getHL1Players (vector< string > ∗, char ∗)

  Get players list in HL1 protocol.

- bool getHL1Challenge (string ∗, char ∗)

  Get HL1 challenge.

- bool getHL1Infos (map< string, string > ∗, char ∗)

  Get HL1 server's settings.

- string getQ3GameType (string)

  Get a Q3 gametype according to a number.

- bool parseWSWinfos (map< string, string > ∗, vector< string > ∗, char ∗)

  Get warsow server's infos (settings and players).

- bool parseQ3infos (map< string, string > ∗, vector< string > ∗, char ∗)

  Get Q3 server's infos (settings and players).

- bool sendQuery (string, string, int ∗, string)

Send a query to a server.

- string getResult (int, char ∗)

  Get a result from a server.

6.15.1 Detailed Description

Provides tools to query game servers.

Provides tools to query game servers

Definition at line 44 of file gameserver.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 GameServer::GameServer (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file gameserver.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

6.15.3 Member Function Documentation

6.15.3.1 bool GameServer::getHL1Challenge (string ∗ *challenge*, char ∗ *datas*)

Get HL1 challenge.

Get challenge in HL1 protocol

Parameters:

challenge string that will contain the challenge

datas Datas to read

Postcondition:

Challenge is updated

Returns:

True if protocol OK, else false

Definition at line 141 of file gameserver.cpp.

References getHLbyte(), and getHLlong().

Referenced by hl().

6.15.3.2 bool GameServer::getHL1Infos (map< string, string > ∗ *settings*, char ∗ *datas*)

Get HL1 server's settings.

Get HL server's info in HL1 protocol

Parameters:

settings Map that will contain settings
datas Datas to read

Postcondition:

Settings are set

Returns:

True if protocol OK, else false

Definition at line 156 of file gameserver.cpp.

References getHLbyte(), getHLlong(), and getHLstring().

Referenced by hl().

6.15.3.3 bool GameServer::getHL1Players (vector< string > ∗ *players*, char ∗ *datas*)

Get players list in HL1 protocol.

Get players list in HL1 protocol

Parameters:

players players list to fill
datas Datas to read

Postcondition:

Players list is updated

Returns:

True if protocol OK, else false

Definition at line 120 of file gameserver.cpp.

References getHLbyte(), and getHLstring().

Referenced by hl().

6.15.3.4 char GameServer::getHLbyte (unsigned int ∗ *index*, char ∗ *datas*)

Get a byte in HL1 protocol.

Get a byte in HL1 protocol Read only one byte

Parameters:

index Index to read datas

datas Datas to read

Postcondition:

The index is updated

Returns:

The byte

Definition at line 92 of file gameserver.cpp.

Referenced by getHL1Challenge(), getHL1Infos(), and getHL1Players().

6.15.3.5 string GameServer::getHLlong (unsigned int $*$ *index*, char $*$ *datas*)

Get a long in HL1 protocol (as a string).

Get a long in HL1 protocol

Parameters:

index Index to read datas
datas Datas to read

Postcondition:

The index is updated

Returns:

The long data

Definition at line 104 of file gameserver.cpp.

Referenced by getHL1Challenge(), and getHL1Infos().

6.15.3.6 string GameServer::getHLstring (unsigned int $*$ *index*, char $*$ *datas*)

Get a string in HL1 protocol.

Get a string in HL1 protocol Read datas until a 0x00 char is found

Parameters:

index Index to read datas
datas Datas to read

Postcondition:

The index is updated

Returns:

The string

Definition at line 73 of file gameserver.cpp.

Referenced by getHL1Infos(), and getHL1Players().

6.15.3.7 string GameServer::getQ3GameType (string *number*)

Get a Q3 gametype according to a number.

Get a Q3 gametype according to a number

Parameters:

number gametype number

Returns:

Gametype string

Definition at line 192 of file gameserver.cpp.

Referenced by q3().

6.15.3.8 string GameServer::getResult (int *sock*, char $*$ *buffer*)

Get a result from a server.

Get a UDP result from a server

Precondition:

A query has been sended

Parameters:

sock Socket used for communication

buffer Buffer that will contain result (answer)

Returns:

A string containing "0" is everything is OK, else contain the error that occured

Definition at line 301 of file gameserver.cpp.

References MAX_CHARS.

Referenced by hl(), q3(), and warsow().

6.15.3.9 bool GameServer::parseQ3infos (map< string, string > $*$ *settings*, vector< string > $*$ *players*, char $*$ *datas*)

Get Q3 server's infos (settings and players).

Parse datas to extract settings and players from a Q3 server

Parameters:

settings map that will contain settings

players vector that will contain players

datas Datas to read

Postcondition:

players and settings are set

Returns:

True if protocol OK, else false

Definition at line 248 of file gameserver.cpp.

References Tools::parseQ3Colors(), and Tools::stringToVector().

Referenced by q3().

6.15.3.10 bool GameServer::parseWSWinfos (map< string, string > * *settings*, vector< string > * *players*, char * *datas*)

Get warsow server's infos (settings and players).

Parse datas to extract settings and players from a warsow server

Parameters:

settings map that will contain settings

players vector that will contain players

datas Datas to read

Postcondition:

players and settings are set

Returns:

True if protocol OK, else false

Definition at line 220 of file gameserver.cpp.

References Tools::parseQ3Colors(), and Tools::stringToVector().

Referenced by warsow().

6.15.3.11 bool GameServer::sendQuery (string *ip*, string *port*, int * *sock*, string *query*)

Send a query to a server.

Send a UDP query to a server

Parameters:

ip Server' ip

port Server's posrt

sock Pointer to a socket

query Query to send

Postcondition:

The socket is opened

Returns:

True if send OK, else false

Definition at line 274 of file gameserver.cpp.

References Tools::strToInt().

Referenced by hl(), q3(), and warsow().

6.15.3.12 long GameServer::strToLong (string *str*)

Convert a string to a long (seems fucked).

Convert a string to a long (seems fucked) /!\ DOESN'T WORK /!\

Parameters:

str String to convert

Returns:

Number corresponding to the string

Definition at line 53 of file gameserver.cpp.

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

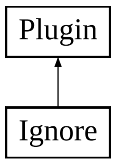
- src/plugins/gameserver.h
- src/plugins/gameserver.cpp

6.16 Ignore Class Reference

Manage ignores.

`#include <ignore.h>`

Inheritance diagram for Ignore::



Public Member Functions

- Ignore (BotKernel ∗)

  Constructor.

- void addIgnore (string, string, unsigned int)

  Add a host to the ignore list.

- bool delIgnore (unsigned int)

  Remove a host from ignore list.

- bool isIgnored (string)

  Tell if a host is ignored.

- vector< string > getIgnoreList ()

  Return the ignore list.

- void purifyList ()

  Clear the XML file from out dated ignores.

Private Member Functions

- void initFile ()

  Initialize the XML file.

Private Attributes

- TiXmlDocument ∗ doc

  Represent the xml document.

- TiXmlNode ∗ root

  Represent documents's root.

6.16.1 Detailed Description

Manage ignores.

This class provides an ignore system that allow the bot to ignore users

Definition at line 51 of file ignore.h.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Ignore::Ignore (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file ignore.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, doc, BotKernel::getDatasDir(), IN_BEFORE_TREATMENT, IN_COMMAND_HANDLER, IN_LOOP, initFile(), Plugin::name, root, and Plugin::version.

6.16.3 Member Function Documentation

6.16.3.1 void Ignore::addIgnore (string *mask*, string *by*, unsigned int *duration*)

Add a host to the ignore list.

Add a host to the ignore list

Parameters:

mask mask to ignore

by User mask that add the ignore

duration Ignore duration (in seconsd)

Definition at line 76 of file ignore.cpp.

References doc, root, and Tools::to_lower().

Referenced by addIgnore().

6.16.3.2 bool Ignore::delIgnore (unsigned int *index*)

Remove a host from ignore list.

Del a host from the ignore list

Parameters:

index Ignore index

Definition at line 96 of file ignore.cpp.

References doc.

Referenced by delIgnore().

6.16.3.3 vector< string > Ignore::getIgnoreList ()

Return the ignore list.

Give the ignore list

Returns:

A vector containing ignored hosts

Definition at line 136 of file ignore.cpp.

References Tools::intToStr(), root, and Tools::strToInt().

Referenced by ignoreList().

6.16.3.4 void Ignore::initFile () `[private]`

Initialize the XML file.

Initilaize the XML file by creating root and first childs (file empty structure)

Definition at line 62 of file ignore.cpp.

References doc, and root.

Referenced by Ignore().

6.16.3.5 bool Ignore::isIgnored (string *host*)

Tell if a host is ignored.

Check if a host is ignored

Parameters:

host Host to check

Returns:

true if ignored, else false

Definition at line 117 of file ignore.cpp.

References Tools::ircMaskMatch(), root, and Tools::to_lower().

Referenced by isIgnored(), and testIgnoredUser().

6.16.3.6 void Ignore::purifyList ()

Clear the XML file from out dated ignores.

Clear ignore list from outdated ignores

Definition at line 164 of file ignore.cpp.

References doc, root, and Tools::strToInt().

Referenced by purifyList().

6.16.4 Member Data Documentation

6.16.4.1 TiXmlDocument∗ Ignore::doc `[private]`

Represent the xml document.

Definition at line 55 of file ignore.h.

Referenced by addIgnore(), delIgnore(), Ignore(), initFile(), and purifyList().

6.16.4.2 TiXmlNode∗ Ignore::root `[private]`

Represent documents's root.

Definition at line 57 of file ignore.h.

Referenced by addIgnore(), getIgnoreList(), Ignore(), initFile(), isIgnored(), and purifyList().

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

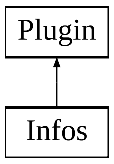
- src/plugins/ignore.h
- src/plugins/ignore.cpp

6.17 Infos Class Reference

Give infos about kernel.

`#include <infos.h>`

Inheritance diagram for Infos::



Public Member Functions

- Infos (BotKernel ∗)

  Constructor.

6.17.1 Detailed Description

Give infos about kernel.

Give infos about the bot

Definition at line 41 of file infos.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Infos::Infos (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file infos.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, IN_FREE_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

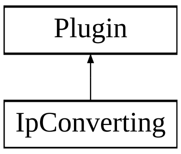
- src/plugins/infos.h
- src/plugins/infos.cpp

6.18 IpConverting Class Reference

Tools for IP converting.

```
#include <ipconverting.h>
```

Inheritance diagram for IpConverting::



Public Member Functions

- IpConverting (BotKernel ∗)

  Constructor.

6.18.1 Detailed Description

Tools for IP converting.

This class provides commands to convert an IP to a Host and a host to all its corresponding IPs

Definition at line 42 of file ipconverting.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 IpConverting::IpConverting (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file ipconverting.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

- src/plugins/ipconverting.h
- src/plugins/ipconverting.cpp

6.19 IRCProtocol Class Reference

Class that convert messages to IRC messages.

`#include <ircprotocol.h>`

Public Member Functions

- IRCProtocol ()

    Constructor.

- ~IRCProtocol ()

    Destructor.

Static Public Member Functions

- static vector< string > identify (string, string, string, string)

    Construct a string to identify to an irc server.

- static string quitServer (string raison="autokilled!")

    Construct a string to quit an irc server.

- static string joinChannel (string)

    Construct a string to join a channel.

- static string leaveChannel (string, string raison="\\o_")

    Construct a string to leave a channel.

- static string changeNick (string)

    Construct a string to change nick.

- static string ping (string)

    Construct a string to ping the server.

- static string pong (string)

    Construct a string to respond to a ping.

- static string sendMsg (string, string)

    Construct a string to send a message.

- static vector< string > sendMsg (string, vector< string >)

    Construct a vector with strings to send several messages.

- static string sendAction (string, string)

    Construct a string to send an action on a channel.

- static string changeTopic (string, string)

    Construct a string to change a channel topic.

- static vector< string > applyModes (string, vector< string >, char, char, unsigned int)

  Construct strings to apply a mode on different people.
- static vector< string > op (vector< string >, string)

  Construct strings to op people on a channel.
- static string op (string, string)

  Construct a string to op a user on a channel.
- static vector< string > unop (vector< string >, string)

  Construct strings to unop people on a channel.
- static string unop (string, string)

  Construct a string to unop a user on a channel.
- static vector< string > voice (vector< string >, string)

  Construct strings to voice people on a channel.
- static string voice (string, string)

  Construct a string to voice a user on a channel.
- static vector< string > unvoice (vector< string >, string)

  Construct strings to unvoice people on a channel.
- static string unvoice (string, string)

  Construct a string to unvoice a user on a channel.
- static string ban (string, string)

  Construct a string to ban a host on a channel.
- static string unban (string, string)

  Construct a string to unban a host on a channel.
- static string sendNotice (string, string)

  Construct a string to send a notice.
- static vector< string > sendNotices (string, vector< string >)

  Construct strings to send notices.
- static string kick (string, string, string)

  Construct a string to kick someone from channel.
- static string who (string, string)

  Construct a string for a WHO command.
- static string invite (string, string)

  Construct a string for a INVITE command.

6.19.1 Detailed Description

Class that convert messages to IRC messages.

This class convert "humain" strings to IRC messages This class allow the bot to connect to different servers type simply by changing its code methods All methods a static, no object needed

Definition at line 44 of file ircprotocol.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 IRCProtocol::IRCProtocol ()

Constructor.

Class contructor

Definition at line 34 of file ircprotocol.cpp.

6.19.2.2 IRCProtocol::~IRCProtocol ()

Destructor.

Class destructor

Definition at line 41 of file ircprotocol.cpp.

6.19.3 Member Function Documentation

6.19.3.1 vector< string > IRCProtocol::applyModes (string *channel*, vector< string > *users_list*, char *sign*, char *mode*, unsigned int *limit*) `[static]`

Construct strings to apply a mode on different people.

Construct strings to apply a mode on different people

Parameters:

channel Channel where apply modes

users_list Users on witch apply modes

mode Mode ti apply

sign Mode sign

limit Limit for size

Returns:

Strings for modes

Definition at line 181 of file ircprotocol.cpp.

Referenced by unbanall().

6.19.3.2 string IRCProtocol::ban (string *mask*, string *channel*) `[static]`

Construct a string to ban a host on a channel.

Format a string to ban a mask on a channel

Parameters:

mask mask to ban

channel ban's channel

Returns:

ban string

Definition at line 302 of file ircprotocol.cpp.

Referenced by ban(), banmask(), and joinHandler().

6.19.3.3 string IRCProtocol::changeNick (string *nick*) `[static]`

Construct a string to change nick.

Format a string to change the bot nickname

Parameters:

nick New nick

Returns:

String containing the message to send to change nick

Definition at line 99 of file ircprotocol.cpp.

Referenced by getMyFirstNick(), identify(), secondaryNick(), and setNick().

6.19.3.4 string IRCProtocol::changeTopic (string *channel*, string *topic*) `[static]`

Construct a string to change a channel topic.

Format a string to change a topic

Parameters:

channel Channel witch topic will change

topic New topic

Returns:

String containing the message to send to change a channel topic

Definition at line 167 of file ircprotocol.cpp.

Referenced by topic(), and topicHandler().

6.19.3.5 vector< string > IRCProtocol::identify (string *pass*, string *ident*, string *name*, string *nick*) `[static]`

Construct a string to identify to an irc server.

Format strings to identify to an IRC server according to parameters

Parameters:

pass IRC server password (empty string if not required)
ident Connection Ident
name Connection Name
nick Connection Nick

Returns:

Vector containing strings to send to the server for authentification

Definition at line 53 of file ircprotocol.cpp.

References changeNick().

Referenced by BotKernel::connect().

6.19.3.6 string IRCProtocol::invite (string *channel*, string *nick*) `[static]`

Construct a string for a INVITE command.

Format a string to send a INVITE command

Parameters:

channel Channel where invite the user
nick Nick to invite

Returns:

String containing the invite command

Definition at line 465 of file ircprotocol.cpp.

Referenced by invite().

6.19.3.7 string IRCProtocol::joinChannel (string *channel*) `[static]`

Construct a string to join a channel.

Format a string to join a channel

Parameters:

channel Channel to join

Returns:

String containing the message to send to join the channel

Definition at line 78 of file ircprotocol.cpp.

Referenced by cycleChannel(), joinChannel(), kickHandler(), onEndOfMOTD(), onInvite(), partHandler(), quitHandler(), and rejoinChan().

6.19.3.8 string IRCProtocol::kick (string *nick*, string *chan*, string *reason*) `[static]`

Construct a string to kick someone from channel.

Format a string to kick a user from a channel

Parameters:

nick User nick to kick

chan Channel from witch one the user is kicked

reason Kick reason, can be empty

Returns:

String containing the message to kick the user

Definition at line 454 of file ircprotocol.cpp.

Referenced by ban(), banmask(), joinHandler(), kick(), kickall(), kickHandler(), masskick(), modeHandler(), and randomKick().

6.19.3.9 string IRCProtocol::leaveChannel (string *channel*, string *reason* = `"\\o_"`) `[static]`

Construct a string to leave a channel.

Format a string to leave a channel

Parameters:

channel Channel to leave

reason Leave msg. Can be empty

Returns:

String containing the message to send to leave the channel

Definition at line 89 of file ircprotocol.cpp.

Referenced by cycleChannel(), kickHandler(), leaveChannel(), partHandler(), and quitHandler().

6.19.3.10 string IRCProtocol::op (string *nick*, string *channel*) `[static]`

Construct a string to op a user on a channel.

Construct a string to op a user on a channel Obsolete : use applyModes instead

Parameters:

nick Nick to op

channel Channel where the nick must be opped

Returns:

OP string

Definition at line 246 of file ircprotocol.cpp.

6.19.3.11 vector< string > IRCProtocol::op (vector< string > *vectorNicks*, string *channel*) `[static]`

Construct strings to op people on a channel.

Format strings to op users on a channel Obsolete : use applyModes instead

Parameters:

vectorNicks Vector containing string containing nicks to op

channel Channel where nicks must be opped

Returns:

Strings containing instructions to op users

Definition at line 212 of file ircprotocol.cpp.

Referenced by joinHandler(), op(), and opall().

6.19.3.12 string IRCProtocol::ping (string *ping*) `[static]`

Construct a string to ping the server.

Format a string to ping the server

Parameters:

ping String that the server must reply to validate the ping

Returns:

String containing the message to send to ping the server

Definition at line 109 of file ircprotocol.cpp.

Referenced by checkConnection().

6.19.3.13 string IRCProtocol::pong (string *pong*) `[static]`

Construct a string to respond to a ping.

Format a string to pong the server

Parameters:

pong string ping answer

Returns:

String containing the message to send to pong the server

Definition at line 119 of file ircprotocol.cpp.

Referenced by pinged().

6.19.3.14 string IRCProtocol::quitServer (string *reason* = `"autokilled!"`) `[static]`

Construct a string to quit an irc server.

Format a string to quit an IRC server

Parameters:

reason Quit message. Can be empty

Returns:

String containing the message to send to quit the IRC server

Definition at line 68 of file ircprotocol.cpp.

Referenced by disconnect().

6.19.3.15 string IRCProtocol::sendAction (string *channel*, string *action*) `[static]`

Construct a string to send an action on a channel.

Send an action on a channel. An action is that kind of message : $*$ trustyrc slaps toto

Parameters:

channel	Channel where to send the action
action	Action to send

Returns:

A string containing the formatted message

Definition at line 156 of file ircprotocol.cpp.

Referenced by slapUser().

6.19.3.16 vector< string > IRCProtocol::sendMsg (string *destination*, vector< string > *messages*) `[static]`

Construct a vector with strings to send several messages.

Format a strings to send a messages to a channel or to a user

Parameters:

destination	Messages receiver (channel or user(pv query))
messages	Messages to send

Returns:

Vector with strings containing the messages to send to the server to send messages on a channel or to a user

Definition at line 141 of file ircprotocol.cpp.

6.19.3.17 string IRCProtocol::sendMsg (string *destination*, string *message*) [static]

Construct a string to send a message.

Format a string to send a message to a channel or to a user

Parameters:

destination Message receiver (channel or user(pv query))

message Message to send

Returns:

String containing the message to send to the server to send a message on a channel or to a user

Definition at line 130 of file ircprotocol.cpp.

Referenced by autoop(), autovoice(), ball(), bug(), bzsearch(), checkBug(), danstonchat(), displayAdvertise(), displayPaste(), endSurvey(), fas(), greplog(), hl(), host2ip(), ip2host(), lamoule(), lastQuote(), lastseen(), launchSurvey(), myFunction(), planet(), player(), protectmodes(), protecttopic(), q3(), quote(), randomKick(), searchQuote(), stopSurvey(), tele(), tell(), top5(), topshot(), toptotal(), trad(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), warsow(), whoowns(), and wiki().

6.19.3.18 string IRCProtocol::sendNotice (string *destination*, string *notice*) [static]

Construct a string to send a notice.

Format a string to send a notice

Parameters:

destination Notice receiver (nick or channel)

notice Notice message

Returns:

String containing the message to send the notice

Definition at line 414 of file ircprotocol.cpp.

Referenced by addad(), addIgnore(), addOnlyon(), addQuote(), addsuperadmin(), addtempsuperadmin(), adinfos(), chanlev(), clearCountDowns(), ctcp_ping(), ctcp_version(), delad(), deletekey(), deleteplayer(), delIgnore(), delOnlyon(), delQuote(), delsuperadmin(), disable(), enable(), BotKernel::executeFunction(), flushconffile(), getconfvalue(), getnbcountdowns(), help(), increase(), isIgnored(), lamoule(), launchSurvey(), load(), loadconffile(), loadnocheck(), moduleinfos(), nextscore(), notice(), online(), prefix(), quoteInfos(), reloadfas(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setSuperAdminPass(), slapme(), stopSurvey(), sysinfos(), unload(), unloadnocheck(), uptime(), version(), and vote().

6.19.3.19 vector< string > IRCProtocol::sendNotices (string *destination*, vector< string > *notices*) [static]

Construct strings to send notices.

Format strings to send notices

Parameters:

destination Notice receiver (nick or channel)

notices Notices messages

Returns:

Strings containing the messages to send the notices

Definition at line 425 of file ircprotocol.cpp.

Referenced by baninfos(), banlist(), chanlev(), commandsStatus(), ignoreList(), listads(), listlibs(), listmodules(), superadminlist(), and whoami().

6.19.3.20 string IRCProtocol::unban (string *mask*, string *channel*) `[static]`

Construct a string to unban a host on a channel.

Format a string to unban a mask on a channel

Parameters:

mask Mask to unban

channel Channel where to unban the user

Returns:

Unban string

Definition at line 313 of file ircprotocol.cpp.

Referenced by bandel(), and Moderation::clearOutBans().

6.19.3.21 string IRCProtocol::unop (string *nick*, string *channel*) `[static]`

Construct a string to unop a user on a channel.

Construct a string to unop a user on a channel

Parameters:

nick Nick to unop

channel Channel where the nick must be unopped

Returns:

UNOP string

Definition at line 291 of file ircprotocol.cpp.

6.19.3.22 vector< string > IRCProtocol::unop (vector< string > *vectorNicks*, string *channel*) `[static]`

Construct strings to unop people on a channel.

Format strings to unop users on a channel Obsolete : use applyModes instead

Parameters:

vectorNicks Vector containing string containing nicks to unop

channel Channel where nicks must be unopped

Returns:

Strings containing instructions to unop users

Definition at line 258 of file ircprotocol.cpp.

Referenced by unop(), and unopall().

6.19.3.23 string IRCProtocol::unvoice (string *nick*, string *channel*) `[static]`

Construct a string to unvoice a user on a channel.

Construct a string to unvoice a user on a channel

Parameters:

nick Nick to unvoice

channel Channel where the nick must be unvoiced

Returns:

voice string

Definition at line 403 of file ircprotocol.cpp.

6.19.3.24 vector< string > IRCProtocol::unvoice (vector< string > *vectorNicks*, string *channel*) `[static]`

Construct strings to unvoice people on a channel.

Format strings to unvoice users on a channel Obsolete : use applyModes instead

Parameters:

vectorNicks Vector containing string containing nicks to unvoice

channel Channel where nicks must be unvoiced

Returns:

Strings containing instructions to unvoice users

Definition at line 370 of file ircprotocol.cpp.

Referenced by unvoice(), and unvoiceall().

6.19.3.25 string IRCProtocol::voice (string *nick*, string *channel*) `[static]`

Construct a string to voice a user on a channel.

Construct a string to voice a user on a channel

Parameters:

nick Nick to voice

channel Channel where the nick must be voiced

Returns:

voice string

Definition at line 358 of file ircprotocol.cpp.

6.19.3.26 vector< string > IRCProtocol::voice (vector< string > *vectorNicks*, string *channel*) `[static]`

Construct strings to voice people on a channel.

Format strings to voice users on a channel Obsolete : use applyModes instead

Parameters:

vectorNicks Vector containing string containing nicks to voice

channel Channel where nicks must be voiced

Returns:

Strings containing instructions to voice users

Definition at line 325 of file ircprotocol.cpp.

Referenced by joinHandler(), voice(), and voiceall().

6.19.3.27 string IRCProtocol::who (string *channel*, string *params*) `[static]`

Construct a string for a WHO command.

Format a string to send a WHO command

Parameters:

channel Channel for the WHO command

params Parameters for WHO command

Returns:

The WHO command

Definition at line 439 of file ircprotocol.cpp.

Referenced by onJoin(), and reloadUsers().

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

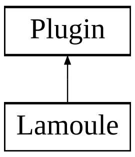
- src/ircprotocol.h
- src/ircprotocol.cpp

6.20 Lamoule Class Reference

Manage lamoule's ladder.

```
#include <lamoule.h>
```

Inheritance diagram for Lamoule::



Public Member Functions

- Lamoule (BotKernel ∗)
  Constructor.
- vector< string > getTopShot ()
  Get the lamoule's topshot.
- void setTopShot (string, string, string)
  Set the lamoule's topshot.
- void addPlayer (string, unsigned int)
  Add a player in the ladder.
- char increaseScore (string, int, unsigned int, bool)
  Increase score for a player.
- vector< TiXmlElement ∗ > sort (sort_criterion, int)
  Sort elements to get stats.
- vector< string > get5first (sort_criterion, int)
  Get 5 first.
- vector< string > getInfosPlayer (string, int)
  Get informations about a player.
- bool deletePlayer (string)
  Remove a player from the ladder.
- void purifyFile (int)
  Purify File.
- bool setNextScore (int)
  Set next score.
- int generateScore ()
  generate score

Private Member Functions

- void initFile ()

  Initialize the XML file.

Private Attributes

- TiXmlDocument ∗ doc

  Represent the xml document.

- TiXmlNode ∗ root

  Represent documents's root.

- unsigned int nextScore

  Stores the next score.

- int MAX_SCORE

  Maximul score.

- int FIRST_FLOOR

  First floor.

- int SECOND_FLOOR

  Second floor.

6.20.1 Detailed Description

Manage lamoule's ladder.

Manage lamoule's ladder. Score are stored in an XML file

Definition at line 57 of file lamoule.h.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Lamoule::Lamoule (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file lamoule.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, doc, FIRST_FLOOR, BotKernel::getDatasDir(), IN_COMMAND_HANDLER, IN_LOOP, initFile(), MAX_-SCORE, Plugin::name, nextScore, root, SECOND_FLOOR, and Plugin::version.

6.20.3 Member Function Documentation

6.20.3.1 void Lamoule::addPlayer (string *nick*, unsigned int *initialScore*)

Add a player in the ladder.

Add a player in the ladder

Parameters:

nick Nick to add
initialScore Initial player's score

Definition at line 125 of file lamoule.cpp.

References doc, Tools::intToStr(), and root.

Referenced by increaseScore().

6.20.3.2 bool Lamoule::deletePlayer (string *nick*)

Remove a player from the ladder.

Delete a player from the ladder

Parameters:

nick Nick to delete

Returns:

true i the nick has been deleted, else false

Definition at line 330 of file lamoule.cpp.

References doc, root, and Tools::to_lower().

Referenced by deleteplayer().

6.20.3.3 int Lamoule::generateScore ()

generate score

Generate a random score. If "nextscore" is set, then it's returned

Returns:

generated score

Definition at line 143 of file lamoule.cpp.

References FIRST_FLOOR, MAX_SCORE, nextScore, Tools::random(), and SECOND_FLOOR.

Referenced by lamoule().

6.20.3.4 vector< string > Lamoule::get5first (sort_criterion *criterion*, int *min_attempts*)

Get 5 first.

Get top 5 depending on score or average

Parameters:

criterion Sort criterion
min_attempts Minimal attempts number to appear in ladder

Returns:

A vector with top 5 players

Definition at line 262 of file lamoule.cpp.

References Tools::doubleToStr(), Tools::intToStr(), sort(), and Tools::strToDouble().

Referenced by top5(), and toptotal().

6.20.3.5 vector< string > Lamoule::getInfosPlayer (string *nick*, int *min_attempts*)

Get informations about a player.

Get informations about a player

- total
- nb lamoule
- average
- reset time
- rank

  Parameters:

  nick Player's nick
  min_attempts Minimal attempts number to have a rank

  Returns:

  A vector containing informations

Definition at line 292 of file lamoule.cpp.

References AVERAGE, Tools::doubleToStr(), Tools::intToStr(), root, sort(), Tools::strToDouble(), and Tools::to_lower().

Referenced by player().

6.20.3.6 vector< string > Lamoule::getTopShot ()

Get the lamoule's topshot.

Get the lamoule's topshot

Returns:

A vector containing the topshot informations (nick,score,date) (Check the vector size before using it ! (empty if a problem append))

Definition at line 89 of file lamoule.cpp.

References doc.

Referenced by increaseScore(), and topshot().

6.20.3.7 char Lamoule::increaseScore (string *nick*, int *score*, unsigned int *diffAttempts*, bool *checkTop*)

Increase score for a player.

Increase score for a player

Parameters:

nick Player's nick

score Player's score

diffAttempts Time between two attempts

checkTop Tell if topshot must be checked

Returns:

'null char' if the player plays too fast, 't' if tophsot, else 'o'

Definition at line 172 of file lamoule.cpp.

References addPlayer(), doc, getTopShot(), Tools::intToStr(), root, setTopShot(), Tools::strToDouble(), Tools::strToInt(), and Tools::to_lower().

Referenced by increase(), and lamoule().

6.20.3.8 void Lamoule::initFile () `[private]`

Initialize the XML file.

Initilaize the XML file by creating root and first childs

Definition at line 68 of file lamoule.cpp.

References doc, and root.

Referenced by Lamoule().

6.20.3.9 void Lamoule::purifyFile (int *reset_time*)

Purify File.

Purify the XML file by deleting player who didn't play since a while

Parameters:

reset_time Time (in seconds) after witch one a player is deleted

Definition at line 346 of file lamoule.cpp.

References doc, root, and Tools::strToInt().

Referenced by purifyFile().

6.20.3.10 bool Lamoule::setNextScore (int *score*)

Set next score.

Set next score

Parameters:

score Next score

Definition at line 362 of file lamoule.cpp.

References MAX_SCORE, and nextScore.

Referenced by nextscore().

6.20.3.11 void Lamoule::setTopShot (string *nick*, string *score*, string *date*)

Set the lamoule's topshot.

Set the lamoule's topshot

Parameters:

nick Topshoter's nick
score Topshot's score
date Topshot's date

Definition at line 108 of file lamoule.cpp.

References doc.

Referenced by increaseScore().

6.20.3.12 vector< TiXmlElement * > Lamoule::sort (sort_criterion *criterion*, int *min_attempts*)

Sort elements to get stats.

Sort a vector elements depending on scores or average

Parameters:

criterion Sort criterion
min_attempts Minimum attempts to be sorted

Postcondition:

The vector is sorted

Returns:

A vecotr containing nodes sorted

Definition at line 213 of file lamoule.cpp.

References AVERAGE, root, Tools::strToDouble(), Tools::strToInt(), and TOTAL.

Referenced by get5first(), and getInfosPlayer().

6.20.4 Member Data Documentation

6.20.4.1 TiXmlDocument* Lamoule::doc `[private]`

Represent the xml document.

Definition at line 61 of file lamoule.h.

Referenced by addPlayer(), deletePlayer(), getTopShot(), increaseScore(), initFile(), Lamoule(), purifyFile(), and setTopShot().

6.20.4.2 int Lamoule::FIRST_FLOOR `[private]`

First floor.

Definition at line 71 of file lamoule.h.

Referenced by generateScore(), and Lamoule().

6.20.4.3 int Lamoule::MAX_SCORE `[private]`

Maximul score.

Definition at line 69 of file lamoule.h.

Referenced by generateScore(), Lamoule(), and setNextScore().

6.20.4.4 unsigned int Lamoule::nextScore `[private]`

Stores the next score.

Definition at line 67 of file lamoule.h.

Referenced by generateScore(), Lamoule(), and setNextScore().

6.20.4.5 TiXmlNode* Lamoule::root `[private]`

Represent documents's root.

Definition at line 63 of file lamoule.h.

Referenced by addPlayer(), deletePlayer(), getInfosPlayer(), increaseScore(), initFile(), Lamoule(), purifyFile(), and sort().

6.20.4.6 int Lamoule::SECOND_FLOOR `[private]`

Second floor.

Definition at line 73 of file lamoule.h.

Referenced by generateScore(), and Lamoule().

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

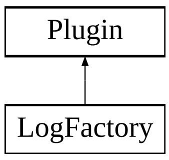
- src/plugins/lamoule.h
- src/plugins/lamoule.cpp

6.21 LogFactory Class Reference

This plugin manage channels logging.

`#include <logfactory.h>`

Inheritance diagram for LogFactory::



Public Member Functions

- LogFactory (BotKernel ∗)

  Constructor.
- ~LogFactory ()

  Destructor.
- bool hasToBeLogged (string)

  Tell if a given channel has to be logged.
- vector< Channel ∗ > getLoggedChannels ()

  Return logged channels.
- void destroyLogs ()

  Destroy logs.
- void cleanLogs ()

  clean logs
- bool newLog (string)

  Open a new log for a channel.
- void closeLog (string)

  Clode and delete a log.
- bool log (string, string)

  Log an event in a LogFile object.

Private Attributes

- map< string, LogFile ∗ > ∗ logs

  Map that sotred LogFile objects.
- BotKernel ∗ kernel

Stores a kernel pointerr.

6.21.1 Detailed Description

This plugin manage channels logging.

This plugin manage channels logging. It uses LogFile class provided by the kernel. LogFiles are stored in datas directory.

Definition at line 45 of file logfactory.h.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 LogFactory::LogFactory (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file logfactory.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, BotKernel::getDatasDir(), BotKernel::getSysLog(), IN_COMMAND_HANDLER, IN_LOOP, IN_TYPE_HANDLER, kernel, LogFile::log(), logs, Plugin::name, OUT_ALL_MSGS, Plugin::version, and WARNING.

6.21.2.2 LogFactory::∼LogFactory ()

Destructor.

Destructor

Definition at line 68 of file logfactory.cpp.

References destroyLogs(), and logs.

6.21.3 Member Function Documentation

6.21.3.1 void LogFactory::cleanLogs ()

clean logs

As configuration can change during process, a log file can be still openned while it has not to be logged anymore. This method will close log where the bot is no more present, or that have not to be logged anymore. Case : the bot join a channel, a log file is openned. Then the configuration change, and the channel has no more to be logged. The bot leave the channel. As it's no more in configuration file, the file is not closed. This method will ensure that no useless logs are openned. Call it in a timer.

Definition at line 123 of file logfactory.cpp.

References closeLog(), BotKernel::getPlugin(), UsersInfos::getUsers(), hasToBeLogged(), Tools::isInVector(), kernel, logs, and pPlugin::object.

6.21.3.2 void LogFactory::closeLog (string *channel*)

Clode and delete a log.

Close a log and delete its object

Parameters:

channel Channel for witch we close log

Definition at line 183 of file logfactory.cpp.

References LogFile::close(), and logs.

Referenced by cleanLogs().

6.21.3.3 void LogFactory::destroyLogs ()

Destroy logs.

Destroy log files

Definition at line 77 of file logfactory.cpp.

References logs.

Referenced by ∼LogFactory().

6.21.3.4 vector< Channel ∗ > LogFactory::getLoggedChannels ()

Return logged channels.

Return channels that are logged and where the bot is present

Returns:

A vector containg a pointer on Channel objects

Definition at line 97 of file logfactory.cpp.

References BotKernel::getCONFF(), BotKernel::getPlugin(), UsersInfos::getUsers(), ConfigurationFile::getValue(), kernel, Plugin::name, pPlugin::object, and Tools::stringToVector().

6.21.3.5 bool LogFactory::hasToBeLogged (string *channel*)

Tell if a given channel has to be logged.

Tell if a given channel has to be logged

Parameters:

channel Channel to test

Returns:

true if the channel has to be logged, else false

Definition at line 88 of file logfactory.cpp.

References BotKernel::getCONFF(), ConfigurationFile::getValue(), Tools::isInVector(), kernel, Plugin::name, and Tools::stringToVector().

Referenced by cleanLogs().

6.21.3.6 bool LogFactory::log (string *channel*, string *event*)

Log an event in a LogFile object.

Log an event in a logfile. If the log file is not opened, this function will try to do it

Parameters:

channel Channel where the event occured

event Event to log

Returns:

true if event has been logged

Definition at line 155 of file logfactory.cpp.

References LogFile::log(), logs, and newLog().

6.21.3.7 bool LogFactory::newLog (string *channel*)

Open a new log for a channel.

Open a new log for a channel and store it

Parameters:

channel Channel to log

Returns:

true is the log file has been opened, else false

Definition at line 173 of file logfactory.cpp.

References BotKernel::getDatasDir(), kernel, logs, and LogFile::open().

Referenced by log().

6.21.4 Member Data Documentation

6.21.4.1 BotKernel* LogFactory::kernel `[private]`

Stores a kernel pointerr.

Definition at line 51 of file logfactory.h.

Referenced by cleanLogs(), getLoggedChannels(), hasToBeLogged(), LogFactory(), and newLog().

6.21.4.2 map<string,LogFile*>* LogFactory::logs `[private]`

Map that sotred LogFile objects.

Definition at line 49 of file logfactory.h.

Referenced by cleanLogs(), closeLog(), destroyLogs(), log(), LogFactory(), newLog(), and ∼LogFactory().

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

- src/plugins/logfactory.h
- src/plugins/logfactory.cpp

6.22 LogFile Class Reference

Class that manage log system.

`#include <logfile.h>`

Public Member Functions

- LogFile (string, bool, bool, string, string)

  Constructor.

- ~LogFile ()

  Destructor.

- bool open ()

  Open log file.

- void close ()

  Close log file.

- void reopen ()

  Close and open log file.

- bool log (string line, log_level ll=NOTUSED)

  Log an event.

- void setPeriodFormat (string)

  Set period format.

- string getPeriodFormat ()

  Get period format.

- void setVerbose (bool)

  Set verbose state.

- bool getVerbose ()

  Get verbose state.

- void setKeepFiles (bool)

  Set keepfiles state.

- bool getKeepFiles ()

  Get keepfiles state.

- log_level getLogLevel ()

  Get log level.

- void setLogLevel (log_level)

  Set log level.

- void setLogLevel (string)

  Set log level.

Private Member Functions

- string systemPeriod ()

  Get system date according to periodFormat.

- void beginLog ()

  Init log file.

- void endLog ()

  Finish log file.

- bool checkFile ()

  Check if log file exists.

- string getLevelTag (log_level)

  Get level tag for a given level.

- log_level strToLogLevel (string)

  Get a log level according to a string.

Private Attributes

- ofstream $*$ stream

  File stream.

- log_level level

  log level

- string baseFileName

  Base file name.

- string period

  Period used.

- string periodFormat

  Period format.

- bool keepFiles

  keepFiles state

- bool verbose

  Verbose state.

6.22.1 Detailed Description

Class that manage log system.

This class is used to log bot events. Differents levels are available : error, warning, info and nothing

Definition at line 50 of file logfile.h.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 LogFile::LogFile (string *name*, bool *verbose*, bool *keep*, string *level*, string *format*)

Constructor.

Constructor. Initialize object members

Parameters:

name Base filename

verbose Set to true if you want a verbose logging (displays messages in console)

keep Set to true if you want to keep a log file when a new one is generated (period change)

level Log level : NOTHING,ERROR,WARNING or INFO

format format to finish log file name. Uses strftime function jokers.

Definition at line 40 of file logfile.cpp.

References baseFileName, setKeepFiles(), setLogLevel(), setPeriodFormat(), setVerbose(), stream, and strToLogLevel().

6.22.2.2 LogFile::∼LogFile ()

Destructor.

Destructor. Close the file and delete stream

Definition at line 52 of file logfile.cpp.

References close(), and stream.

6.22.3 Member Function Documentation

6.22.3.1 void LogFile::beginLog () `[private]`

Init log file.

Initialize log file

Definition at line 256 of file logfile.cpp.

References stream.

Referenced by open().

6.22.3.2 bool LogFile::checkFile () `[private]`

Check if log file exists.

Check if the log file exists

Returns:

True if file exists, else false

Definition at line 245 of file logfile.cpp.

References baseFileName, and period.

6.22.3.3 void LogFile::close ()

Close log file.

Finish the log file and close it.

Definition at line 76 of file logfile.cpp.

References endLog(), and stream.

Referenced by LogFactory::closeLog(), reopen(), and ∼LogFile().

6.22.3.4 void LogFile::endLog () `[private]`

Finish log file.

Finish log file

Definition at line 267 of file logfile.cpp.

References stream.

Referenced by close().

6.22.3.5 bool LogFile::getKeepFiles ()

Get keepfiles state.

Get keepFiles state

Returns:

keepFiles state

Definition at line 221 of file logfile.cpp.

References keepFiles.

6.22.3.6 string LogFile::getLevelTag (log_level *ll*) `[private]`

Get level tag for a given level.

Get a tag according to a given log level

Parameters:

ll Log level given

Returns:

Level taf according to the log level

Definition at line 280 of file logfile.cpp.

References ERROR, INFO, NOTHING, NOTUSED, and WARNING.

Referenced by log().

6.22.3.7 log_level LogFile::getLogLevel ()

Get log level.

Get log level

Returns:

actual log level

Definition at line 205 of file logfile.cpp.

References level.

6.22.3.8 string LogFile::getPeriodFormat ()

Get period format.

Get period format

Returns:

Period format

Definition at line 181 of file logfile.cpp.

References periodFormat.

Referenced by systemPeriod().

6.22.3.9 bool LogFile::getVerbose ()

Get verbose state.

Get verbose state

Returns:

verbose state

Definition at line 237 of file logfile.cpp.

References verbose.

6.22.3.10 bool LogFile::log (string *line*, log_level *ll* = `NOTUSED`)

Log an event.

Log an event to the log file, and displays it if verbose is set to true. If the log file is not present, it will be created. If the period has change (you log a file per month, and month has changed for example), log file will be closed and a new one will be created, with a name according to the new period. If keepFiles is set to true, the old file will be kept, else, this function will delete it.

Parameters:

line Line to log

ll log level used for this line. Can be NOTUSED (default, log level system is not used, the line will be logged without taking care about level), ERROR,WARNING,INFO

Returns:

true if the line has been logged, else false

Definition at line 112 of file logfile.cpp.

References baseFileName, ERROR, getLevelTag(), INFO, keepFiles, level, NOTHING, NOTUSED, period, reopen(), stream, systemPeriod(), verbose, and WARNING.

Referenced by BotKernel::addCountDown(), addIgnore(), addOnlyon(), addsuperadmin(), addtempsuperadmin(), bannedHandler(), checkConnection(), clearCountDowns(), BotKernel::connect(), deletekey(), delIgnore(), delOnlyon(), delsuperadmin(), disable(), disconnect(), enable(), error(), BotKernel::executeFunction(), FedoraProject::FedoraProject(), flushconffile(), joinChannel(), kickall(), kickHandler(), leaveChannel(), load(), loadconffile(), loadnocheck(), BotKernel::loadPlugin(), BotKernel::loadPlugins(), LogFactory::log(), LogFactory::LogFactory(), masskick(), onEndOfMOTD(), onInvite(), opall(), randomKick(), BotKernel::reconnect(), rejoinChan(), reloadfas(), reset(), BotKernel::run(), secondaryNick(), BotKernel::send(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), BotKernel::stop(), BotKernel::storeFunction(), RemoteControl::tcpServer(), unload(), unloadnocheck(), BotKernel::unloadPlugin(), unopall(), unvoiceall(), voiceall(), and BotKernel::∼BotKernel().

6.22.3.11 bool LogFile::open ()

Open log file.

Open the log file and initilize it. If the file does not exists, it's created, according to base filename and period format

Returns:

true if log file has been opened, else false

Definition at line 62 of file logfile.cpp.

References baseFileName, beginLog(), period, stream, and systemPeriod().

Referenced by BotKernel::BotKernel(), LogFactory::newLog(), and reopen().

6.22.3.12 void LogFile::reopen ()

Close and open log file.

Close the log file, then open it

Definition at line 85 of file logfile.cpp.

References close(), and open().

Referenced by log().

6.22.3.13 void LogFile::setKeepFiles (bool *state*)

Set keepfiles state.

Set keepFiles state

Parameters:

state State to set

Definition at line 213 of file logfile.cpp.

References keepFiles.

Referenced by LogFile(), and setlogkeepfiles().

6.22.3.14 void LogFile::setLogLevel (string *ll*)

Set log level.

Set log level

Parameters:

ll log level

Definition at line 189 of file logfile.cpp.

References level, and strToLogLevel().

6.22.3.15 void LogFile::setLogLevel (log_level *ll*)

Set log level.

Set log level

Parameters:

ll log level

Definition at line 197 of file logfile.cpp.

References level.

Referenced by LogFile(), and setloglevel().

6.22.3.16 void LogFile::setPeriodFormat (string *format*)

Set period format.

Set period format

Parameters:

format period format

Definition at line 173 of file logfile.cpp.

References periodFormat.

Referenced by LogFile(), and setlogperiod().

6.22.3.17 void LogFile::setVerbose (bool *state*)

Set verbose state.

Set verbose state

Parameters:

state Verbose state

Definition at line 229 of file logfile.cpp.

References verbose.

Referenced by LogFile().

6.22.3.18 log_level LogFile::strToLogLevel (string *ll*) `[private]`

Get a log level according to a string.

Convert a string log level to a log_level format

Parameters:

ll log level to convert

Returns:

log level converted

Definition at line 156 of file logfile.cpp.

References ERROR, INFO, NOTHING, NOTUSED, and WARNING.

Referenced by LogFile(), and setLogLevel().

6.22.3.19 string LogFile::systemPeriod () `[private]`

Get system date according to periodFormat.

Give system period (time) according to given format

Returns:

System period

Definition at line 94 of file logfile.cpp.

References getPeriodFormat().

Referenced by log(), and open().

6.22.4 Member Data Documentation

6.22.4.1 string LogFile::baseFileName `[private]`

Base file name.

Definition at line 88 of file logfile.h.

Referenced by checkFile(), log(), LogFile(), and open().

6.22.4.2 bool LogFile::keepFiles `[private]`

keepFiles state

Definition at line 96 of file logfile.h.

Referenced by getKeepFiles(), log(), and setKeepFiles().

6.22.4.3 log_level LogFile::level `[private]`

log level

Definition at line 86 of file logfile.h.

Referenced by getLogLevel(), log(), and setLogLevel().

6.22.4.4 string LogFile::period `[private]`

Period used.

Definition at line 90 of file logfile.h.

Referenced by checkFile(), log(), and open().

6.22.4.5 string LogFile::periodFormat `[private]`

Period format.

Definition at line 92 of file logfile.h.

Referenced by getPeriodFormat(), and setPeriodFormat().

6.22.4.6 ofstream* LogFile::stream `[private]`

File stream.

Definition at line 84 of file logfile.h.

Referenced by beginLog(), close(), endLog(), log(), LogFile(), open(), and ~LogFile().

6.22.4.7 bool LogFile::verbose `[private]`

Verbose state.

Definition at line 98 of file logfile.h.

Referenced by getVerbose(), log(), and setVerbose().

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

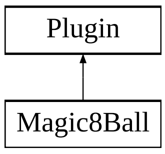
- src/logfile.h
- src/logfile.cpp

6.23 Magic8Ball Class Reference

magic 8 ball game

`#include <magic8ball.h>`

Inheritance diagram for Magic8Ball::



Public Member Functions

- Magic8Ball (BotKernel ∗)

  Constructor.

- string getRandomAnswer ()

  Return a random answer.

Private Attributes

- string answers [20]

  magic answers

6.23.1 Detailed Description

magic 8 ball game

Plugin simulating magic 8 ball game see `http://en.wikipedia.org/wiki/Magic_8-Ball`

Definition at line 42 of file magic8ball.h.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Magic8Ball::Magic8Ball (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file magic8ball.cpp.

References answers, Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_-HANDLER, Plugin::name, and Plugin::version.

6.23.3 Member Function Documentation

6.23.3.1 string Magic8Ball::getRandomAnswer ()

Return a random answer.

Get a random answer

Returns:

the answer

Definition at line 68 of file magic8ball.cpp.

References answers, and Tools::random().

Referenced by ball().

6.23.4 Member Data Documentation

6.23.4.1 string Magic8Ball::answers[20] `[private]`

magic answers

Definition at line 46 of file magic8ball.h.

Referenced by getRandomAnswer(), and Magic8Ball().

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

- src/plugins/magic8ball.h
- src/plugins/magic8ball.cpp

6.24 Message Class Reference

Class that manage messages from the irc server.

`#include <message.h>`

Public Member Functions

- Message (string)

  Constructor.

- Message ()

  Constructor.

- ∼Message ()

  Destructor.

- vector< string > getSplit ()

  Return all parts of the message.

- void setMessage (string)

  Set the message string.

- unsigned int nbParts ()

  Return parts number.

- string getPart (unsigned int)

  Return a part of the message.

- string getSender ()

  Return the message's sender's informations (nick,ident,host).

- string getNickSender ()

  Message's sender's nick.

- string getHostSender ()

  Message's sender's host.

- string getIdentSender ()

  Message's sender's ident.

- bool isPrivate ()

  True if the message is a private one.

- bool isPublic ()

  True if the message is a public one (channel).

- string getSource ()

  Return the message source (channel or botnick).

- string getMessage ()

  Return the raw msg (irc format).

- time_t getElapsedTime ()

  Get elapsed time between message creation and now.

Private Attributes

- string message

  The raw message.

- vector< string > split

  Message splitted (by spaces).

- bool pv

  True oif the message is private.

- time_t timestamp

  timestamp of the message

6.24.1 Detailed Description

Class that manage messages from the irc server.

This class stores messages from the irc server and parse them to make informations easier to obtain. Please be carefull. All methods works with message comming FROM the server, not for those that are sended TO the server

Definition at line 44 of file message.h.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Message::Message (string *message*)

Constructor.

Class constructor Split different parts of the messaeg to make it easier to use

Parameters:

message IRC message

Definition at line 37 of file message.cpp.

References setMessage().

6.24.2.2 Message::Message ()

Constructor.

Constructor Used for unknow string that are set after whith setMessage()

Definition at line 46 of file message.cpp.

References message, and split.

6.24.2.3 Message::∼Message ()

Destructor.

Class destructor

Definition at line 55 of file message.cpp.

6.24.3 Member Function Documentation

6.24.3.1 time_t Message::getElapsedTime ()

Get elapsed time between message creation and now.

Get elapsed time between message creation and now (in seconds)

Returns:

Elapsed time between message creation and now

Definition at line 214 of file message.cpp.

References timestamp.

Referenced by testMsgTimestamp().

6.24.3.2 string Message::getHostSender ()

Message's sender's host.

Get message sender's host

Returns:

message sender's host

Definition at line 132 of file message.cpp.

References getPart(), message, and split.

Referenced by joinHandler(), kickHandler(), modeHandler(), and onJoin().

6.24.3.3 string Message::getIdentSender ()

Message's sender's ident.

Get message sender's ident

Returns:

message sender's ident

Definition at line 146 of file message.cpp.

References getPart(), message, and split.

Referenced by onJoin().

6.24.3.4 string Message::getMessage ()

Return the raw msg (irc format).

Get the raw message

Returns:

The raw message

Definition at line 205 of file message.cpp.

References message.

Referenced by BotKernel::addCountDown(), displayAdvertise(), error(), launchSurvey(), rejoinChan(), and BotKernel::send().

6.24.3.5 string Message::getNickSender ()

Message's sender's nick.

Get message sender's nick

Returns:

message sender's nick

Definition at line 118 of file message.cpp.

References getPart(), and message.

Referenced by addad(), addIgnore(), addOnlyon(), addQuote(), addsuperadmin(), addtempsuperadmin(), adinfos(), autoop(), autovoice(), ban(), bandel(), baninfos(), banlist(), banmask(), chanlev(), clearCountDowns(), commandsStatus(), ctcp_ping(), ctcp_version(), delad(), deletekey(), deleteplayer(), delIgnore(), delOnlyon(), delQuote(), delsuperadmin(), disable(), enable(), BotKernel::executeFunction(), flushconffile(), getconfvalue(), getnbcountdowns(), help(), ignoreList(), increase(), isIgnored(), joinHandler(), kick(), kickall(), kickHandler(), lamoule(), launchSurvey(), listads(), listlibs(), listmodules(), load(), loadconffile(), loadnocheck(), masskick(), modeHandler(), modeHandlerProtect(), moduleinfos(), myFunction(), nextscore(), nick(), nickHandler(), onJoin(), online(), onPart(), onQuit(), op(), opall(), partHandler(), prefix(), privmsgHandler(), quitHandler(), quoteInfos(), randomKick(), reloadfas(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setSuperAdminPass(), slapme(), slapUser(), stopSurvey(), superadminlist(), sysinfos(), topic(), topicHandler(), unautoop(), unautovoice(), unbanall(), unload(), unloadnocheck(), unop(), unopall(), unvoice(), unvoiceall(), uptime(), version(), voice(), voiceall(), vote(), and whoami().

6.24.3.6 string Message::getPart (unsigned int *index*)

Return a part of the message.

Return a part of the message. This part is the one pointed by the given index

Parameters:

index Index of the wanted message part (start to zero)

Returns:

Part of the message. empty if index is superior to the parts message number

Definition at line 180 of file message.cpp.

References split.

Referenced by addad(), addIgnore(), addOnlyon(), addsuperadmin(), addtempsuperadmin(), adinfos(), allowedCommandCheck(), ban(), bandel(), baninfos(), banmask(), bannedHandler(), bug(), chanlev(), checkBug(), ctcp_ping(), cycleChannel(), danstonchat(), delad(), deletekey(), deleteplayer(), delIgnore(), delOnlyon(), delQuote(), delsuperadmin(), disable(), enable(), event352(), fas(), getconfvalue(), getHostSender(), getIdentSender(), getNickSender(), getSender(), getSource(), hl(), host2ip(), increase(), invite(), ip2host(), isIgnored(), joinChannel(), kick(), kickHandler(), lastseen(), leaveChannel(), load(), loadnocheck(), masskick(), mode(), modeHandler(), modeHandlerProtect(), moduleinfos(), BotKernel::msgTreatment(), nextscore(), nick_changed(), nickHandler(), notice(), onInvite(), onKick(), op(), pinged(), planet(), player(), privmsgHandler(), q3(), quote(), quoteInfos(), sendHandler(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), slapme(), tell(), testIgnoredUser(), testMsgTimestamp(), topicInfos(), topicJoin(), trad(), unload(), unloadnocheck(), unop(), unvoice(), voice(), vote(), warsow(), whoowns(), and wiki().

6.24.3.7 string Message::getSender ()

Return the message's sender's informations (nick,ident,host).

Give the message sender (nick host ident)

Returns:

message sender

Definition at line 104 of file message.cpp.

References getPart(), and split.

Referenced by addad(), addIgnore(), addOnlyon(), addQuote(), addsuperadmin(), addtempsuperadmin(), adinfos(), autoop(), autovoice(), ban(), bandel(), baninfos(), banlist(), banmask(), chanlev(), clearCountDowns(), commandsStatus(), cycleChannel(), delad(), deletekey(), deleteplayer(), delIgnore(), delOnlyon(), delQuote(), delsuperadmin(), disable(), disconnect(), enable(), flushconffile(), getconfvalue(), getnbcountdowns(), ignoreList(), increase(), invite(), isIgnored(), joinChannel(), joinHandler(), kick(), kickall(), kickHandler(), leaveChannel(), listads(), listlibs(), listmodules(), load(), loadconffile(), loadnocheck(), masskick(), modeHandler(), modeHandlerProtect(), moduleinfos(), nextscore(), notice(), onInvite(), op(), opall(), partHandler(), protectmodes(), protecttopic(), quoteInfos(), randomKick(), raw(), reauth(), reloadfas(), reset(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), stopSurvey(), superadminlist(), tell(), testIgnoredUser(), testMsgTimestamp(), topic(), topicHandler(), unautoop(), unautovoice(), unbanall(), unload(), unloadnocheck(), unop(), unopall(), unprotectmodes(), unprotecttopic(), unvoice(), unvoiceall(), voice(), voiceall(), and whoami().

6.24.3.8 string Message::getSource ()

Return the message source (channel or botnick).

Get the message source (channel or bot nick if private)

Returns:

Message source

Definition at line 196 of file message.cpp.

References getPart().

Referenced by allowedCommandCheck(), autoop(), autovoice(), ball(), ban(), bandel(), baninfos(), banlist(), banmask(), bug(), bzsearch(), checkBug(), danstonchat(), displayPaste(), endSurvey(), fas(), greplog(), hl(), host2ip(), ip2host(), joinHandler(), kick(), kickall(), kickHandler(), lamoule(), lastQuote(), lastseen(), launchSurvey(), masskick(), mode(), modeHandler(), modeHandlerProtect(), myFunction(), nick(), onInvite(), onJoin(), onKick(), onPart(), op(), opall(), partHandler(), planet(), player(), privmsgHandler(), protectmodes(), protecttopic(), q3(), quote(), randomKick(), searchQuote(), setMessage(), slapUser(), stopSurvey(), tele(), top5(), topic(), topicHandler(), topshot(), toptotal(), trad(), unautoop(), unautovoice(), unbanall(), unop(), unopall(), unprotectmodes(), unprotecttopic(), unvoice(), unvoiceall(), voice(), voiceall(), vote(), warsow(), whoowns(), and wiki().

6.24.3.9 vector< string > Message::getSplit ()

Return all parts of the message.

Return different parts of the message

Returns:

A vector containing string representing parts of the message

Definition at line 95 of file message.cpp.

References split.

Referenced by addad(), addIgnore(), addOnlyon(), addQuote(), addsuperadmin(), addtempsuperadmin(), allowedCommandCheck(), ban(), banmask(), bzsearch(), chanlev(), cycleChannel(), deletekey(), delIgnore(), delOnlyon(), delsuperadmin(), disable(), enable(), event005(), getconfvalue(), greplog(), host2ip(), ip2host(), isIgnored(), joinChannel(), kick(), kickHandler(), leaveChannel(), masskick(), mode(), modeHandler(), BotKernel::msgTreatment(), notice(), op(), partHandler(), planet(), privmsgHandler(), quitHandler(), raw(), searchQuote(), sendHandler(), setconfvalue(), setNick(), setSuperAdminPass(), slapUser(), tell(), topic(), topicHandler(), topicJoin(), trad(), unop(), unvoice(), voice(), and wiki().

6.24.3.10 bool Message::isPrivate ()

True if the message is a private one.

Tell if the message is private to the bot

Returns:

True if it's a private message, else false

Definition at line 160 of file message.cpp.

References pv.

Referenced by addad(), addIgnore(), addOnlyon(), addsuperadmin(), addtempsuperadmin(), adinfos(), chanlev(), clearCountDowns(), commandsStatus(), cycleChannel(), delad(), deletekey(), delIgnore(), delOnlyon(), delsuperadmin(), disable(), disconnect(), enable(), flushconffile(), getconfvalue(), getnbcountdowns(), ignoreList(), invite(), isIgnored(), isPublic(), joinChannel(), leaveChannel(), listads(), listlibs(), listmodules(), load(), loadconffile(), loadnocheck(), moduleinfos(), notice(), onInvite(), raw(), reauth(), reloadfas(), reset(), setconfvalue(), setlogkeepfiles(), setloglevel(), setlogperiod(), setNick(), setSuperAdminPass(), superadminlist(), tell(), unload(), and unloadnocheck().

6.24.3.11 bool Message::isPublic ()

True if the message is a public one (channel).

Tell if the message is public (on a channel)

Returns:

True if it's a public message, else false

Definition at line 169 of file message.cpp.

References isPrivate().

Referenced by addQuote(), allowedCommandCheck(), autoop(), autovoice(), ball(), ban(), bandel(), baninfos(), banlist(), banmask(), bug(), bzsearch(), checkBug(), danstonchat(), deleteplayer(), delQuote(), displayPaste(), fas(), greplog(), hl(), host2ip(), increase(), ip2host(), kick(), kickall(), lamoule(), lastQuote(), lastseen(), launchSurvey(), masskick(), myFunction(), nextscore(), op(), opall(), planet(), player(), privmsgHandler(), protectmodes(), protecttopic(), q3(), quote(), quoteInfos(), randomKick(), searchQuote(), slapme(), stopSurvey(), tele(), top5(), topic(), topshot(), toptotal(), trad(), unautoop(), unautovoice(), unbanall(), unop(), unopall(), unprotectmodes(), unprotecttopic(), unvoice(), unvoiceall(), voice(), voiceall(), vote(), warsow(), whoowns(), and wiki().

6.24.3.12 unsigned int Message::nbParts ()

Return parts number.

Return parts number's message

Returns:

parts number

Definition at line 87 of file message.cpp.

References split.

Referenced by addad(), addQuote(), adinfos(), ball(), ban(), bandel(), baninfos(), banmask(), bug(), bzsearch(), checkBug(), danstonchat(), delad(), deleteplayer(), delQuote(), fas(), greplog(), hl(), increase(), invite(), lastseen(), load(), loadnocheck(), moduleinfos(), BotKernel::msgTreatment(), nextscore(), partHandler(), planet(), player(), q3(), quote(), quoteInfos(), searchQuote(), setlogkeepfiles(), setloglevel(), setlogperiod(), slapme(), trad(), unload(), unloadnocheck(), warsow(), whoowns(), and wiki().

6.24.3.13 void Message::setMessage (string *message*)

Set the message string.

Set the message string and split it

Parameters:

message IRC message

Definition at line 64 of file message.cpp.

References getSource(), pv, split, Tools::stringToVector(), and timestamp.

Referenced by addad(), Message(), BotKernel::run(), and BotKernel::send().

6.24.4 Member Data Documentation

6.24.4.1 string Message::message `[private]`

The raw message.

Definition at line 80 of file message.h.

Referenced by getHostSender(), getIdentSender(), getMessage(), getNickSender(), and Message().

6.24.4.2 bool Message::pv `[private]`

True oif the message is private.

Definition at line 84 of file message.h.

Referenced by isPrivate(), and setMessage().

6.24.4.3 vector<string> Message::split `[private]`

Message splitted (by spaces).

Definition at line 82 of file message.h.

Referenced by getHostSender(), getIdentSender(), getPart(), getSender(), getSplit(), Message(), nbParts(), and setMessage().

6.24.4.4 time_t Message::timestamp `[private]`

timestamp of the message

Definition at line 86 of file message.h.

Referenced by getElapsedTime(), and setMessage().

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

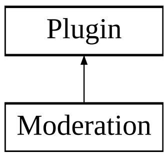
- src/message.h
- src/message.cpp

6.25 Moderation Class Reference

Channel moderation.

`#include <moderation.h>`

Inheritance diagram for Moderation::



Public Member Functions

- Moderation (BotKernel ∗)

  Constructor.

- bool addBan (string, string, unsigned int, string, string)

  add a ban on a channel

- string delBan (string, unsigned int)

  del a ban on a channel

- bool isBanned (string, string)

  tell if a user is banned

- vector< string > getBanList (string)

  give ban list on a channel

- vector< string > banInfos (string, unsigned int)

  give details about a ban

- vector< string > clearList (string)

  clear all bans for a channel and return them (to apply it on the chan)

- vector< string > clearOutBans (vector< string >)

  CLear all outdated bans for all channels and return them (to apply it on chans).

- bool checkAccess (string, string, unsigned int, BotKernel ∗)

  Check access on a chan for a user.

- bool hasOpPrivileges (string, string, string, BotKernel ∗)

  Check if a user is opped, has level >= 2 or is super admin.

- vector< string ∗ > getChanUsersList (string, BotKernel ∗)

  Get a channel users list.

- bool checkMode (string, string, char, BotKernel ∗)

Check if a user has the given mode on a given channel using UsersInfos module.

- unsigned int getRejoinAttempts (string)

  Get rejoin attempts number for a channel.

- void bumpRejoinAttempts (string)

  Bump rejoin attempts for a channel.

- void clearRejoinAttempts (string)

  Clear attempts for a channel.

Private Member Functions

- void initFile ()

  Initialize the XML file.

Private Attributes

- TiXmlDocument ∗ doc

  Represent the xml document.

- TiXmlNode ∗ root

  Represent documents's root.

- map< string, int > rejoinAttempts

  Stores rejoin attempts.

6.25.1 Detailed Description

Channel moderation.

This plugin allow channels moderation (op, voice, kick,bans etc ...) with acces (if admin plugin is loaded)

Definition at line 53 of file moderation.h.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Moderation::Moderation (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file moderation.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, doc, BotKernel::getDatasDir(), IN_COMMAND_HANDLER, IN_LOOP, IN_TYPE_HANDLER, initFile(), Plugin::name, root, and Plugin::version.

6.25.3 Member Function Documentation

6.25.3.1 bool Moderation::addBan (string *channel*, string *mask*, unsigned int *duration*, string *by*, string *reason*)

add a ban on a channel

Add a ban for a host on a given channel Stores the ban in a XML file

Parameters:

channel Channel on witch the user is banned

mask User's mask : nick!ident@host

duration ban time (seconds)

by banner mask

reason Reason for the ban

Returns:

true if the user has been banned, else false

Definition at line 115 of file moderation.cpp.

References doc, and isBanned().

Referenced by ban(), and banmask().

6.25.3.2 vector< string > Moderation::banInfos (string *channel*, unsigned int *index*)

give details about a ban

Return informations about a ban

Parameters:

channel Ban channel

index Ban index

Returns:

A vector containing informations

Definition at line 222 of file moderation.cpp.

References doc, and Tools::strToInt().

Referenced by baninfos().

6.25.3.3 void Moderation::bumpRejoinAttempts (string *channel*)

Bump rejoin attempts for a channel.

Bump rejoin attempts for a channel

Parameters:

channel Channel for which we want to bump attempts number

Definition at line 447 of file moderation.cpp.

References rejoinAttempts.

Referenced by rejoinChan().

6.25.3.4 bool Moderation::checkAccess (string *channel*, string *mask*, unsigned int *level*, BotKernel ∗ *b*)

Check access on a chan for a user.

.Check if a user has the given access on the given channel. The mode is the access one, in the xml file, not the one on the channel

Parameters:

channel Channel where to check access

mask Mask to check

level Level to check

b Pointer to the bot kerne

Returns:

True if the user has the given access, else false

Definition at line 325 of file moderation.cpp.

References BotKernel::getPlugin(), Admin::getUserLevel(), and pPlugin::object.

Referenced by modeHandlerProtect(), protectmodes(), protecttopic(), topicHandler(), unprotectmodes(), and unprotecttopic().

6.25.3.5 bool Moderation::checkMode (string *channel*, string *nick*, char *mode*, BotKernel ∗ *b*)

Check if a user has the given mode on a given channel using UsersInfos module.

Check if a user has the given mode on a given channel using UsersInfos module

Parameters:

mode Mode to test for the user

channel Channel to check

nick Nick to check

b Pointer to the bot kernel

Precondition:

"UsersInfos must be loaded

Returns:

True if the user has the mode, else false

Definition at line 381 of file moderation.cpp.

References BotKernel::getPlugin(), UsersInfos::hasMode(), and pPlugin::object.

Referenced by clearOutBans(), kickHandler(), opall(), partHandler(), quitHandler(), unopall(), unvoiceall(), and voiceall().

6.25.3.6 vector< string > Moderation::clearList (string *channel*)

clear all bans for a channel and return them (to apply it on the chan)

Clear all bans for a given channel even if they are not out dated Return them to be deleted on the channel

Parameters:

channel Channel where delete bans

Returns:

A vector containing the deleted bans

Definition at line 255 of file moderation.cpp.

References doc.

Referenced by unbanall().

6.25.3.7 vector< string > Moderation::clearOutBans (vector< string > *myChans*)

CLear all outdated bans for all channels and return them (to apply it on chans).

Clear all outdated bans Return them to be deleted on the channel

Parameters:

myChans Chans where the bot is (whithout '#' first char), to be sure to not delete bans where the bot is not present

Returns:

A vector containing the deleted bans

Definition at line 280 of file moderation.cpp.

References doc, Tools::isInVector(), Tools::strToInt(), and IRCProtocol::unban().

Referenced by clearOutBans().

6.25.3.8 void Moderation::clearRejoinAttempts (string *channel*)

Clear attempts for a channel.

Clear attempts for a channel

Parameters:

channel Channel for which we want to clear rejoin attempts

Definition at line 462 of file moderation.cpp.

References rejoinAttempts.

Referenced by joinHandler().

6.25.3.9 string Moderation::delBan (string *channel*, unsigned int *index*)

del a ban on a channel

Delete a ban and return the mask (to delete it on the chan)

Parameters:

channel Channel where delete the ban
index Ban index

Returns:

ban's mask deleted, or empty string if ban not found

Definition at line 174 of file moderation.cpp.

References doc.

Referenced by bandel().

6.25.3.10 vector< string > Moderation::getBanList (string *channel*)

give ban list on a channel

Return ban list for a given channel

Parameters:

channel Channel to list

Returns:

A vector containing ban list

Definition at line 198 of file moderation.cpp.

References doc, and Tools::intToStr().

Referenced by banlist().

6.25.3.11 vector< string * > Moderation::getChanUsersList (string *channel*, BotKernel * *b*)

Get a channel users list.

Get a channel users list using "usersinfos" plugin Users a stored in a string tab like that : tab[0]=nick; tab[1]=host; tab[2]=ident; tab[3]=status;

Parameters:

channel Channel to get the list
b Pointer to the bot kernel

Precondition:

UsersInfos module must be loaded

Returns:

A vector containing the users list

Definition at line 409 of file moderation.cpp.

References BotKernel::getPlugin(), UsersInfos::getUsers(), and pPlugin::object.

Referenced by banmask(), kickall(), opall(), randomKick(), unopall(), unvoiceall(), and voiceall().

6.25.3.12 unsigned int Moderation::getRejoinAttempts (string *channel*)

Get rejoin attempts number for a channel.

Get rejoin attempts number for a channel

Parameters:

channel Channel for which we want to get rejoin attempts number

Returns:

Rejoin attempts number

Definition at line 432 of file moderation.cpp.

References rejoinAttempts.

Referenced by rejoinChan().

6.25.3.13 bool Moderation::hasOpPrivileges (string *channel*, string *mask*, string *nick*, BotKernel ∗ *b*)

Check if a user is opped, has level >= 2 or is super admin.

Check if a user is opped, has level >= 2 or is super admin using "admin" and "usersinfos" modules

Parameters:

channel Channel where to check access

mask Mask to check access on the given channel

nick Nick to check access on the given channel

b Pointer to the bot kernel

Returns:

True if the user, has op privileges, else false

Precondition:

Admin and UsersInfos module must be loaded

Definition at line 348 of file moderation.cpp.

References BotKernel::getPlugin(), Admin::getUserLevel(), UsersInfos::hasMode(), Admin::isSuperAdmin(), and pPlugin::object.

Referenced by autoop(), autovoice(), ban(), bandel(), baninfos(), banlist(), banmask(), kick(), kickall(), masskick(), op(), opall(), randomKick(), topic(), unautoop(), unautovoice(), unbanall(), unop(), unopall(), unvoice(), unvoiceall(), voice(), and voiceall().

6.25.3.14 void Moderation::initFile () `[private]`

Initialize the XML file.

Initilaize the XML file by creating root and first childs (file empty structure)

Definition at line 95 of file moderation.cpp.

References doc, and root.

Referenced by Moderation().

6.25.3.15 bool Moderation::isBanned (string *channel*, string *mask*)

tell if a user is banned

Tell if a user is banned on a ginven channel

Parameters:

channel Channel to test

mask User's mask

Returns:

True is the user is banned, else false

Definition at line 152 of file moderation.cpp.

References doc, and Tools::ircMaskMatch().

Referenced by addBan(), and joinHandler().

6.25.4 Member Data Documentation

6.25.4.1 TiXmlDocument∗ Moderation::doc `[private]`

Represent the xml document.

Definition at line 57 of file moderation.h.

Referenced by addBan(), banInfos(), clearList(), clearOutBans(), delBan(), getBanList(), initFile(), is-Banned(), and Moderation().

6.25.4.2 map<string,int> Moderation::rejoinAttempts `[private]`

Stores rejoin attempts.

Definition at line 63 of file moderation.h.

Referenced by bumpRejoinAttempts(), clearRejoinAttempts(), and getRejoinAttempts().

6.25.4.3 TiXmlNode∗ Moderation::root `[private]`

Represent documents's root.

Definition at line 59 of file moderation.h.

Referenced by initFile(), and Moderation().

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

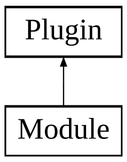
- src/plugins/moderation.h
- src/plugins/moderation.cpp

6.26 Module Class Reference

Modules management.

`#include <module.h>`

Inheritance diagram for Module::



Public Member Functions

- Module (BotKernel ∗)

  Constructor.

6.26.1 Detailed Description

Modules management.

This plugin manage modules

Definition at line 44 of file module.h.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Module::Module (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file module.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, IN_-COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

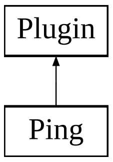
- src/plugins/module.h
- src/plugins/module.cpp

6.27 Ping Class Reference

Manage ping events.

`#include <ping.h>`

Inheritance diagram for Ping::



Public Member Functions

- Ping (BotKernel ∗)

  Construcor.

- void setPonged (bool)

  Set ponged value.

- bool getPonged ()

  Get ponged value.

Private Attributes

- bool ponged

  Indicate if the server has sended a good pong.

6.27.1 Detailed Description

Manage ping events.

This class provides methods to manage ping events and test connection with the server

Definition at line 43 of file ping.h.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Ping::Ping (BotKernel ∗ *b*)

Construcor.

Constructor

Definition at line 34 of file ping.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_FIRST_WORD, IN_LOOP, IN_TYPE_HANDLER, Plugin::name, setPonged(), and Plugin::version.

6.27.3 Member Function Documentation

6.27.3.1 bool Ping::getPonged ()

Get ponged value.

Get ponged value

Returns:

Ponged value

Definition at line 61 of file ping.cpp.

References ponged.

Referenced by checkConnection().

6.27.3.2 void Ping::setPonged (bool *value*)

Set ponged value.

Set ponged value

Parameters:

value Ponged value

Definition at line 52 of file ping.cpp.

References ponged.

Referenced by checkConnection(), Ping(), and pongMe().

6.27.4 Member Data Documentation

6.27.4.1 bool Ping::ponged `[private]`

Indicate if the server has sended a good pong.

Definition at line 47 of file ping.h.

Referenced by getPonged(), and setPonged().

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

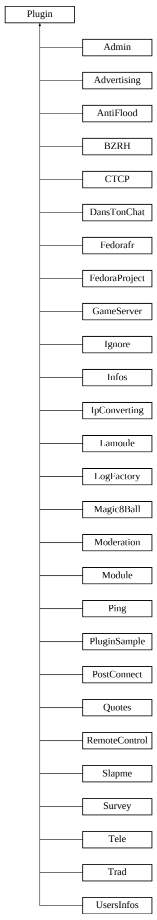
- src/plugins/ping.h
- src/plugins/ping.cpp

6.28 Plugin Class Reference

Class that manage a plugin.

`#include <plugin.h>`

Inheritance diagram for Plugin::



Public Member Functions

- Plugin ()

  Constructor.

- virtual ~Plugin ()

  Destructor.

- vector< StructFunctionStorage > getFunctions ()

  Return plugin functions.

- string getAuthor ()

  Return author.

- string getDescription ()

  Return description.

- string getVersion ()

  Return version.

- string getName ()

  Return name.

- bool checkMembers ()

  Check if all members are given.

- void bindFunction (string, func_type, string, time_t, unsigned int)

  Register a plugin function.

- void ∗ getHandle ()

  Get plugin's handle.

- void setHandle (void ∗)

  Set plugin's handle.

- void addRequirement (string)

  Add a plugin that is required for the plugin.

- vector< string > getRequirements ()

  Get requirements list.

- bool requires (string)

  Tell if this plugin requires an other one.

Protected Attributes

- string author

  Plugin author.

- string description

  Plugin descritpion.

- string version

  Plugin version.

- string name

  Plugin name.

- vector< StructFunctionStorage > funcs

  PLugins functions.

- void ∗ handle

Stores plugin's handle.

- vector< string > requirements

  stores plugins names that are required for the plugin

6.28.1 Detailed Description

Class that manage a plugin.

This class is a plugin head witch all the plugin class must derive. It contains function to get functions from the plugin, and check if all informations are given

Definition at line 83 of file plugin.h.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Plugin::Plugin ()

Constructor.

Class constructor Initialize private attributes

Definition at line 35 of file plugin.cpp.

References author, description, funcs, handle, name, requirements, and version.

6.28.2.2 Plugin::∼Plugin () `[virtual]`

Destructor.

Class destructor

Definition at line 49 of file plugin.cpp.

6.28.3 Member Function Documentation

6.28.3.1 void Plugin::addRequirement (string *plugin*)

Add a plugin that is required for the plugin.

Add a requirement for this plugin. It means that to load this plugin the requirement will have to be loaded before.

Parameters:

plugin Plugin that is required

Definition at line 163 of file plugin.cpp.

References requirements.

Referenced by Advertising::Advertising(), AntiFlood::AntiFlood(), FedoraProject::FedoraProject(), Ignore::Ignore(), Lamoule::Lamoule(), LogFactory::LogFactory(), Moderation::Moderation(), Module::Module(), Quotes::Quotes(), and Survey::Survey().

6.28.3.2 void Plugin::bindFunction (string *highlightedWord*, func_type *type*, string *symbole*, time_t *lastExec*, unsigned int *timeout*)

Register a plugin function.

Bind a plugin function. Called in plugin constructor, it initialise the function registration (before kernel). Used after plugin constructor has no effect (use kernel -> registerFunction instead)

Parameters:

highlightedWord Highlighted word (command), or time between two executions in seconds (for IN_-LOOP plugins)

type Function type

symbole function name to execute

lastExec last time plugin was executed

timeout Timeout for function (in seconds)

Definition at line 126 of file plugin.cpp.

References funcs, StructFunctionStorage::function, StructFunctionStorage::handle, StructFunctionStorage::highlightedWord, StructFunctionStorage::lastExec, StructFunctionStorage::object, StructFunctionStorage::symbole, StructFunctionStorage::timeout, and StructFunctionStorage::type.

Referenced by Admin::Admin(), Advertising::Advertising(), AntiFlood::AntiFlood(), BZRH::BZRH(), CTCP::CTCP(), DansTonChat::DansTonChat(), Fedorafr::Fedorafr(), FedoraProject::FedoraProject(), GameServer::GameServer(), Ignore::Ignore(), Infos::Infos(), IpConverting::IpConverting(), Lamoule::Lamoule(), LogFactory::LogFactory(), Magic8Ball::Magic8Ball(), Moderation::Moderation(), Module::Module(), Ping::Ping(), PluginSample::PluginSample(), PostConnect::PostConnect(), Quotes::Quotes(), RemoteControl::RemoteControl(), Slapme::Slapme(), Survey::Survey(), Tele::Tele(), Trad::Trad(), and UsersInfos::UsersInfos().

6.28.3.3 bool Plugin::checkMembers ()

Check if all members are given.

Check if all attributes have benn completed

Returns:

True if ok, else false

Definition at line 102 of file plugin.cpp.

References funcs, getAuthor(), getDescription(), getName(), and getVersion().

Referenced by BotKernel::loadPlugin().

6.28.3.4 string Plugin::getAuthor ()

Return author.

Get plugin author

Returns:

plugin author

Definition at line 57 of file plugin.cpp.

References author.

Referenced by checkMembers(), and moduleinfos().

6.28.3.5 string Plugin::getDescription ()

Return description.

Get plugin description

Returns:

plugin description

Definition at line 66 of file plugin.cpp.

References description.

Referenced by checkMembers(), and moduleinfos().

6.28.3.6 vector< StructFunctionStorage > Plugin::getFunctions ()

Return plugin functions.

Return shared functions

Returns:

Shared functions

Definition at line 93 of file plugin.cpp.

References funcs.

Referenced by BotKernel::loadPlugin().

6.28.3.7 void * Plugin::getHandle ()

Get plugin's handle.

Get plugin's handle

Returns:

Plugin's handle

Definition at line 144 of file plugin.cpp.

References handle.

Referenced by BotKernel::addCountDown(), and BotKernel::registerFunction().

6.28.3.8 string Plugin::getName ()

Return name.

Get plugin name

Returns:

plugin name

Definition at line 84 of file plugin.cpp.

References name.

Referenced by addsuperadmin(), addtempsuperadmin(), autoop(), autovoice(), ban(), banmask(), banned-Handler(), bzsearch(), checkBug(), checkMembers(), danstonchat(), deletekey(), delsuperadmin(), get-MyFirstNick(), help(), joinHandler(), kickHandler(), lamoule(), launchSurvey(), BotKernel::loadPlugin(), modeHandler(), modeHandlerProtect(), onEndOfMOTD(), planet(), player(), protectmodes(), protect-topic(), purifyFile(), randomKick(), rejoinChan(), RemoteControl::RemoteControl(), secondaryNick(), setconfvalue(), setSuperAdminPass(), testMsgTimestamp(), top5(), topicHandler(), toptotal(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), and wiki().

6.28.3.9 vector< string > Plugin::getRequirements ()

Get requirements list.

Get requirements list.

Returns:

Requirements list (vector)

Definition at line 172 of file plugin.cpp.

References requirements.

Referenced by BotKernel::loadPlugin().

6.28.3.10 string Plugin::getVersion ()

Return version.

Get plugin version

Returns:

plugin version

Definition at line 75 of file plugin.cpp.

References version.

Referenced by checkMembers(), and moduleinfos().

6.28.3.11 bool Plugin::requires (string *plugin*)

Tell if this plugin requires an other one.

Tell if this plugin requires an other one.

Parameters:

plugin Plugin to test

Returns:

true is this plugin requires the given one, else false

Definition at line 182 of file plugin.cpp.

References Tools::isInVector(), and requirements.

6.28.3.12 void Plugin::setHandle (void ∗ *handle*)

Set plugin's handle.

Set plugin's handle

Parameters:

handle Plugin's handle

Definition at line 153 of file plugin.cpp.

Referenced by BotKernel::loadPlugin().

6.28.4 Member Data Documentation

6.28.4.1 string Plugin::author `[protected]`

Plugin author.

Definition at line 86 of file plugin.h.

Referenced by Admin::Admin(), Advertising::Advertising(), AntiFlood::AntiFlood(), BZRH::BZRH(), CTCP::CTCP(), DansTonChat::DansTonChat(), Fedorafr::Fedorafr(), FedoraProject::FedoraProject(), GameServer::GameServer(), getAuthor(), Ignore::Ignore(), Infos::Infos(), IpConverting::IpConverting(), Lamoule::Lamoule(), LogFactory::LogFactory(), Magic8Ball::Magic8Ball(), Moderation::Moderation(), Module::Module(), Ping::Ping(), Plugin(), PluginSample::PluginSample(), PostConnect::PostConnect(), Quotes::Quotes(), RemoteControl::RemoteControl(), Slapme::Slapme(), Survey::Survey(), Tele::Tele(), Trad::Trad(), and UsersInfos::UsersInfos().

6.28.4.2 string Plugin::description `[protected]`

Plugin descritpion.

Definition at line 88 of file plugin.h.

Referenced by Admin::Admin(), Advertising::Advertising(), AntiFlood::AntiFlood(), BZRH::BZRH(), CTCP::CTCP(), DansTonChat::DansTonChat(), Fedorafr::Fedorafr(), FedoraProject::FedoraProject(), GameServer::GameServer(), getDescription(), Ignore::Ignore(), Infos::Infos(), IpConverting::IpConverting(), Lamoule::Lamoule(), LogFactory::LogFactory(), Magic8Ball::Magic8Ball(), Moderation::Moderation(), Module::Module(), Ping::Ping(), Plugin(), PluginSample::PluginSample(), PostConnect::PostConnect(), Quotes::Quotes(), RemoteControl::RemoteControl(), Slapme::Slapme(), Survey::Survey(), Tele::Tele(), Trad::Trad(), and UsersInfos::UsersInfos().

6.28.4.3 vector<StructFunctionStorage> Plugin::funcs `[protected]`

PLugins functions.

Definition at line 94 of file plugin.h.

Referenced by bindFunction(), checkMembers(), getFunctions(), and Plugin().

6.28.4.4 void∗ Plugin::handle [protected]

Stores plugin's handle.

Definition at line 96 of file plugin.h.

Referenced by getHandle(), and Plugin().

6.28.4.5 string Plugin::name [protected]

Plugin name.

Definition at line 92 of file plugin.h.

Referenced by Admin::Admin(), Advertising::Advertising(), AntiFlood::AntiFlood(), BZRH::BZRH(), CTCP::CTCP(), DansTonChat::DansTonChat(), Fedorafr::Fedorafr(), FedoraProject::FedoraProject(), GameServer::GameServer(), LogFactory::getLoggedChannels(), getName(), LogFactory::hasToBeLogged(), Ignore::Ignore(), Infos::Infos(), IpConverting::IpConverting(), Lamoule::Lamoule(), LogFactory::LogFactory(), Magic8Ball::Magic8Ball(), Moderation::Moderation(), Module::Module(), Ping::Ping(), Plugin(), PluginSample::PluginSample(), PostConnect::PostConnect(), Quotes::Quotes(), RemoteControl::RemoteControl(), Slapme::Slapme(), Survey::Survey(), Tele::Tele(), Trad::Trad(), and UsersInfos::UsersInfos().

6.28.4.6 vector<string> Plugin::requirements [protected]

stores plugins names that are required for the plugin

Definition at line 98 of file plugin.h.

Referenced by addRequirement(), getRequirements(), Plugin(), and requires().

6.28.4.7 string Plugin::version [protected]

Plugin version.

Definition at line 90 of file plugin.h.

Referenced by Admin::Admin(), Advertising::Advertising(), AntiFlood::AntiFlood(), BZRH::BZRH(), CTCP::CTCP(), DansTonChat::DansTonChat(), Fedorafr::Fedorafr(), FedoraProject::FedoraProject(), GameServer::GameServer(), getVersion(), Ignore::Ignore(), Infos::Infos(), IpConverting::IpConverting(), Lamoule::Lamoule(), LogFactory::LogFactory(), Magic8Ball::Magic8Ball(), Moderation::Moderation(), Module::Module(), Ping::Ping(), Plugin(), PluginSample::PluginSample(), PostConnect::PostConnect(), Quotes::Quotes(), RemoteControl::RemoteControl(), Slapme::Slapme(), Survey::Survey(), Tele::Tele(), Trad::Trad(), and UsersInfos::UsersInfos().

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

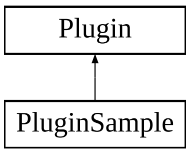
- src/plugin.h
- src/plugin.cpp

6.29 PluginSample Class Reference

Plugin class example.

```
#include <pluginsample.h>
```

Inheritance diagram for PluginSample::



Public Member Functions

- PluginSample (BotKernel ∗)

  Constructor.

6.29.1 Detailed Description

Plugin class example.

This class provides a simple plugin example

Definition at line 41 of file pluginsample.h.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 PluginSample::PluginSample (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file pluginsample.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

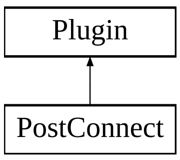
- src/plugins/pluginsample.h
- src/plugins/pluginsample.cpp

6.30 PostConnect Class Reference

Afer connect plugin.

`#include <postconnect.h>`

Inheritance diagram for PostConnect::



Public Member Functions

- PostConnect (BotKernel ∗)

  Constructor.

- unsigned int getNickRetreiveAttempts ()

  get nick retreive attemtps

- void bumpNickRetreiveAttempts ()

  Increase nick retreive attempts.

- void resetNickRetreiveAttempts ()

  Rest nick retreive attempts.

Private Attributes

- unsigned int nickRetreiveAttempts

  stores attempts number

6.30.1 Detailed Description

Afer connect plugin.

This class plugin mamage all actions performed after server connection (join channels, auth to bots, and raw actions)

Definition at line 39 of file postconnect.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 PostConnect::PostConnect (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file postconnect.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_TYPE_HANDLER, Plugin::name, resetNickRetreiveAttempts(), and Plugin::version.

6.30.3 Member Function Documentation

6.30.3.1 void PostConnect::bumpNickRetreiveAttempts ()

Increase nick retreive attempts.

Increase nick take attempts

Definition at line 59 of file postconnect.cpp.

References nickRetreiveAttempts.

Referenced by getMyFirstNick().

6.30.3.2 unsigned int PostConnect::getNickRetreiveAttempts ()

get nick retreive attemtps

get nick take attemtps

Returns:

attempts number

Definition at line 51 of file postconnect.cpp.

References nickRetreiveAttempts.

Referenced by getMyFirstNick().

6.30.3.3 void PostConnect::resetNickRetreiveAttempts ()

Rest nick retreive attempts.

Rest nick take attempts

Definition at line 67 of file postconnect.cpp.

References nickRetreiveAttempts.

Referenced by nick_changed(), and PostConnect().

6.30.4 Member Data Documentation

6.30.4.1 unsigned int PostConnect::nickRetreiveAttempts `[private]`

stores attempts number

Definition at line 43 of file postconnect.h.

Referenced by bumpNickRetreiveAttempts(), getNickRetreiveAttempts(), and resetNickRetreiveAttempts().

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

- src/plugins/postconnect.h
- src/plugins/postconnect.cpp

6.31 pPlugin Struct Reference

Plugin object and header storage.

`#include <plugin.h>`

Public Attributes

- string name
- void ∗ handle
- Plugin ∗ object
- plugin_constructor creator
- plugin_destructor destructor

6.31.1 Detailed Description

Plugin object and header storage.

Definition at line 53 of file plugin.h.

6.31.2 Member Data Documentation

6.31.2.1 plugin_constructor pPlugin::creator

Definition at line 58 of file plugin.h.

Referenced by BotKernel::loadPlugin().

6.31.2.2 plugin_destructor pPlugin::destructor

Definition at line 59 of file plugin.h.

Referenced by BotKernel::loadPlugin().

6.31.2.3 void∗ pPlugin::handle

Definition at line 56 of file plugin.h.

Referenced by BotKernel::loadPlugin(), and reauth().

6.31.2.4 string pPlugin::name

Definition at line 55 of file plugin.h.

Referenced by BotKernel::loadPlugin().

6.31.2.5 Plugin∗ pPlugin::object

Definition at line 57 of file plugin.h.

Referenced by addad(), addIgnore(), adinfos(), ban(), Moderation::checkAccess(), Moderation::checkMode(), LogFactory::cleanLogs(), clearOutBans(), delad(), deleteplayer(), delIgnore(), delQuote(), Moderation::getChanUsersList(), LogFactory::getLoggedChannels(), Moderation::hasOpPrivileges(), ignoreList(), increase(), invite(), isIgnored(), joinHandler(), kickHandler(), lamoule(), listads(), listlibs(), listmodules(), load(), loadnocheck(), BotKernel::loadPlugin(), modeHandler(), modeHandlerProtect(), moduleinfos(), myThread(), nextscore(), partHandler(), protectmodes(), protecttopic(), quitHandler(), quoteInfos(), reauth(), reloadfas(), stopSurvey(), testMsgTimestamp(), topicHandler(), topicJoin(), unload(), unloadnocheck(), unprotectmodes(), and unprotecttopic().

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

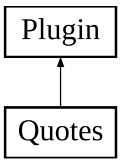
- src/plugin.h

6.32 Quotes Class Reference

Quotes management (storage and access).

```
#include <quotes.h>
```

Inheritance diagram for Quotes::



Public Member Functions

- Quotes (BotKernel $*$)

  Constructor.

- void addQuote (string, string)

  Add a quote.

- string getQuote (unsigned int)

  Return a quote.

- string getRandomQuote ()

  Return a random quote.

- vector< string > searchQuote (string)

  Search quotes according to a pattern.

- bool delQuote (unsigned int)

  Delete a quote.

- string getLastQuote ()

  Return last inserted quote.

- string quoteInfos (unsigned int)

  Return informations aboute a quote.

Private Member Functions

- unsigned int getNbChilds (TiXmlNode $*$)

  Return nodes's child's number.

Private Attributes

- TiXmlDocument ∗ doc

  Represent the xml document.

- TiXmlNode ∗ root

  Represent documents's root.

- unsigned int nbQuotes

  Quotes number.

6.32.1 Detailed Description

Quotes management (storage and access).

This plugin stores quotes in a XML file and give access to them

Definition at line 50 of file quotes.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 Quotes::Quotes (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file quotes.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, doc, BotKernel::getDatasDir(), getNbChilds(), IN_COMMAND_HANDLER, Plugin::name, nbQuotes, root, and Plugin::version.

6.32.3 Member Function Documentation

6.32.3.1 void Quotes::addQuote (string *host*, string *quote*)

Add a quote.

Add a quote in the XML tree

Parameters:

host Quoter's host

quote Quote's text

Definition at line 88 of file quotes.cpp.

References doc, nbQuotes, and root.

Referenced by addQuote().

6.32.3.2 bool Quotes::delQuote (unsigned int *index*)

Delete a quote.

Delete a quote

Parameters:

index Quote index (start at 1)

Returns:

True if deleted, else false

Definition at line 182 of file quotes.cpp.

References doc, nbQuotes, and root.

Referenced by delQuote().

6.32.3.3 string Quotes::getLastQuote ()

Return last inserted quote.

Return last inserted quote

Returns:

last inserted quote text

Definition at line 201 of file quotes.cpp.

References getQuote(), and nbQuotes.

Referenced by lastQuote().

6.32.3.4 unsigned int Quotes::getNbChilds (TiXmlNode ∗ *node*) `[private]`

Return nodes's child's number.

Return child number for a node

Parameters:

node Node that we want child's number

Returns:

Node's child number

Definition at line 66 of file quotes.cpp.

Referenced by Quotes().

6.32.3.5 string Quotes::getQuote (unsigned int *index*)

Return a quote.

Return a quote

Parameters:

index Quote's index (start at 1)

Returns:

Quote's text

Definition at line 109 of file quotes.cpp.

References doc, Tools::intToStr(), and nbQuotes.

Referenced by getLastQuote(), getRandomQuote(), and quote().

6.32.3.6 string Quotes::getRandomQuote ()

Return a random quote.

Return a random quote

Returns:

Random quote text

Definition at line 130 of file quotes.cpp.

References getQuote(), nbQuotes, and Tools::random().

Referenced by quote().

6.32.3.7 string Quotes::quoteInfos (unsigned int *index*)

Return informations aboute a quote.

Return informations (date en quoter) about a quote

Parameters:

index Quote index (start at 1)

Returns:

Quote's informations

Definition at line 211 of file quotes.cpp.

References doc.

Referenced by quoteInfos().

6.32.3.8 vector< string > Quotes::searchQuote (string *pattern*)

Search quotes according to a pattern.

Return quotes a quote matching to a given pattern an quotes numbers matching too

Parameters:

pattern Pattern used for search

Returns:

A quote and all quotes numbers matching

Definition at line 145 of file quotes.cpp.

References doc, Tools::intToStr(), nbQuotes, Tools::random(), and Tools::to_lower().

Referenced by searchQuote().

6.32.4 Member Data Documentation

6.32.4.1 TiXmlDocument∗ Quotes::doc `[private]`

Represent the xml document.

Definition at line 54 of file quotes.h.

Referenced by addQuote(), delQuote(), getQuote(), quoteInfos(), Quotes(), and searchQuote().

6.32.4.2 unsigned int Quotes::nbQuotes `[private]`

Quotes number.

Definition at line 58 of file quotes.h.

Referenced by addQuote(), delQuote(), getLastQuote(), getQuote(), getRandomQuote(), Quotes(), and searchQuote().

6.32.4.3 TiXmlNode∗ Quotes::root `[private]`

Represent documents's root.

Definition at line 56 of file quotes.h.

Referenced by addQuote(), delQuote(), and Quotes().

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

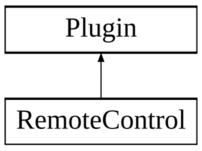
- src/plugins/quotes.h
- src/plugins/quotes.cpp

6.33 RemoteControl Class Reference

Plugin that allow remote TCP control.

```
#include <remotecontrol.h>
```

Inheritance diagram for RemoteControl::



Public Member Functions

- RemoteControl (BotKernel ∗)

  Constructor.

- ~RemoteControl ()

  Destructor.

- void setSocketList (struct timeval ∗, fd_set ∗)

  Prepare FD for select and return highest sock.

- int manageNewConnection (int)

  manage incomming connections

- void ∗ tcpServer (BotKernel ∗)

  TCP server to receive connections.

Private Attributes

- CPPThread ∗ pt

  Thread object.

- int sockfd

  Socket used to accept connections.

- int ∗ clients

  Array to store connections.

- unsigned int MYPORT

  Port used for connection.

- unsigned int MAXCLIENTS

  Maximum clients that will be able to connect.

Static Private Attributes

- static const int BACKLOG = 5

  Listen backlog.

- static const int MAXDATASIZE = 100

  Max data size fo communication.

6.33.1 Detailed Description

Plugin that allow remote TCP control.

This plugins make the bot listen on a port. A client can connect to the bot with a telnet command and can then control the bot by sending commands

Definition at line 46 of file remotecontrol.h.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 RemoteControl::RemoteControl (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 36 of file remotecontrol.cpp.

References Plugin::author, Plugin::bindFunction(), clients, Plugin::description, CPPThread::exec(), BotKernel::getCONFF(), Plugin::getName(), ConfigurationFile::getValue(), IN_FIRST_WORD, MAXCLIENTS, MYPORT, myThread(), Plugin::name, pt, Tools::strToUnsignedInt(), Plugin::version, and WARNING.

6.33.2.2 RemoteControl::∼RemoteControl ()

Destructor.

Destructor

Definition at line 57 of file remotecontrol.cpp.

References clients, MAXCLIENTS, pt, and sockfd.

6.33.3 Member Function Documentation

6.33.3.1 int RemoteControl::manageNewConnection (int *sock*)

manage incomming connections

Manage new connections. If there is enough room connection will be stored, else, connexion have to be closed

Parameters:

sock Incomming client socket

Returns:

Client slot (-1 if no room)

Definition at line 181 of file remotecontrol.cpp.

References clients, and MAXCLIENTS.

Referenced by tcpServer().

6.33.3.2 void RemoteControl::setSocketList (struct timeval ∗ *tv*, fd_set ∗ *readfds*)

Prepare FD for select and return highest sock.

Construct a socket list to be used by select. Sockets are added to fd list

Parameters:

tv timeval structure

readfds fd_set will store sockets

Definition at line 162 of file remotecontrol.cpp.

References clients, MAXCLIENTS, and sockfd.

Referenced by tcpServer().

6.33.3.3 void ∗ RemoteControl::tcpServer (BotKernel ∗ *b*)

TCP server to receive connections.

TCP server to receive connections. This function will wait for a connection and talk to the kernel to execute command requested by the user connected

Parameters:

b Pointer on the kernel

Precondition:

This function must be called in a threaded function to don't block the bot

Returns:

NOTHING. Juste a NULL pointer

Definition at line 77 of file remotecontrol.cpp.

References BACKLOG, clients, ERROR, BotKernel::getSysLog(), INFO, Tools::intToStr(), LogFile::log(), manageNewConnection(), MAXCLIENTS, MAXDATASIZE, MYPORT, setSocketList(), sockfd, and WARNING.

Referenced by myThread().

6.33.4 Member Data Documentation

6.33.4.1 const int RemoteControl::BACKLOG = 5 `[static, private]`

Listen backlog.

Definition at line 60 of file remotecontrol.h.

Referenced by tcpServer().

6.33.4.2 int* RemoteControl::clients [private]

Array to store connections.

Definition at line 54 of file remotecontrol.h.

Referenced by manageNewConnection(), RemoteControl(), setSocketList(), tcpServer(), and ∼RemoteControl().

6.33.4.3 unsigned int RemoteControl::MAXCLIENTS [private]

Maximum clients that will be able to connect.

Definition at line 58 of file remotecontrol.h.

Referenced by manageNewConnection(), RemoteControl(), setSocketList(), tcpServer(), and ∼RemoteControl().

6.33.4.4 const int RemoteControl::MAXDATASIZE = 100 [static, private]

Max data size fo communication.

Definition at line 62 of file remotecontrol.h.

Referenced by tcpServer().

6.33.4.5 unsigned int RemoteControl::MYPORT [private]

Port used for connection.

Definition at line 56 of file remotecontrol.h.

Referenced by RemoteControl(), and tcpServer().

6.33.4.6 CPPThread* RemoteControl::pt [private]

Thread object.

Definition at line 50 of file remotecontrol.h.

Referenced by RemoteControl(), and ∼RemoteControl().

6.33.4.7 int RemoteControl::sockfd [private]

Socket used to accept connections.

Definition at line 52 of file remotecontrol.h.

Referenced by setSocketList(), tcpServer(), and ∼RemoteControl().

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

- src/plugins/remotecontrol.h

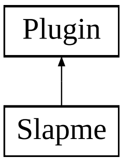
- src/plugins/remotecontrol.cpp

6.34 Slapme Class Reference

Plugin used to slap users.

`#include <slapme.h>`

Inheritance diagram for Slapme::



Public Member Functions

- Slapme (BotKernel ∗)

  Constructor.

6.34.1 Detailed Description

Plugin used to slap users.

This plugins allow people to make the bo slap them after a countdown is elapsed

Definition at line 41 of file slapme.h.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 Slapme::Slapme (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file slapme.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

- src/plugins/slapme.h
- src/plugins/slapme.cpp

6.35 Socket Class Reference

Class that manage the connection with the server.

`#include <socket.h>`

Public Member Functions

- Socket ()

  Constructor.

- ∼Socket ()

  Destructor.

- bool sendStr (string)

  Send a string to the server.

- bool closeSock ()

  Close the socket.

- bool connectSock (int, string, string)

  Connect the socket to the server.

- bool getState ()

  Get connection state.

- string receive ()

  Receive a string from the server.

Private Attributes

- bool state

  Socket state.

- int mySock

  Socket handle.

6.35.1 Detailed Description

Class that manage the connection with the server.

This class manage the connection with the irc server. It uses a socket for communication

Definition at line 53 of file socket.h.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 Socket::Socket ()

Constructor.

Class constructor

Definition at line 34 of file socket.cpp.

References mySock, and state.

6.35.2.2 Socket::~Socket ()

Destructor.

Clas destructor

Definition at line 43 of file socket.cpp.

References closeSock().

6.35.3 Member Function Documentation

6.35.3.1 bool Socket::closeSock ()

Close the socket.

Close the socket

Returns:

true if socket well closed, else false

Definition at line 95 of file socket.cpp.

References mySock, and state.

Referenced by receive(), BotKernel::reconnect(), BotKernel::run(), and ~Socket().

6.35.3.2 bool Socket::connectSock (int *port*, string *serverName*, string *dedicatedIP*)

Connect the socket to the server.

Connect the socket to a server The socket can be "block type" and use a dedicated IP adress

Parameters:

port	Server port
serverName	Server IP or hostname
dedicatedIP	Dedicated IP used for connextion. Empty if not used

Returns:

True if connection OK, else false

Definition at line 56 of file socket.cpp.

References mySock, and state.

Referenced by BotKernel::connect(), danstonchat(), planet(), tele(), trad(), and wiki().

6.35.3.3 bool Socket::getState ()

Get connection state.

Get connection state

Returns:

true if the socket is connected, else false

Definition at line 110 of file socket.cpp.

References state.

Referenced by BotKernel::run().

6.35.3.4 string Socket::receive ()

Receive a string from the server.

Receive a string from the server

Returns:

Received string? Empty if nothing received

Definition at line 119 of file socket.cpp.

References closeSock(), mySock, and state.

Referenced by danstonchat(), planet(), BotKernel::run(), tele(), trad(), and wiki().

6.35.3.5 bool Socket::sendStr (string *strData*)

Send a string to the server.

Send a string to the server

Parameters:

strData String to send

Returns:

true if send OK, else false

Definition at line 153 of file socket.cpp.

References mySock.

Referenced by danstonchat(), planet(), BotKernel::send(), tele(), trad(), and wiki().

6.35.4 Member Data Documentation

6.35.4.1 int Socket::mySock `[private]`

Socket handle.

Definition at line 74 of file socket.h.

Referenced by closeSock(), connectSock(), receive(), sendStr(), and Socket().

6.35.4.2 bool Socket::state `[private]`

Socket state.

Definition at line 72 of file socket.h.

Referenced by closeSock(), connectSock(), getState(), receive(), and Socket().

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

- src/socket.h
- src/socket.cpp

6.36 struct_survey Struct Reference

Plugin object and header storage.

```
#include <survey.h>
```

Public Attributes

- string channel
- string question
- unsigned int time
- vector< string > answers
- vector< int > results
- vector< string > voters
- vector< plugin_function > functions
- plugin_function countDown

6.36.1 Detailed Description

Plugin object and header storage.

Definition at line 38 of file survey.h.

6.36.2 Member Data Documentation

6.36.2.1 vector<string> struct_survey::answers

Definition at line 43 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.2 string struct_survey::channel

Definition at line 40 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.3 plugin_function struct_survey::countDown

Definition at line 47 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.4 vector<plugin_function> struct_survey::functions

Definition at line 46 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.5 string struct_survey::question

Definition at line 41 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.6 vector<int> struct_survey::results

Definition at line 44 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.7 unsigned int struct_survey::time

Definition at line 42 of file survey.h.

Referenced by Survey::launchSurvey().

6.36.2.8 vector<string> struct_survey::voters

Definition at line 45 of file survey.h.

Referenced by Survey::launchSurvey().

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

- src/plugins/survey.h

6.37 StructFunctionStorage Struct Reference

Plugin function storage.

```
#include <plugin.h>
```

Public Attributes

- void ∗ handle
- string highlightedWord
- Plugin ∗ object
- func_type type
- string symbole
- plugin_function function
- time_t lastExec
- unsigned int timeout
- bool back

6.37.1 Detailed Description

Plugin function storage.

Definition at line 63 of file plugin.h.

6.37.2 Member Data Documentation

6.37.2.1 bool StructFunctionStorage::back

Definition at line 73 of file plugin.h.

Referenced by BotKernel::executeFunction(), and threadFunc().

6.37.2.2 plugin_function StructFunctionStorage::function

Definition at line 70 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), BotKernel::loadPlugin(), BotKernel::registerFunction(), BotKernel::storeFunction(), and threadFunc().

6.37.2.3 void∗ StructFunctionStorage::handle

Definition at line 65 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), BotKernel::loadPlugin(), and BotKernel::registerFunction().

6.37.2.4 string StructFunctionStorage::highlightedWord

Definition at line 66 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), and BotKernel::registerFunction().

6.37.2.5 time_t StructFunctionStorage::lastExec

Definition at line 71 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), and BotKernel::registerFunction().

6.37.2.6 Plugin* StructFunctionStorage::object

Definition at line 67 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), BotKernel::registerFunction(), and threadFunc().

6.37.2.7 string StructFunctionStorage::symbole

Definition at line 69 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), BotKernel::executeFunction(), BotKernel::loadPlugin(), BotKernel::registerFunction(), and BotKernel::storeFunction().

6.37.2.8 unsigned int StructFunctionStorage::timeout

Definition at line 72 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), BotKernel::executeFunction(), and BotKernel::registerFunction().

6.37.2.9 func_type StructFunctionStorage::type

Definition at line 68 of file plugin.h.

Referenced by BotKernel::addCountDown(), Plugin::bindFunction(), BotKernel::executeFunction(), BotKernel::registerFunction(), and BotKernel::storeFunction().

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

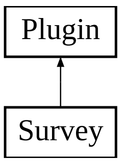
- src/plugin.h

6.38 Survey Class Reference

This plugin manages surveys.

`#include <survey.h>`

Inheritance diagram for Survey::



Public Member Functions

- Survey (BotKernel ∗)

  Constructor.
- bool launchSurvey (string, string, unsigned int, vector< string >)

  Launch a survey.
- bool stopSurvey (string)

  Stop a survey.
- bool vote (string, string, string)

  Vote.
- vector< string > finishSurvey (string)

  Finish a survey.
- vector< plugin_function > getSurveyFunctions (string)

  Get survey's functions.
- bool setSurveyFunctions (string, vector< plugin_function >)

  Set survey's functions.
- plugin_function getCountDown (string)

  get countdown pointer
- bool setCountDown (string, plugin_function)

  set countdown pointer

Private Member Functions

- int getAnswerId (vector< string >, string)

  Get an answer's id.
- bool surveyRunning (string)

Test if a survey is running.

Private Attributes

- vector< struct_survey > surveys

  Surveys storage.

6.38.1 Detailed Description

This plugin manages surveys.

This plugin manages surveys

Definition at line 55 of file survey.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Survey::Survey (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file survey.cpp.

References Plugin::addRequirement(), Plugin::author, Plugin::bindFunction(), Plugin::description, IN_-COMMAND_HANDLER, Plugin::name, surveys, and Plugin::version.

6.38.3 Member Function Documentation

6.38.3.1 vector< string > Survey::finishSurvey (string *channel*)

Finish a survey.

Finish the survey

Parameters:

channel Channel where finish the survey

Returns:

A vector containing results

Definition at line 80 of file survey.cpp.

References Tools::intToStr(), and surveys.

Referenced by endSurvey().

6.38.3.2 int Survey::getAnswerId (vector< string > *answers*, string *answer*) `[private]`

Get an answer's id.

Get an answer's id

Parameters:

answers Answers list

answer Answer to check

Returns:

Answer's id (-1 if not present)

Definition at line 226 of file survey.cpp.

Referenced by vote().

6.38.3.3 plugin_function Survey::getCountDown (string *channel*)

get countdown pointer

Get survey's countdown

Parameters:

channel Channel's survey

Returns:

countdown pointer

Definition at line 193 of file survey.cpp.

References surveys.

Referenced by stopSurvey().

6.38.3.4 vector< plugin_function > Survey::getSurveyFunctions (string *channel*)

Get survey's functions.

Get survey's functions

Parameters:

channel Channel's survey

Returns:

A vector containing functions

Definition at line 161 of file survey.cpp.

References surveys.

Referenced by endSurvey(), and stopSurvey().

6.38.3.5 bool Survey::launchSurvey (string *channel*, string *question*, unsigned int *time*, vector< string > *answers*)

Launch a survey.

Launch a survey on a channel

Parameters:

channel Channel where launch the survey

question Survey's question

time Survey length (in seconds)

answers Possible answers

Returns:

true is the survey has been launched, else false

Definition at line 54 of file survey.cpp.

References struct_survey::answers, struct_survey::channel, struct_survey::countDown, struct_-survey::functions, struct_survey::question, struct_survey::results, surveyRunning(), surveys, struct_-survey::time, and struct_survey::voters.

Referenced by launchSurvey().

6.38.3.6 bool Survey::setCountDown (string *channel*, plugin_function *function*)

set countdown pointer

Set survey's countdown

Parameters:

channel Channel's survey

function Countdown function

Returns:

True if "set" is OK, else false

Definition at line 209 of file survey.cpp.

References surveys.

Referenced by launchSurvey().

6.38.3.7 bool Survey::setSurveyFunctions (string *channel*, vector< plugin_function > *functions*)

Set survey's functions.

Set survey's functions

Parameters:

channel Channel's survey

functions A vector functions

Returns:

True if "set" is OK, else false

Definition at line 177 of file survey.cpp.

References surveys.

Referenced by launchSurvey().

6.38.3.8 bool Survey::stopSurvey (string *channel*)

Stop a survey.

Stop a survey on a channel

Parameters:

channel Channel where stop the survey

Returns:

true if the survey has been stopped, else false

Definition at line 129 of file survey.cpp.

References surveys.

Referenced by launchSurvey(), and stopSurvey().

6.38.3.9 bool Survey::surveyRunning (string *channel*) `[private]`

Test if a survey is running.

Test if a survey is running on a channel

Parameters:

channel Channel where test if a survey is running

Returns:

true if a channel is running, else false

Definition at line 147 of file survey.cpp.

References surveys.

Referenced by launchSurvey().

6.38.3.10 bool Survey::vote (string *channel*, string *nick*, string *answer*)

Vote.

Register a vote for a user

Parameters:

channel Channel where the user votes

nick User's nick

answer User's answer

Returns:

true if the vote has been registered, else false

Definition at line 104 of file survey.cpp.

References getAnswerId(), Tools::isInVector(), and surveys.

Referenced by vote().

6.38.4 Member Data Documentation

6.38.4.1 vector<struct_survey> Survey::surveys `[private]`

Surveys storage.

Definition at line 59 of file survey.h.

Referenced by finishSurvey(), getCountDown(), getSurveyFunctions(), launchSurvey(), setCountDown(), setSurveyFunctions(), stopSurvey(), Survey(), surveyRunning(), and vote().

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

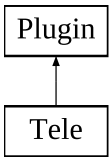
- src/plugins/survey.h
- src/plugins/survey.cpp

6.39 Tele Class Reference

Display french TV program.

`#include <tele.h>`

Inheritance diagram for Tele::



Public Member Functions

- Tele (BotKernel $*$)

  Constructor.

6.39.1 Detailed Description

Display french TV program.

Display french TV program

Definition at line 41 of file tele.h.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 Tele::Tele (BotKernel $*$ *b*)

Constructor.

Constructor

Definition at line 34 of file tele.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

- src/plugins/tele.h
- src/plugins/tele.cpp

6.40 threadInfos Struct Reference

Stores thread informations.

`#include <cppthread.h>`

Public Attributes

- threadProcess process
- void * args
- bool running
- bool finished

6.40.1 Detailed Description

Stores thread informations.

Definition at line 36 of file cppthread.h.

6.40.2 Member Data Documentation

6.40.2.1 void* threadInfos::args

Definition at line 38 of file cppthread.h.

Referenced by CPPThread::exec(), and CPPThread::threadStartup().

6.40.2.2 bool threadInfos::finished

Definition at line 40 of file cppthread.h.

Referenced by CPPThread::CPPThread(), CPPThread::isFinished(), and CPPThread::threadStartup().

6.40.2.3 threadProcess threadInfos::process

Definition at line 37 of file cppthread.h.

Referenced by CPPThread::exec(), and CPPThread::threadStartup().

6.40.2.4 bool threadInfos::running

Definition at line 39 of file cppthread.h.

Referenced by CPPThread::CPPThread(), CPPThread::isRunning(), CPPThread::terminate(), and CPPThread::threadStartup().

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

- src/cppthread.h

6.41 ThreadParams Struct Reference

Thread information storage.

```
#include <botkernel.h>
```

Public Attributes

- StructFunctionStorage ∗ function
- Message ∗ msg
- BotKernel ∗ b
- sem_t ∗ sem

6.41.1 Detailed Description

Thread information storage.

Definition at line 71 of file botkernel.h.

6.41.2 Member Data Documentation

6.41.2.1 BotKernel∗ ThreadParams::b

Definition at line 74 of file botkernel.h.

Referenced by BotKernel::executeFunction(), and threadFunc().

6.41.2.2 StructFunctionStorage∗ ThreadParams::function

Definition at line 72 of file botkernel.h.

Referenced by BotKernel::executeFunction(), and threadFunc().

6.41.2.3 Message∗ ThreadParams::msg

Definition at line 73 of file botkernel.h.

Referenced by BotKernel::executeFunction(), and threadFunc().

6.41.2.4 sem_t∗ ThreadParams::sem

Definition at line 75 of file botkernel.h.

Referenced by BotKernel::executeFunction(), and threadFunc().

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

- src/botkernel.h

6.42 Tools Class Reference

Class that provides tools for programmation.

`#include <tools.h>`

Public Member Functions

- Tools ()

  Constructor.

- ~Tools ()

  Destructor.

Static Public Member Functions

- static string asciiToHexa (string)

  Convert an ascii string to an hexadecimal string.

- static string hexaToAscii (string)

  Convert an hexadecimal string to an ascii string.

- static string intToStr (int)

  Convert a int to a string.

- static string doubleToStr (double)

  Convert a double to a string.

- static double strToDouble (string)

  Convert a string to a double.

- static int strToInt (string)

  Convert a string to a int.

- static unsigned int strToUnsignedInt (string)

  Convert a string to an unsigned int.

- static unsigned int strtimeToSeconds (string)

  Convert a string time to seconds.

- static string to_lower (string)

  Convert a string to lower case.

- static string to_upper (string)

  Convert a string to upper case.

- static int random (int min, int max)

  Give a random number.

- static string vectorToString (vector< string >, string, unsigned int start=0)
  Convert a vector to a string.
- static vector< string > stringToVector (string, string, unsigned int start=0)
  Convert a string to a vector.
- static vector< string > gatherVectorElements (vector< string >, string, unsigned int)
  Gather vector elements.
- static string escapeChar (string, char)
  Escape a char in a string.
- static void log (string, string, bool timestamp=true, bool truncate=false)
  Log a string in a file.
- static string urlencode (string)
  Encode a string to a URL format.
- static string clearAccents (string)
  Clear accents from a sentence.
- static string cleanHTML (string)
  Clean HTML code in a string.
- static bool isInVector (vector< string >, string)
  Tell if a string is in a vector.
- static void delStrFromVector (vector< string > ∗, string)
  Withdraw a string from a vector.
- static string parseQ3Colors (string)
  Parse Quake colors from a string.
- static bool ircMaskMatch (string, string)
  Tell if an irc host match to a mask.
- static int masksMatch (char ∗, char ∗)
  Tell if two masks match.
- static bool copyFile (string, string)
  Copy a file.

6.42.1 Detailed Description

Class that provides tools for programmation.

Provide different static methods for regular treatment. Those methods can be used everywhere in the code an doesn't need an object instantiation.

Definition at line 47 of file tools.h.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 Tools::Tools ()

Constructor.

Constructor

Definition at line 36 of file tools.cpp.

6.42.2.2 Tools::∼Tools ()

Destructor.

Destructor

Definition at line 44 of file tools.cpp.

6.42.3 Member Function Documentation

6.42.3.1 string Tools::asciiToHexa (string *asciiStr*) `[static]`

Convert an ascii string to an hexadecimal string.

Convert an ascii string to an hexadecimal string

Parameters:

asciiStr String to convert

Returns:

Hexadecimal string

Definition at line 54 of file tools.cpp.

6.42.3.2 string Tools::cleanHTML (string *str*) `[static]`

Clean HTML code in a string.

Clean all HTML chars in a string : Replace accents marks and delete font marks

Parameters:

str String with no html tags

Definition at line 455 of file tools.cpp.

Referenced by danstonchat(), BZRH::getBugInfos(), BZRH::searchBugs(), tele(), and trad().

6.42.3.3 string Tools::clearAccents (string *str*) `[static]`

Clear accents from a sentence.

Clear accents from a sentence

Parameters:

str String to clear

Returns:

String with no accents

Definition at line 433 of file tools.cpp.

Referenced by danstonchat(), and tele().

6.42.3.4 static bool Tools::copyFile (string, string) `[static]`

Copy a file.

6.42.3.5 void Tools::delStrFromVector (vector< string > * *v*, string *str*) `[static]`

Withdraw a string from a vector.

Delete a string from a vector

Parameters:

v Vector<string> pointer witch must be delete a string
str string to delete

Definition at line 102 of file tools.cpp.

Referenced by unautoop(), unautovoice(), unprotectmodes(), and unprotecttopic().

6.42.3.6 string Tools::doubleToStr (double *number*) `[static]`

Convert a double to a string.

Convert a double to a string

Parameters:

number double number to convert

Returns:

String convertion result

Definition at line 134 of file tools.cpp.

Referenced by Lamoule::get5first(), Lamoule::getInfosPlayer(), and player().

6.42.3.7 string Tools::escapeChar (string *str*, char *c*) `[static]`

Escape a char in a string.

Escape a char in a string

Parameters:

str String containing chars

c Char to escape in the string

Returns:

Initial string with escaped chars

Definition at line 358 of file tools.cpp.

6.42.3.8 vector< string > Tools::gatherVectorElements (vector< string > *v*, string *separator*, unsigned int *length*) `[static]`

Gather vector elements.

Gather vector's elements to make a vector with less elements

Parameters:

v vector for witch gather elements

separator separator for elements

length vector's element length

Returns:

The new vector with gathered elements

Definition at line 336 of file tools.cpp.

Referenced by banlist(), chanlev(), commandsStatus(), listlibs(), listmodules(), onEndOfMOTD(), and superadminlist().

6.42.3.9 string Tools::hexaToAscii (string *hexaStr*) `[static]`

Convert an hexadecimal string to an ascii string.

Convert an hexadecimal string to an ascii string

Parameters:

hexaStr String to convert

Returns:

Ascii string

Definition at line 68 of file tools.cpp.

6.42.3.10 string Tools::intToStr (int *number*) `[static]`

Convert a int to a string.

Convert a int to a string

Parameters:

number int number to convert

Returns:

String convertion result

Definition at line 122 of file tools.cpp.

Referenced by addad(), Advertising::addAdvertise(), BotKernel::addCountDown(), Lamoule::addPlayer(), Survey::finishSurvey(), Lamoule::get5first(), Moderation::getBanList(), Ignore::getIgnoreList(), Lamoule::getInfosPlayer(), getnbcountdowns(), Quotes::getQuote(), hl(), Lamoule::increaseScore(), lamoule(), launchSurvey(), BotKernel::loadPlugins(), online(), q3(), BotKernel::run(), BZRH::searchBugs(), Quotes::searchQuote(), slapme(), Admin::superAdminList(), RemoteControl::tcpServer(), uptime(), warsow(), and whoami().

6.42.3.11 bool Tools::ircMaskMatch (string *request*, string *mask*) `[static]`

Tell if an irc host match to a mask.

Check if an IRC host match to a mask

Parameters:

request Irc host

mask mask

Returns:

true if host match to regex, else false

Definition at line 521 of file tools.cpp.

Referenced by banmask(), Admin::getUserLevel(), Moderation::isBanned(), Ignore::isIgnored(), and Admin::isSuperAdmin().

6.42.3.12 bool Tools::isInVector (vector< string > *v*, string *str*) `[static]`

Tell if a string is in a vector.

Check if a string is in a vector string

Parameters:

v vector to check

str string to check

Returns:

true if present, else false

Definition at line 83 of file tools.cpp.

Referenced by autoop(), autovoice(), LogFactory::cleanLogs(), Moderation::clearOutBans(), Admin::commandOK(), ConfigurationFile::getValue(), LogFactory::hasToBeLogged(), joinHandler(), launchSurvey(), modeHandlerProtect(), protectmodes(), protecttopic(), Plugin::requires(), topicHandler(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), and Survey::vote().

6.42.3.13 void Tools::log (string *fileName*, string *str*, bool *timestamp* = `true`, bool *truncate* = `false`) `[static]`

Log a string in a file.

Log an event in a file, with timestamp

Parameters:

fileName Log file

str Log event

timestamp tell if a timestamp must appear in the log line

truncate tell if the file must be truncated

Definition at line 383 of file tools.cpp.

Referenced by BotKernel::run().

6.42.3.14 int Tools::masksMatch (char ∗ *str1*, char ∗ *str2*) `[static]`

Tell if two masks match.

Check if two masks match Thanks to "BigBourin" (fr) for this function

Parameters:

str1 first mask

str2 second mask

Returns:

1 if match, else 0

Definition at line 536 of file tools.cpp.

Referenced by Admin::getMaskLevel(), and Admin::maskIsSuperAdmin().

6.42.3.15 string Tools::parseQ3Colors (string *raw*) `[static]`

Parse Quake colors from a string.

Strip all Quake III Arena color codes from a string

Parameters:

raw String with q3 color codes

Returns:

String with no q3 colors

Definition at line 489 of file tools.cpp.

Referenced by GameServer::parseQ3infos(), GameServer::parseWSWinfos(), q3(), and warsow().

6.42.3.16 int Tools::random (int *min*, int *max*) `[static]`

Give a random number.

Generate a random int between a min and a max value

Parameters:

min Minimum random value

max Maximum random value

Returns:

radom int

Definition at line 268 of file tools.cpp.

Referenced by Lamoule::generateScore(), Magic8Ball::getRandomAnswer(), Quotes::getRandomQuote(), lamoule(), randomKick(), and Quotes::searchQuote().

6.42.3.17 vector< string > Tools::stringToVector (string *str*, string *separator*, unsigned int *start* = 0) `[static]`

Convert a string to a vector.

Split a string in elements inserted in a vector

Parameters:

str String to split

separator Delimitor for string

start Index of the first element to split

Returns:

A vector containing strings

Definition at line 304 of file tools.cpp.

Referenced by autoop(), autovoice(), danstonchat(), LogFactory::getLoggedChannels(), Fedorafr::getWikiLinks(), LogFactory::hasToBeLogged(), hl(), joinHandler(), launchSurvey(), ConfigurationFile::load(), FedoraProject::loadFasFile(), BotKernel::loadPlugins(), modeHandlerProtect(), onEndOfMOTD(), GameServer::parseQ3infos(), GameServer::parseWSWinfos(), protectmodes(), protecttopic(), q3(), BotKernel::run(), Message::setMessage(), topicHandler(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), and warsow().

6.42.3.18 unsigned int Tools::strtimeToSeconds (string *strtime*) `[static]`

Convert a string time to seconds.

Convert a 'string time' to seconds. Exemple : 2d6h2m1s = 194521 seconds d for 'days' h for 'hours' m for 'minuts' s for 'seconds'

Parameters:

strtime String time to convert

Returns:

seconds convertion

Definition at line 211 of file tools.cpp.

References strToUnsignedInt().

Referenced by addad(), addIgnore(), addtempsuperadmin(), ban(), banmask(), launchSurvey(), and slapme().

6.42.3.19 double Tools::strToDouble (string *str*) `[static]`

Convert a string to a double.

Convert a string to a double if operation fails, result is 0.0

Parameters:

str string to convert

Returns:

Double convertion result

Definition at line 147 of file tools.cpp.

Referenced by Lamoule::get5first(), Lamoule::getInfosPlayer(), Lamoule::increaseScore(), player(), and Lamoule::sort().

6.42.3.20 int Tools::strToInt (string *str*) `[static]`

Convert a string to a int.

Convert a string to a int if operation fails, result is 0

Parameters:

str string to convert

Returns:

Int convertion result

Definition at line 167 of file tools.cpp.

Referenced by adinfos(), Advertising::Advertising(), bandel(), baninfos(), Moderation::banInfos(), chanlev(), Moderation::clearOutBans(), Admin::clearTempAdmins(), BotKernel::connect(), Advertising::deleteOutdatedAds(), delIgnore(), delQuote(), delsuperadmin(), Advertising::getAdvertisesList(), Ignore::getIgnoreList(), Admin::getMaskLevel(), Admin::getUserLevel(), increase(), Lamoule::increaseScore(), lamoule(), nextscore(), onEndOfMOTD(), planet(), player(), purifyFile(), Lamoule::purifyFile(), Ignore::purifyList(), quote(), quoteInfos(), BotKernel::run(), GameServer::sendQuery(), Lamoule::sort(), Admin::superAdminList(), testMsgTimestamp(), top5(), toptotal(), and wiki().

6.42.3.21 unsigned int Tools::strToUnsignedInt (string *str*) `[static]`

Convert a string to an unsigned int.

Convert a string to an unsigned int if operation fails, result is 0

Parameters:

str string to convert

Returns:

Int convertion result

Definition at line 187 of file tools.cpp.

Referenced by BotKernel::addCountDown(), bannedHandler(), danstonchat(), getMyFirstNick(), launch-Survey(), rejoinChan(), RemoteControl::RemoteControl(), BZRH::searchBugs(), secondaryNick(), str-timeToSeconds(), and topicInfos().

6.42.3.22 string Tools::to_lower (string *str*) `[static]`

Convert a string to lower case.

Put an entire string to lower case

Parameters:

str string to convert

Returns:

string converted to lower case

Definition at line 237 of file tools.cpp.

Referenced by Admin::addChannel(), Ignore::addIgnore(), Admin::addOnlyonCommand(), Admin::addSuperAdmin(), Admin::addTempSuperAdmin(), Admin::addUser(), Admin::chanLevels(), Admin::channelExists(), Admin::commandOK(), Admin::delChannel(), Lamoule::deletePlayer(), Admin::delOnlyonCommand(), Admin::delUser(), Admin::disableCommand(), Admin::enableCommand(), Lamoule::getInfosPlayer(), Admin::getMaskLevel(), Admin::getUserLevel(), Lamoule::increaseScore(), Ignore::isIgnored(), Admin::isSuperAdmin(), Admin::maskIsSuperAdmin(), BotKernel::msgTreatment(), Quotes::searchQuote(), Admin::updateUserLevel(), and Admin::userExists().

6.42.3.23 string Tools::to_upper (string *str*) `[static]`

Convert a string to upper case.

Put an entire string to upper case

Parameters:

str string to convert

Returns:

string converted to upper case

Definition at line 252 of file tools.cpp.

6.42.3.24 string Tools::urlencode (string *str*) `[static]`

Encode a string to a URL format.

Convert special chars from a string to be compatible to URL format

Parameters:

str String to encode

Returns:

String compatible to URL format

Definition at line 411 of file tools.cpp.

Referenced by BZRH::getBugInfos(), planet(), BZRH::searchBugs(), trad(), FedoraProject::whoowns(), and wiki().

6.42.3.25 string Tools::vectorToString (vector< string > *vec*, string *separator*, unsigned int *start* = `0`) `[static]`

Convert a vector to a string.

Convert a vector<string> to a string

Parameters:

vec Vector to convert

separator Elements separator in generated string

start Indice for starting conversion

Returns:

A string containing vector values

Definition at line 280 of file tools.cpp.

Referenced by addad(), addQuote(), ban(), banmask(), bzsearch(), fas(), greplog(), hl(), kick(), kickHandler(), launchSurvey(), leaveChannel(), notice(), partHandler(), planet(), privmsgHandler(), q3(), quitHandler(), raw(), searchQuote(), sendHandler(), slapUser(), tell(), topic(), topicHandler(), topicJoin(), trad(), unautoop(), unautovoice(), unprotectmodes(), unprotecttopic(), warsow(), and wiki().

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

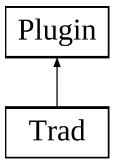
- src/tools.h
- src/tools.cpp

6.43 Trad Class Reference

Provides a command to translate a sentence from a language to an other using translate.google.com.

`#include <trad.h>`

Inheritance diagram for Trad::



Public Member Functions

- Trad (BotKernel ∗)

    Constructor.

6.43.1 Detailed Description

Provides a command to translate a sentence from a language to an other using translate.google.com.

Provides a command to translate a sentence from a language to an other using translate.google.com

Definition at line 41 of file trad.h.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Trad::Trad (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file trad.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_COMMAND_HANDLER, Plugin::name, and Plugin::version.

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

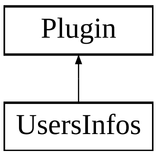
- src/plugins/trad.h
- src/plugins/trad.cpp

6.44 UsersInfos Class Reference

Follow users modes on channels.

```
#include <usersinfos.h>
```

Inheritance diagram for UsersInfos::



Public Member Functions

- UsersInfos (BotKernel ∗)

  Constructor.

- ∼UsersInfos ()

  Destructor.

- void addPrefixe (char, char)

  Add a prefixe translation.

- char getPrefixe (char)

  Get a prefixe translation.

- string getPrefixes ()

  Get all prefixes.

- map< string, Channel ∗ > ∗ getUsers ()

  Return the users attribute pointer.

- bool hasMode (string, string, char)

  tell if a user has the given mode on a given channel

- vector< string > ∗ getLastQuitChannels ()

  Get channels where the last user who quitted was present.

Private Attributes

- vector< string > prefixes

  Vector for prefixes translation (@=>o,+=>v).

- map< string, Channel ∗ > users

  Channels storage.

- vector< string > lastQuitChannels

Stores channels where the last user who quitted was present.

6.44.1 Detailed Description

Follow users modes on channels.

This plugin gives functions to follow users modes on channels and keep informations (nick,ident,host,status) Can be usefull for operator commands and ban masks

Definition at line 45 of file usersinfos.h.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 UsersInfos::UsersInfos (BotKernel ∗ *b*)

Constructor.

Constructor

Definition at line 34 of file usersinfos.cpp.

References Plugin::author, Plugin::bindFunction(), Plugin::description, IN_LOOP, IN_TYPE_-HANDLER, lastQuitChannels, Plugin::name, and Plugin::version.

6.44.2.2 UsersInfos::∼UsersInfos ()

Destructor.

Destructor

Definition at line 56 of file usersinfos.cpp.

References users.

6.44.3 Member Function Documentation

6.44.3.1 void UsersInfos::addPrefixe (char *mode*, char *prefixe*)

Add a prefixe translation.

Add a prefixe translation for a mode Examples : @ for o % for h + for v

Parameters:

mode Mode for witch you give a prefixe

prefixe Prefixe corresponding to the given mode

Definition at line 73 of file usersinfos.cpp.

References prefixes.

Referenced by event005().

6.44.3.2 vector< string > ∗ UsersInfos::getLastQuitChannels ()

Get channels where the last user who quitted was present.

Get channels where the last user who quitted was present return Channels where the last user who quitted was present

Definition at line 143 of file usersinfos.cpp.

References lastQuitChannels.

Referenced by onQuit(), and quitHandler().

6.44.3.3 char UsersInfos::getPrefixe (char *mode*)

Get a prefixe translation.

Get a prefix correspondig to a mode Examples : @ for o % for h + for v

Parameters:

mode Mode for witch you want the prefix

Returns:

Prefix corresponding to the given mode

Definition at line 90 of file usersinfos.cpp.

References prefixes.

Referenced by mode().

6.44.3.4 string UsersInfos::getPrefixes ()

Get all prefixes.

Get all prefixes

Returns:

all prefixes in a string

Definition at line 107 of file usersinfos.cpp.

References prefixes.

Referenced by modeHandlerProtect().

6.44.3.5 map< string, Channel ∗ > ∗ UsersInfos::getUsers ()

Return the users attribute pointer.

Gives A pointer to the users's kernel attribute

Returns:

A pointer to the users's storage attribute

Definition at line 152 of file usersinfos.cpp.

References users.

Referenced by ban(), LogFactory::cleanLogs(), clearOutBans(), Moderation::getChanUsersList(), LogFactory::getLoggedChannels(), kickHandler(), lamoule(), modeHandler(), partHandler(), quitHandler(), topicHandler(), and topicJoin().

6.44.3.6 bool UsersInfos::hasMode (string *channel*, string *nick*, char *mode*)

tell if a user has the given mode on a given channel

Tell if a user has the given mode on a given channel

Parameters:

mode Mode to test for the user

channel Channel to check

nick Nick to check

Returns:

True if the user has the mode, else false

Definition at line 123 of file usersinfos.cpp.

References users.

Referenced by Moderation::checkMode(), and Moderation::hasOpPrivileges().

6.44.4 Member Data Documentation

6.44.4.1 vector<string> UsersInfos::lastQuitChannels `[private]`

Stores channels where the last user who quitted was present.

Definition at line 53 of file usersinfos.h.

Referenced by getLastQuitChannels(), and UsersInfos().

6.44.4.2 vector<string> UsersInfos::prefixes `[private]`

Vector for prefixes translation (@=>o,+=>v).

Definition at line 49 of file usersinfos.h.

Referenced by addPrefixe(), getPrefixe(), and getPrefixes().

6.44.4.3 map<string,Channel*> UsersInfos::users `[private]`

Channels storage.

Definition at line 51 of file usersinfos.h.

Referenced by getUsers(), hasMode(), and ~UsersInfos().

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

- src/plugins/usersinfos.h
- src/plugins/usersinfos.cpp

Chapter 7

File Documentation

7.1 src/botkernel.cpp File Reference

BotKernel implementation file.

```
#include "botkernel.h"
```

Functions

- void ∗ threadFunc (void ∗)

  Thread function to manage timeouts.

7.1.1 Detailed Description

BotKernel implementation file.

Definition in file botkernel.cpp.

7.1.2 Function Documentation

7.1.2.1 void ∗ threadFunc (void ∗ *args*)

Thread function to manage timeouts.

Definition at line 1045 of file botkernel.cpp.

References ThreadParams::b, StructFunctionStorage::back, StructFunctionStorage::function, ThreadParams::function, ThreadParams::msg, StructFunctionStorage::object, and ThreadParams::sem.

Referenced by BotKernel::executeFunction().

7.2 src/botkernel.h File Reference

BotKernel header file.

```
#include "message.h"
#include "ircprotocol.h"
#include "configurationfile.h"
#include "logfile.h"
#include "tools.h"
#include "socket.h"
#include "plugin.h"
#include "cppthread.h"
#include <sys/time.h>
#include <sys/types.h>
#include <semaphore.h>
#include <errno.h>
#include <dlfcn.h>
#include <dirent.h>
#include <time.h>
#include <list>
```

Classes

- struct AntiExcessFlood

  Anti excess-flood variables.

- struct CountDownFunction

  Countdown information storage.

- struct ThreadParams

  Thread information storage.

- class BotKernel

  Bot kernel class.

7.2.1 Detailed Description

BotKernel header file.

Definition in file botkernel.h.

7.3 src/channel.cpp File Reference

Channel implementation file.

```
#include "channel.h"
```

7.3.1 Detailed Description

Channel implementation file.

Definition in file channel.cpp.

7.4 src/channel.h File Reference

Channel header file.

```
#include "tools.h"
#include <vector>
#include <string>
#include <iostream>
```

Classes

- class Channel

  Channel management class.

7.4.1 Detailed Description

Channel header file.

Definition in file channel.h.

7.5 src/configurationfile.cpp File Reference

ConfigurationFile implementation file.

```
#include "configurationfile.h"
```

7.5.1 Detailed Description

ConfigurationFile implementation file.

Definition in file configurationfile.cpp.

7.6 src/configurationfile.h File Reference

ConfigurationFile header file.

```
#include "tools.h"
#include <string>
#include <fstream>
#include <map>
```

Classes

- class ConfigurationFile

  Configuration file class.

7.6.1 Detailed Description

ConfigurationFile header file.

Definition in file configurationfile.h.

7.7 src/cppthread.cpp File Reference

CPPThread implementation file.

```
#include "cppthread.h"
#include <iostream>
```

7.7.1 Detailed Description

CPPThread implementation file.

Definition in file cppthread.cpp.

7.8 src/cppthread.h File Reference

CPPThread header file.

```
#include <pthread.h>
```

Classes

- struct threadInfos

  Stores thread informations.

- class CPPThread

  pthread C++ wrapper

Typedefs

- typedef void ∗(∗ threadProcess)(void ∗)

7.8.1 Detailed Description

CPPThread header file.

Definition in file cppthread.h.

7.8.2 Typedef Documentation

7.8.2.1 typedef void∗(∗ threadProcess)(void ∗)

Definition at line 34 of file cppthread.h.

7.9 src/ircprotocol.cpp File Reference

IRCProtocol implementation file.

```
#include "ircprotocol.h"
```

7.9.1 Detailed Description

IRCProtocol implementation file.

Definition in file ircprotocol.cpp.

7.10 src/ircprotocol.h File Reference

IRCProtocol header file.

```
#include <string>
#include <vector>
```

Classes

- class IRCProtocol
  Class that convert messages to IRC messages.

7.10.1 Detailed Description

IRCProtocol header file.

Definition in file ircprotocol.h.

7.11 src/logfile.cpp File Reference

LogFile implementation file.

```
#include "logfile.h"
```

7.11.1 Detailed Description

LogFile implementation file.

Definition in file logfile.cpp.

7.12 src/logfile.h File Reference

LogFile header file.

```
#include <sys/stat.h>
#include <strings.h>
#include <string>
#include <fstream>
#include <iostream>
#include <map>
```

Classes

- class LogFile

  Class that manage log system.

Enumerations

- enum log_level {
  NOTUSED, NOTHING, ERROR, WARNING,
  INFO }

  Log levels.

7.12.1 Detailed Description

LogFile header file.

Definition in file logfile.h.

7.12.2 Enumeration Type Documentation

7.12.2.1 enum log_level

Log levels.

Enumerator:

NOTUSED

NOTHING

ERROR

WARNING

INFO

Definition at line 41 of file logfile.h.

7.13 src/main.cpp File Reference

Main program.

```
#include "botkernel.h"
#include "cppthread.h"
```

Functions

- vector< string > listConfFiles (string)
- void launchThreads (vector< string >, vector< CPPThread * > *, vector< BotKernel * > *)
- void * launchBot (void *)
- void displayHelp (string, bool)
- int main (int nbArgs, char *arrayArgs[ ])

7.13.1 Detailed Description

Main program.

Construct a botkernel and launch it.

Definition in file main.cpp.

7.13.2 Function Documentation

7.13.2.1 void displayHelp (string *firstArg*, bool *quit*)

Definition at line 131 of file main.cpp.

Referenced by main().

7.13.2.2 void * launchBot (void * *arg*)

Definition at line 126 of file main.cpp.

Referenced by launchThreads().

7.13.2.3 void launchThreads (vector< string > *confFiles*, vector< CPPThread * > * *threads*, vector< BotKernel * > * *bots*)

Definition at line 118 of file main.cpp.

References launchBot().

Referenced by main().

7.13.2.4 vector< string > listConfFiles (string *confDir*)

Definition at line 96 of file main.cpp.

Referenced by main().

7.13.2.5 int main (int *nbArgs*, char ∗ *arrayArgs*[ ])

Definition at line 39 of file main.cpp.

References displayHelp(), launchThreads(), and listConfFiles().

7.14 src/message.cpp File Reference

Message implementation file.

```
#include "message.h"
```

7.14.1 Detailed Description

Message implementation file.

Definition in file message.cpp.

7.15 src/message.h File Reference

Message header file.

```
#include "tools.h"
#include <string>
#include <iostream>
```

Classes

- class Message

    Class that manage messages from the irc server.

7.15.1 Detailed Description

Message header file.

Definition in file message.h.

7.16 src/plugin.cpp File Reference

Plugin implementation file.

```
#include "plugin.h"
```

7.16.1 Detailed Description

Plugin implementation file.

Definition in file plugin.cpp.

7.17 src/plugin.h File Reference

Plugin header file.

```
#include "message.h"
#include <time.h>
#include <vector>
#include <string>
```

Classes

- struct pPlugin
  Plugin object and header storage.
- struct StructFunctionStorage
  Plugin function storage.
- class Plugin
  Class that manage a plugin.

Typedefs

- typedef bool(∗ plugin_function)(Message ∗, Plugin ∗, BotKernel ∗)
  Plugin function prototype.
- typedef Plugin ∗(∗ plugin_constructor)(BotKernel ∗)
  Plugin object constructor prototype.
- typedef void(∗ plugin_destructor)(Plugin ∗)
  Plugin objet destructor prototype.

Enumerations

- enum func_type {
  IN_LOOP, IN_COMMAND_HANDLER, IN_FREE_COMMAND_HANDLER, IN_TYPE_-HANDLER,
  IN_BEFORE_TREATMENT, IN_ALL_MSGS, IN_FIRST_WORD, COUNTDOWN,
  OUT_ALL_MSGS }
  Plugin types.

7.17.1 Detailed Description

Plugin header file.

Definition in file plugin.h.

7.17.2 Typedef Documentation

7.17.2.1 typedef Plugin∗(∗ plugin_constructor)(BotKernel ∗)

Plugin object constructor prototype.

Definition at line 48 of file plugin.h.

7.17.2.2 typedef void(∗ plugin_destructor)(Plugin ∗)

Plugin objet destructor prototype.

Definition at line 50 of file plugin.h.

7.17.2.3 typedef bool(∗ plugin_function)(Message ∗, Plugin ∗, BotKernel ∗)

Plugin function prototype.

Definition at line 46 of file plugin.h.

7.17.3 Enumeration Type Documentation

7.17.3.1 enum func_type

Plugin types.

Enumerator:

IN_LOOP

IN_COMMAND_HANDLER

IN_FREE_COMMAND_HANDLER

IN_TYPE_HANDLER

IN_BEFORE_TREATMENT

IN_ALL_MSGS

IN_FIRST_WORD

COUNTDOWN

OUT_ALL_MSGS

Definition at line 41 of file plugin.h.

7.18 src/plugins/admin.cpp File Reference

Admin implementation file.

```
#include "admin.h"
```

Functions

- Plugin ∗ contruct_admin (BotKernel ∗b)
- void destroy_admin (Plugin ∗p)
- bool addsuperadmin (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool addtempsuperadmin (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool setSuperAdminPass (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool clearTemporaryAdmins (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool superadminlist (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool delsuperadmin (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool reset (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool disconnect (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool getconfvalue (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool setconfvalue (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool deletekey (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool loadconffile (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool flushconffile (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool setloglevel (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool setlogkeepfiles (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool setlogperiod (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool addOnlyon (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool delOnlyon (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool enable (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool disable (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool commandsStatus (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool allowedCommandCheck (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool chanlev (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool joinChannel (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool leaveChannel (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool cycleChannel (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool setNick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool tell (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool notice (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool raw (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool whoami (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool reauth (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool onInvite (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool error (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool getnbcountdowns (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool clearCountDowns (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.18.1 Detailed Description

Admin implementation file.

Definition in file admin.cpp.

7.18.2 Function Documentation

7.18.2.1 bool addOnlyon (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1014 of file admin.cpp.

References Admin::addOnlyonCommand(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.2 bool addsuperadmin (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 756 of file admin.cpp.

References Admin::addSuperAdmin(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), ConfigurationFile::getValue(), INFO, Message::isPrivate(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.3 bool addtempsuperadmin (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 776 of file admin.cpp.

References Admin::addTempSuperAdmin(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), ConfigurationFile::getValue(), INFO, Message::isPrivate(), LogFile::log(), BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strtimeToSeconds().

7.18.2.4 bool allowedCommandCheck (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1096 of file admin.cpp.

References Admin::commandOK(), BotKernel::getCONFF(), Message::getPart(), Message::getSource(), Message::getSplit(), ConfigurationFile::getValue(), and Message::isPublic().

7.18.2.5 bool chanlev (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1113 of file admin.cpp.

References Admin::chanLevels(), Tools::gatherVectorElements(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), Admin::getUserLevel(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), IRCProtocol::sendNotice(), IRCProtocol::sendNotices(), Tools::strToInt(), and Admin::updateUserLevel().

7.18.2.6 bool clearCountDowns (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1323 of file admin.cpp.

References BotKernel::getCountDowns(), Message::getNickSender(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.7 bool clearTemporaryAdmins (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 813 of file admin.cpp.

7.18.2.8 bool commandsStatus (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1086 of file admin.cpp.

References Admin::commandsStatus(), Tools::gatherVectorElements(), Message::getNickSender(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), and IRCProtocol::sendNotices().

7.18.2.9 Plugin∗ contruct_admin (BotKernel ∗ *b*)

Definition at line 748 of file admin.cpp.

7.18.2.10 bool cycleChannel (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1171 of file admin.cpp.

References Message::getPart(), Message::getSender(), Message::getSplit(), Message::isPrivate(), Admin::isSuperAdmin(), IRCProtocol::joinChannel(), IRCProtocol::leaveChannel(), and BotKernel::send().

7.18.2.11 bool deletekey (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 905 of file admin.cpp.

References ConfigurationFile::delKey(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.12 bool delOnlyon (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1032 of file admin.cpp.

References Admin::delOnlyonCommand(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.13 bool delsuperadmin (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 828 of file admin.cpp.

References Admin::delSuperAdmin(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), ConfigurationFile::getValue(), INFO, Message::isPrivate(), LogFile::log(), BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.18.2.14 void destroy_admin (Plugin ∗ *p*)

Definition at line 752 of file admin.cpp.

7.18.2.15 bool disable (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1068 of file admin.cpp.

References Admin::disableCommand(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.16 bool disconnect (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 860 of file admin.cpp.

References Message::getSender(), BotKernel::getSysLog(), Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), IRCProtocol::quitServer(), BotKernel::send(), BotKernel::stop(), and WARNING.

7.18.2.17 bool enable (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1050 of file admin.cpp.

References Admin::enableCommand(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.18 bool error (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1310 of file admin.cpp.

References ERROR, Message::getMessage(), BotKernel::getSysLog(), and LogFile::log().

Referenced by BotKernel::loadPlugin().

7.18.2.19 bool flushconffile (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 952 of file admin.cpp.

References ConfigurationFile::flush(), BotKernel::getCONFF(), Message::getNickSender(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), IRCProtocol::sendNotice(), and WARNING.

7.18.2.20 bool getconfvalue (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 874 of file admin.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), ConfigurationFile::getValue(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.21 bool getnbcountdowns (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1315 of file admin.cpp.

References BotKernel::getCountDowns(), Message::getNickSender(), Message::getSender(), Tools::intToStr(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), and IRCProtocol::sendNotice().

7.18.2.22 bool joinChannel (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1137 of file admin.cpp.

References Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), IRCProtocol::joinChannel(), LogFile::log(), and BotKernel::send().

7.18.2.23 bool leaveChannel (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1154 of file admin.cpp.

References Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), IRCProtocol::leaveChannel(), LogFile::log(), BotKernel::send(), and Tools::vectorToString().

7.18.2.24 bool loadconffile (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 929 of file admin.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), ConfigurationFile::load(), LogFile::log(), BotKernel::send(), IRCProtocol::sendNotice(), and WARNING.

7.18.2.25 bool notice (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1223 of file admin.cpp.

References Message::getPart(), Message::getSender(), Message::getSplit(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), IRCProtocol::sendNotice(), and Tools::vectorToString().

7.18.2.26 bool onInvite (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1297 of file admin.cpp.

References Message::getPart(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), IRCProtocol::joinChannel(), LogFile::log(), and BotKernel::send().

7.18.2.27 bool raw (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1239 of file admin.cpp.

References Message::getSender(), Message::getSplit(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), and Tools::vectorToString().

7.18.2.28 bool reauth (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1275 of file admin.cpp.

References BotKernel::getPlugin(), Message::getSender(), pPlugin::handle, Message::isPrivate(), Admin::isSuperAdmin(), and pPlugin::object.

7.18.2.29 bool reset (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 848 of file admin.cpp.

References Message::getSender(), BotKernel::getSysLog(), Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::setConnected(), and WARNING.

7.18.2.30 bool setconfvalue (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 887 of file admin.cpp.

References BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), IRCProtocol::sendNotice(), and ConfigurationFile::setValue().

7.18.2.31 bool setlogkeepfiles (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 988 of file admin.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), Message::getPart(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendNotice(), LogFile::setKeepFiles(), and ConfigurationFile::setValue().

7.18.2.32 bool setloglevel (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 975 of file admin.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), Message::getPart(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendNotice(), LogFile::setLogLevel(), and ConfigurationFile::setValue().

7.18.2.33 bool setlogperiod (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1001 of file admin.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), Message::getPart(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendNotice(), LogFile::setPeriodFormat(), and ConfigurationFile::setValue().

7.18.2.34 bool setNick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1188 of file admin.cpp.

References IRCProtocol::changeNick(), BotKernel::getCONFF(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), BotKernel::send(), BotKernel::setNick(), and ConfigurationFile::setValue().

7.18.2.35 bool setSuperAdminPass (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 796 of file admin.cpp.

References BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), ConfigurationFile::getValue(), Message::isPrivate(), LogFile::log(), BotKernel::send(), IRCProtocol::sendNotice(), ConfigurationFile::setValue(), and WARNING.

7.18.2.36 bool superadminlist (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 818 of file admin.cpp.

References Tools::gatherVectorElements(), Message::getNickSender(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), IRCProtocol::sendNotices(), and Admin::superAdminList().

7.18.2.37 bool tell (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1207 of file admin.cpp.

References Message::getPart(), Message::getSender(), Message::getSplit(), Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::vectorToString().

7.18.2.38 bool whoami (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1255 of file admin.cpp.

References Admin::getChannelsList(), Message::getNickSender(), Message::getSender(), Admin::getUserLevel(), Tools::intToStr(), Admin::isSuperAdmin(), BotKernel::send(), and IRCProtocol::sendNotices().

7.19 src/plugins/admin.h File Reference

Admin header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../tinyxml/tinyxml.h"
#include "postconnect.h"
#include <iostream>
```

Classes

- class Admin

  Bot access management.

7.19.1 Detailed Description

Admin header file.

Definition in file admin.h.

7.20 src/plugins/advertising.cpp File Reference

Advertising implementation file.

```
#include "advertising.h"
```

Functions

- bool displayAdvertise (Message ∗, Plugin ∗, BotKernel ∗)
- Plugin ∗ contruct_advertising (BotKernel ∗b)
- void destroy_advertising (Plugin ∗p)
- bool addad (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool delad (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool adinfos (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool listads (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool cleanList (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.20.1 Detailed Description

Advertising implementation file.

Definition in file advertising.cpp.

7.20.2 Function Documentation

7.20.2.1 bool addad (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 244 of file advertising.cpp.

References Advertising::addAdvertise(), BotKernel::addCountDown(), displayAdvertise(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSplit(), Tools::intToStr(), Message::isPrivate(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), Message::setMessage(), Tools::strtimeToSeconds(), and Tools::vectorToString().

7.20.2.2 bool adinfos (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 289 of file advertising.cpp.

References Advertising::getAdvertiseInfos(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.20.2.3 bool cleanList (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 327 of file advertising.cpp.

References Advertising::deleteOutdatedAds().

7.20.2.4 Plugin∗ contruct_advertising (BotKernel ∗ *b*)

Definition at line 226 of file advertising.cpp.

7.20.2.5 bool delad (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 271 of file advertising.cpp.

References Advertising::delAdvertise(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotice().

7.20.2.6 void destroy_advertising (Plugin ∗ *p*)

Definition at line 230 of file advertising.cpp.

7.20.2.7 bool displayAdvertise (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 234 of file advertising.cpp.

References Advertising::getAdvertiseInfos(), Message::getMessage(), BotKernel::send(), and IRCProtocol::sendMsg().

Referenced by addad(), and Advertising::launchAdvertise().

7.20.2.8 bool listads (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 314 of file advertising.cpp.

References Advertising::getAdvertisesList(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotices().

7.21 src/plugins/advertising.h File Reference

Advertising header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../tinyxml/tinyxml.h"
#include "admin.h"
#include <time.h>
#include <iostream>
```

Classes

- class Advertising

  Plugin managing ads.

7.21.1 Detailed Description

Advertising header file.

Definition in file advertising.h.

7.22 src/plugins/antiflood.cpp File Reference

AntiFlood implementation file.

```
#include "antiflood.h"
```

Functions

- Plugin ∗ contruct_antiflood (BotKernel ∗b)
- void destroy_antiflood (Plugin ∗p)
- bool testMsgTimestamp (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.22.1 Detailed Description

AntiFlood implementation file.

Definition in file antiflood.cpp.

7.22.2 Function Documentation

7.22.2.1 Plugin∗ contruct_antiflood (BotKernel ∗ *b*)

Definition at line 46 of file antiflood.cpp.

7.22.2.2 void destroy_antiflood (Plugin ∗ *p*)

Definition at line 50 of file antiflood.cpp.

7.22.2.3 bool testMsgTimestamp (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 54 of file antiflood.cpp.

References BotKernel::getCONFF(), Message::getElapsedTime(), Plugin::getName(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), ConfigurationFile::getValue(), Admin::isSuperAdmin(), pPlugin::object, and Tools::strToInt().

7.23 src/plugins/antiflood.h File Reference

AntiFlood header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "admin.h"
#include <iostream>
```

Classes

- class AntiFlood

  Plugin that able the bot to detect flood.

7.23.1 Detailed Description

AntiFlood header file.

Definition in file antiflood.h.

7.24 src/plugins/bzrh.cpp File Reference

BZRH implementation file.

```
#include "bzrh.h"
```

Functions

- Plugin ∗ contruct_bzrh (BotKernel ∗b)
- void destroy_bzrh (Plugin ∗p)
- bool bzsearch (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool bug (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool checkBug (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.24.1 Detailed Description

BZRH implementation file.

Definition in file bzrh.cpp.

7.24.2 Function Documentation

7.24.2.1 bool bug (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 256 of file bzrh.cpp.

References BZRH::getBugInfos(), Message::getPart(), Message::getSource(), Message::isPublic(), Message::nbParts(), BotKernel::send(), and IRCProtocol::sendMsg().

7.24.2.2 bool bzsearch (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 248 of file bzrh.cpp.

References BotKernel::getCONFF(), Plugin::getName(), Message::getSource(), Message::getSplit(), ConfigurationFile::getValue(), Message::isPublic(), Message::nbParts(), BZRH::searchBugs(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::vectorToString().

7.24.2.3 bool checkBug (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 264 of file bzrh.cpp.

References BZRH::getBugInfos(), BotKernel::getCONFF(), Plugin::getName(), Message::getPart(), Message::getSource(), ConfigurationFile::getValue(), Message::isPublic(), Message::nbParts(), BotKernel::send(), and IRCProtocol::sendMsg().

7.24.2.4 Plugin∗ contruct_bzrh (BotKernel ∗ *b*)

Definition at line 240 of file bzrh.cpp.

7.24.2.5 void destroy_bzrh (Plugin $*$ *p*)

Definition at line 244 of file bzrh.cpp.

7.25 src/plugins/bzrh.h File Reference

BZRH header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "curl/curl.h"
#include <iostream>
```

Classes

- class BZRH

  BZRH provides commands t query bugzilla.redhat.com.

7.25.1 Detailed Description

BZRH header file.

Definition in file bzrh.h.

7.26 src/plugins/ctcp.cpp File Reference

CTCP implementation file.

```
#include "ctcp.h"
```

Functions

- Plugin ∗ contruct_ctcp (BotKernel ∗b)
- void destroy_ctcp (Plugin ∗p)
- bool ctcp_ping (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool ctcp_version (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.26.1 Detailed Description

CTCP implementation file.

Definition in file ctcp.cpp.

7.26.2 Function Documentation

7.26.2.1 Plugin∗ contruct_ctcp (BotKernel ∗ *b*)

Definition at line 46 of file ctcp.cpp.

7.26.2.2 bool ctcp_ping (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 54 of file ctcp.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::send(), and IRCProtocol::sendNotice().

7.26.2.3 bool ctcp_version (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 59 of file ctcp.cpp.

References Message::getNickSender(), BotKernel::getVersion(), BotKernel::send(), and IRCProtocol::sendNotice().

7.26.2.4 void destroy_ctcp (Plugin ∗ *p*)

Definition at line 50 of file ctcp.cpp.

7.27 src/plugins/ctcp.h File Reference

CTCP header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class CTCP

  Provide CTCP Answers.

7.27.1 Detailed Description

CTCP header file.

Definition in file ctcp.h.

7.28 src/plugins/danstonchat.cpp File Reference

DansTonChat implementation file.

```
#include "danstonchat.h"
```

Functions

- Plugin ∗ contruct_danstonchat (BotKernel ∗b)
- void destroy_danstonchat (Plugin ∗p)
- bool danstonchat (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.28.1 Detailed Description

DansTonChat implementation file.

Definition in file danstonchat.cpp.

7.28.2 Function Documentation

7.28.2.1 Plugin∗ contruct_danstonchat (BotKernel ∗ *b*)

Definition at line 46 of file danstonchat.cpp.

7.28.2.2 bool danstonchat (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 54 of file danstonchat.cpp.

References Tools::cleanHTML(), Tools::clearAccents(), Socket::connectSock(), BotKernel::getCONFF(), Plugin::getName(), Message::getPart(), Message::getSource(), ConfigurationFile::getValue(), Message::isPublic(), Message::nbParts(), Socket::receive(), BotKernel::send(), IRCProtocol::sendMsg(), Socket::sendStr(), Tools::stringToVector(), and Tools::strToUnsignedInt().

7.28.2.3 void destroy_danstonchat (Plugin ∗ *p*)

Definition at line 50 of file danstonchat.cpp.

7.29 src/plugins/danstonchat.h File Reference

DansTonChat header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class DansTonChat

  Display quotes from danstonchat.com.

7.29.1 Detailed Description

DansTonChat header file.

Definition in file danstonchat.h.

7.30 src/plugins/fedorafr.cpp File Reference

Fedorafr implementation file.

```
#include "fedorafr.h"
```

Functions

- Plugin ∗ contruct_fedorafr (BotKernel ∗b)
- void destroy_fedorafr (Plugin ∗p)
- bool wiki (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool planet (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool displayPaste (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.30.1 Detailed Description

Fedorafr implementation file.

Definition in file fedorafr.cpp.

7.30.2 Function Documentation

7.30.2.1 Plugin∗ contruct_fedorafr (BotKernel ∗ *b*)

Definition at line 72 of file fedorafr.cpp.

7.30.2.2 void destroy_fedorafr (Plugin ∗ *p*)

Definition at line 76 of file fedorafr.cpp.

7.30.2.3 bool displayPaste (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 184 of file fedorafr.cpp.

References Message::getSource(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendMsg().

7.30.2.4 bool planet (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 134 of file fedorafr.cpp.

References Socket::connectSock(), BotKernel::getCONFF(), Plugin::getName(), Message::getPart(), Message::getSource(), Message::getSplit(), ConfigurationFile::getValue(), Message::isPublic(), Message::nbParts(), Socket::receive(), BotKernel::send(), IRCProtocol::sendMsg(), Socket::sendStr(), Tools::strToInt(), Tools::urlencode(), and Tools::vectorToString().

7.30.2.5 bool wiki (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 80 of file fedorafr.cpp.

References Socket::connectSock(), BotKernel::getCONFF(), Plugin::getName(), Message::getPart(), Message::getSource(), Message::getSplit(), ConfigurationFile::getValue(), Fedorafr::getWikiLinks(), Message::isPublic(), Message::nbParts(), Socket::receive(), BotKernel::send(), IRCProtocol::sendMsg(), Socket::sendStr(), Tools::strToInt(), Tools::urlencode(), and Tools::vectorToString().

7.31 src/plugins/fedorafr.h File Reference

Fedorafr header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class Fedorafr

  Class that provides stuff to search on Fedora-fr.org wiki or planet.

7.31.1 Detailed Description

Fedorafr header file.

Definition in file fedorafr.h.

7.32 src/plugins/fedoraproject.cpp File Reference

FedoraProject implementation file.

`#include "fedoraproject.h"`

Functions

- Plugin ∗ contruct_fedoraproject (BotKernel ∗b)
- void destroy_fedoraproject (Plugin ∗p)
- bool whoowns (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool fas (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool reloadfas (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.32.1 Detailed Description

FedoraProject implementation file.

Definition in file fedoraproject.cpp.

7.32.2 Function Documentation

7.32.2.1 Plugin∗ contruct_fedoraproject (BotKernel ∗ *b*)

Definition at line 151 of file fedoraproject.cpp.

7.32.2.2 void destroy_fedoraproject (Plugin ∗ *p*)

Definition at line 155 of file fedoraproject.cpp.

7.32.2.3 bool fas (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 167 of file fedoraproject.cpp.

References FedoraProject::getFasUserInfos(), Message::getPart(), Message::getSource(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::vectorToString().

7.32.2.4 bool reloadfas (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 175 of file fedoraproject.cpp.

References BotKernel::getDatasDir(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), FedoraProject::loadFasFile(), LogFile::log(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and WARNING.

7.32.2.5 bool whoowns (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 159 of file fedoraproject.cpp.

References Message::getPart(), Message::getSource(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendMsg(), and FedoraProject::whoowns().

7.33 src/plugins/fedoraproject.h File Reference

FedoraProject header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "admin.h"
#include "curl/curl.h"
#include <fstream>
#include <map>
#include <iostream>
```

Classes

- class FedoraProject

  Plugin in connection with fedora project.

7.33.1 Detailed Description

FedoraProject header file.

Definition in file fedoraproject.h.

7.34 src/plugins/gameserver.cpp File Reference

GameServer implementation file.

```
#include "gameserver.h"
```

Functions

- Plugin ∗ contruct_gameserver (BotKernel ∗b)
- void destroy_gameserver (Plugin ∗p)
- bool q3 (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool warsow (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool hl (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.34.1 Detailed Description

GameServer implementation file.

Definition in file gameserver.cpp.

7.34.2 Function Documentation

7.34.2.1 Plugin∗ contruct_gameserver (BotKernel ∗ *b*)

Definition at line 332 of file gameserver.cpp.

7.34.2.2 void destroy_gameserver (Plugin ∗ *p*)

Definition at line 336 of file gameserver.cpp.

7.34.2.3 bool hl (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 410 of file gameserver.cpp.

References GameServer::getHL1Challenge(), GameServer::getHL1Infos(), GameServer::getHL1Players(), Message::getPart(), GameServer::getResult(), Message::getSource(), Tools::intToStr(), Message::isPublic(), MAX_CHARS, Message::nbParts(), BotKernel::send(), IRCProtocol::sendMsg(), GameServer::sendQuery(), Tools::stringToVector(), and Tools::vectorToString().

7.34.2.4 bool q3 (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 340 of file gameserver.cpp.

References Message::getPart(), GameServer::getQ3GameType(), GameServer::getResult(), Message::getSource(), Tools::intToStr(), Message::isPublic(), MAX_CHARS, Message::nbParts(), Tools::parseQ3Colors(), GameServer::parseQ3infos(), BotKernel::send(), IRCProtocol::sendMsg(), GameServer::sendQuery(), Tools::stringToVector(), and Tools::vectorToString().

7.34.2.5 bool warsow (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 375 of file gameserver.cpp.

References Message::getPart(), GameServer::getResult(), Message::getSource(), Tools::intToStr(), Message::isPublic(), MAX_CHARS, Message::nbParts(), Tools::parseQ3Colors(), GameServer::parseWSWinfos(), BotKernel::send(), IRCProtocol::sendMsg(), GameServer::sendQuery(), Tools::stringToVector(), and Tools::vectorToString().

7.35 src/plugins/gameserver.h File Reference

GameServer header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
#include <map>
```

Classes

- class GameServer

  Provides tools to query game servers.

Variables

- const int MAX_CHARS = 1000

  Maximum chars for server's answer.

7.35.1 Detailed Description

GameServer header file.

Definition in file gameserver.h.

7.35.2 Variable Documentation

7.35.2.1 const int MAX_CHARS = 1000

Maximum chars for server's answer.

Definition at line 38 of file gameserver.h.

Referenced by GameServer::getResult(), hl(), q3(), and warsow().

7.36 src/plugins/ignore.cpp File Reference

Ignore implementation file.

```
#include "ignore.h"
```

Functions

- Plugin ∗ contruct_ignore (BotKernel ∗b)
- void destroy_ignore (Plugin ∗p)
- bool isIgnored (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool addIgnore (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool delIgnore (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool ignoreList (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool purifyList (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool testIgnoredUser (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.36.1 Detailed Description

Ignore implementation file.

Definition in file ignore.cpp.

7.36.2 Function Documentation

7.36.2.1 bool addIgnore (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 240 of file ignore.cpp.

References Ignore::addIgnore(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strtimeToSeconds().

7.36.2.2 Plugin∗ contruct_ignore (BotKernel ∗ *b*)

Definition at line 183 of file ignore.cpp.

7.36.2.3 bool delIgnore (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 293 of file ignore.cpp.

References Ignore::delIgnore(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSplit(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.36.2.4 void destroy_ignore (Plugin ∗ *p*)

Definition at line 187 of file ignore.cpp.

7.36.2.5 bool ignoreList (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 339 of file ignore.cpp.

References Ignore::getIgnoreList(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotices().

7.36.2.6 bool isIgnored (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 191 of file ignore.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSplit(), Ignore::isIgnored(), Message::isPrivate(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotice().

7.36.2.7 bool purifyList (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 378 of file ignore.cpp.

References Ignore::purifyList().

7.36.2.8 bool testIgnoredUser (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 384 of file ignore.cpp.

References Message::getPart(), Message::getSender(), and Ignore::isIgnored().

7.37 src/plugins/ignore.h File Reference

Ignore header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../tinyxml/tinyxml.h"
#include "admin.h"
#include <iostream>
```

Classes

- class Ignore

  Manage ignores.

7.37.1 Detailed Description

Ignore header file.

Definition in file ignore.h.

7.38 src/plugins/infos.cpp File Reference

Infos implementation file.

```
#include "infos.h"
```

Functions

- Plugin ∗ contruct_infos (BotKernel ∗b)
- void destroy_infos (Plugin ∗p)
- bool uptime (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool version (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool sysinfos (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool online (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool prefix (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool help (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.38.1 Detailed Description

Infos implementation file.

Definition in file infos.cpp.

7.38.2 Function Documentation

7.38.2.1 Plugin∗ contruct_infos (BotKernel ∗ *b*)

Definition at line 51 of file infos.cpp.

7.38.2.2 void destroy_infos (Plugin ∗ *p*)

Definition at line 55 of file infos.cpp.

7.38.2.3 bool help (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 123 of file infos.cpp.

References BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), ConfigurationFile::getValue(), BotKernel::send(), and IRCProtocol::sendNotice().

7.38.2.4 bool online (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 99 of file infos.cpp.

References Message::getNickSender(), BotKernel::getStartOnline(), Tools::intToStr(), BotKernel::send(), and IRCProtocol::sendNotice().

7.38.2.5 bool prefix (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 117 of file infos.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), ConfigurationFile::getValue(), BotKernel::send(), and IRCProtocol::sendNotice().

Referenced by launchSurvey().

7.38.2.6 bool sysinfos (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 82 of file infos.cpp.

References BotKernel::getDatasDir(), Message::getNickSender(), BotKernel::send(), and IRCProtocol::sendNotice().

7.38.2.7 bool uptime (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 59 of file infos.cpp.

References Message::getNickSender(), BotKernel::getStartTime(), Tools::intToStr(), BotKernel::send(), and IRCProtocol::sendNotice().

7.38.2.8 bool version (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 77 of file infos.cpp.

References Message::getNickSender(), BotKernel::getVersion(), BotKernel::send(), and IRCProtocol::sendNotice().

7.39 src/plugins/infos.h File Reference

Infos header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class Infos

  Give infos about kernel.

7.39.1 Detailed Description

Infos header file.

Definition in file infos.h.

7.40 src/plugins/ipconverting.cpp File Reference

IpConverting implementation file.

```
#include "ipconverting.h"
```

Functions

- Plugin ∗ contruct_ipconverting (BotKernel ∗b)
- void destroy_ipconverting (Plugin ∗p)
- bool host2ip (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool ip2host (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.40.1 Detailed Description

IpConverting implementation file.

Definition in file ipconverting.cpp.

7.40.2 Function Documentation

7.40.2.1 Plugin∗ contruct_ipconverting (BotKernel ∗ *b*)

Definition at line 46 of file ipconverting.cpp.

7.40.2.2 void destroy_ipconverting (Plugin ∗ *p*)

Definition at line 50 of file ipconverting.cpp.

7.40.2.3 bool host2ip (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 54 of file ipconverting.cpp.

References Message::getPart(), Message::getSource(), Message::getSplit(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendMsg().

7.40.2.4 bool ip2host (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 77 of file ipconverting.cpp.

References Message::getPart(), Message::getSource(), Message::getSplit(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendMsg().

7.41 src/plugins/ipconverting.h File Reference

IpConverting header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class IpConverting

  Tools for IP converting.

7.41.1 Detailed Description

IpConverting header file.

Definition in file ipconverting.h.

7.42 src/plugins/lamoule.cpp File Reference

Lamoule implementation file.

```
#include "lamoule.h"
```

Functions

- Plugin ∗ contruct_lamoule (BotKernel ∗b)
- void destroy_lamoule (Plugin ∗p)
- bool lamoule (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topshot (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool nextscore (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool increase (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool player (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool deleteplayer (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool top5 (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool toptotal (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool purifyFile (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.42.1 Detailed Description

Lamoule implementation file.

Definition in file lamoule.cpp.

7.42.2 Function Documentation

7.42.2.1 Plugin∗ contruct_lamoule (BotKernel ∗ *b*)

Definition at line 375 of file lamoule.cpp.

7.42.2.2 bool deleteplayer (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 545 of file lamoule.cpp.

References Lamoule::deletePlayer(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::isPublic(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotice().

7.42.2.3 void destroy_lamoule (Plugin ∗ *p*)

Definition at line 379 of file lamoule.cpp.

7.42.2.4 bool increase (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 512 of file lamoule.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Lamoule::increaseScore(), Message::isPublic(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.42.2.5 bool lamoule (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 383 of file lamoule.cpp.

References Lamoule::generateScore(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSource(), UsersInfos::getUsers(), ConfigurationFile::getValue(), Lamoule::increaseScore(), Tools::intToStr(), Message::isPublic(), nick(), pPlugin::object, Tools::random(), BotKernel::send(), IRCProtocol::sendMsg(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.42.2.6 bool nextscore (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 495 of file lamoule.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::isPublic(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), Lamoule::setNextScore(), and Tools::strToInt().

7.42.2.7 bool player (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 529 of file lamoule.cpp.

References Tools::doubleToStr(), BotKernel::getCONFF(), Lamoule::getInfosPlayer(), Plugin::getName(), Message::getPart(), Message::getSource(), ConfigurationFile::getValue(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendMsg(), Tools::strToDouble(), and Tools::strToInt().

7.42.2.8 bool purifyFile (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 586 of file lamoule.cpp.

References BotKernel::getCONFF(), Plugin::getName(), ConfigurationFile::getValue(), Lamoule::purifyFile(), and Tools::strToInt().

7.42.2.9 bool top5 (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 562 of file lamoule.cpp.

References AVERAGE, Lamoule::get5first(), BotKernel::getCONFF(), Plugin::getName(), Message::getSource(), ConfigurationFile::getValue(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::strToInt().

7.42.2.10 bool topshot (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 486 of file lamoule.cpp.

References Message::getSource(), Lamoule::getTopShot(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendMsg().

7.42.2.11 bool toptotal (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 574 of file lamoule.cpp.

References Lamoule::get5first(), BotKernel::getCONFF(), Plugin::getName(), Message::getSource(), ConfigurationFile::getValue(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendMsg(), Tools::strToInt(), and TOTAL.

7.43 src/plugins/lamoule.h File Reference

Lamoule header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../tinyxml/tinyxml.h"
#include "admin.h"
#include "usersinfos.h"
#include <iostream>
```

Classes

- class Lamoule

  Manage lamoule's ladder.

Enumerations

- enum sort_criterion { TOTAL, AVERAGE }

  Available sort criterions.

7.43.1 Detailed Description

Lamoule header file.

Definition in file lamoule.h.

7.43.2 Enumeration Type Documentation

7.43.2.1 enum sort_criterion

Available sort criterions.

Enumerator:

TOTAL

AVERAGE

Definition at line 47 of file lamoule.h.

7.44 src/plugins/logfactory.cpp File Reference

LogFactory implementation file.

```
#include "logfactory.h"
```

Functions

- Plugin ∗ contruct_logfactory (BotKernel ∗b)
- void destroy_logfactory (Plugin ∗p)
- bool greplog (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool lastseen (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool joinHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topicJoin (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topicInfos (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool partHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool quitHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool nickHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topicHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool kickHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool modeHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool privmsgHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool cleanLogs (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool sendHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.44.1 Detailed Description

LogFactory implementation file.

Definition in file logfactory.cpp.

7.44.2 Function Documentation

7.44.2.1 bool cleanLogs (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 408 of file logfactory.cpp.

7.44.2.2 Plugin∗ contruct_logfactory (BotKernel ∗ *b*)

Definition at line 194 of file logfactory.cpp.

7.44.2.3 void destroy_logfactory (Plugin ∗ *p*)

Definition at line 198 of file logfactory.cpp.

7.44.2.4 bool greplog (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 202 of file logfactory.cpp.

References BotKernel::getDatasDir(), Message::getSource(), Message::getSplit(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::vectorToString().

Referenced by lastseen().

7.44.2.5 bool joinHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 254 of file logfactory.cpp.

References BotKernel::getNick(), Message::getNickSender(), Message::getSender(), and Message::getSource().

7.44.2.6 bool kickHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 347 of file logfactory.cpp.

References BotKernel::getNick(), Message::getNickSender(), Message::getPart(), Message::getSource(), Message::getSplit(), and Tools::vectorToString().

7.44.2.7 bool lastseen (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 228 of file logfactory.cpp.

References BotKernel::getDatasDir(), Message::getPart(), Message::getSource(), greplog(), Message::isPublic(), Message::nbParts(), BotKernel::send(), and IRCProtocol::sendMsg().

7.44.2.8 bool modeHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 359 of file logfactory.cpp.

References Message::getNickSender(), Message::getPart(), and Message::getSource().

7.44.2.9 bool nickHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 314 of file logfactory.cpp.

References BotKernel::getNick(), Message::getNickSender(), and Message::getPart().

7.44.2.10 bool partHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 289 of file logfactory.cpp.

References BotKernel::getNick(), Message::getNickSender(), Message::getSender(), Message::getSource(), Message::getSplit(), Message::nbParts(), and Tools::vectorToString().

7.44.2.11 bool privmsgHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 387 of file logfactory.cpp.

References Message::getNickSender(), Message::getPart(), Message::getSource(), Message::getSplit(), Message::isPublic(), and Tools::vectorToString().

7.44.2.12 bool quitHandler (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 301 of file logfactory.cpp.

References UsersInfos::getLastQuitChannels(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSplit(), pPlugin::object, and Tools::vectorToString().

7.44.2.13 bool sendHandler (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 413 of file logfactory.cpp.

References BotKernel::getNick(), Message::getPart(), Message::getSplit(), and Tools::vectorToString().

7.44.2.14 bool topicHandler (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 340 of file logfactory.cpp.

References Message::getNickSender(), Message::getSource(), Message::getSplit(), and Tools::vectorToString().

7.44.2.15 bool topicInfos (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 280 of file logfactory.cpp.

References Message::getPart(), and Tools::strToUnsignedInt().

7.44.2.16 bool topicJoin (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 273 of file logfactory.cpp.

References Message::getPart(), Message::getSplit(), and Tools::vectorToString().

7.45 src/plugins/logfactory.h File Reference

LogFactory header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../channel.h"
#include "usersinfos.h"
#include <iostream>
#include <map>
```

Classes

- class LogFactory

  This plugin manage channels logging.

7.45.1 Detailed Description

LogFactory header file.

Definition in file logfactory.h.

7.46 src/plugins/magic8ball.cpp File Reference

Magic8Ball implementation file.

```
#include "magic8ball.h"
```

Functions

- Plugin ∗ contruct_magic8ball (BotKernel ∗b)
- void destroy_magic8ball (Plugin ∗p)
- bool ball (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.46.1 Detailed Description

Magic8Ball implementation file.

Definition in file magic8ball.cpp.

7.46.2 Function Documentation

7.46.2.1 bool ball (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 82 of file magic8ball.cpp.

References Magic8Ball::getRandomAnswer(), Message::getSource(), Message::isPublic(), Message::nbParts(), BotKernel::send(), and IRCProtocol::sendMsg().

7.46.2.2 Plugin∗ contruct_magic8ball (BotKernel ∗ *b*)

Definition at line 74 of file magic8ball.cpp.

7.46.2.3 void destroy_magic8ball (Plugin ∗ *p*)

Definition at line 78 of file magic8ball.cpp.

7.47 src/plugins/magic8ball.h File Reference

Magic8Ball header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class Magic8Ball

  magic 8 ball game

7.47.1 Detailed Description

Magic8Ball header file.

Definition in file magic8ball.h.

7.48 src/plugins/moderation.cpp File Reference

Moderation implementation file.

```
#include "moderation.h"
```

Functions

- Plugin ∗ contruct_moderation (BotKernel ∗b)
- void destroy_moderation (Plugin ∗p)
- bool unbanall (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool bandel (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool baninfos (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool banlist (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool ban (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool banmask (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool op (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unop (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool voice (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unvoice (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topic (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool kick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool masskick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool opall (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unopall (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool voiceall (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unvoiceall (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool kickall (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool randomKick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool modeHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool modeHandlerProtect (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool joinHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool partHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool quitHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool kickHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool autoop (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unautoop (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool autovoice (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unautovoice (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool protecttopic (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unprotecttopic (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool protectmodes (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unprotectmodes (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool clearOutBans (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool invite (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool rejoinChan (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool bannedHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topicJoin (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool topicHandler (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.48.1 Detailed Description

Moderation implementation file.

Definition in file moderation.cpp.

7.48.2 Function Documentation

7.48.2.1 bool autoop (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 1032 of file moderation.cpp.

References BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), Tools::isInVector(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), and Tools::stringToVector().

7.48.2.2 bool autovoice (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 1069 of file moderation.cpp.

References BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), Tools::isInVector(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), and Tools::stringToVector().

7.48.2.3 bool ban (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 526 of file moderation.cpp.

References Moderation::addBan(), IRCProtocol::ban(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), Message::getSplit(), UsersInfos::getUsers(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), Message::isPublic(), IRCProtocol::kick(), Message::nbParts(), pPlugin::object, BotKernel::send(), Tools::strtimeToSeconds(), and Tools::vectorToString().

7.48.2.4 bool bandel (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 492 of file moderation.cpp.

References Moderation::delBan(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Moderation::hasOpPrivileges(), Message::isPublic(), Message::nbParts(), BotKernel::send(), Tools::strToInt(), and IRCProtocol::unban().

7.48.2.5 bool baninfos (Message $*$ *m*, Plugin $*$ *p*, BotKernel $*$ *b*)

Definition at line 506 of file moderation.cpp.

References Moderation::banInfos(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Moderation::hasOpPrivileges(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendNotices(), and Tools::strToInt().

7.48.2.6 bool banlist (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 516 of file moderation.cpp.

References Tools::gatherVectorElements(), Moderation::getBanList(), Message::getNickSender(), Message::getSender(), Message::getSource(), Moderation::hasOpPrivileges(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendNotices().

7.48.2.7 bool banmask (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 561 of file moderation.cpp.

References Moderation::addBan(), IRCProtocol::ban(), Moderation::getChanUsersList(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), Tools::ircMaskMatch(), Message::isPublic(), IRCProtocol::kick(), Message::nbParts(), BotKernel::send(), Tools::strtimeToSeconds(), and Tools::vectorToString().

7.48.2.8 bool bannedHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1245 of file moderation.cpp.

References BotKernel::addCountDown(), BotKernel::getCONFF(), Plugin::getName(), Message::getPart(), BotKernel::getSysLog(), ConfigurationFile::getValue(), INFO, LogFile::log(), rejoinChan(), and Tools::strToUnsignedInt().

7.48.2.9 bool clearOutBans (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1194 of file moderation.cpp.

References Moderation::checkMode(), Moderation::clearOutBans(), BotKernel::getNick(), BotKernel::getPlugin(), UsersInfos::getUsers(), pPlugin::object, and BotKernel::send().

7.48.2.10 Plugin∗ contruct_moderation (BotKernel ∗ *b*)

Definition at line 472 of file moderation.cpp.

7.48.2.11 void destroy_moderation (Plugin ∗ *p*)

Definition at line 476 of file moderation.cpp.

7.48.2.12 bool invite (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1219 of file moderation.cpp.

References Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Admin::getUserLevel(), IRCProtocol::invite(), Message::isPrivate(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, and BotKernel::send().

7.48.2.13 bool joinHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 908 of file moderation.cpp.

References IRCProtocol::ban(), Moderation::clearRejoinAttempts(), BotKernel::getCONFF(), Message::getHostSender(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), Admin::getUserLevel(), ConfigurationFile::getValue(), Moderation::isBanned(), Tools::isInVector(), IRCProtocol::kick(), nick(), pPlugin::object, IRCProtocol::op(), BotKernel::send(), Tools::stringToVector(), and IRCProtocol::voice().

7.48.2.14 bool kick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 670 of file moderation.cpp.

References BotKernel::getNick(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), Moderation::hasOpPrivileges(), Message::isPublic(), IRCProtocol::kick(), BotKernel::send(), and Tools::vectorToString().

7.48.2.15 bool kickall (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 775 of file moderation.cpp.

References Moderation::getChanUsersList(), BotKernel::getNick(), Message::getNickSender(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), IRCProtocol::kick(), LogFile::log(), and BotKernel::send().

7.48.2.16 bool kickHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 984 of file moderation.cpp.

References Moderation::checkMode(), BotKernel::getCONFF(), Message::getHostSender(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), Admin::getUserLevel(), UsersInfos::getUsers(), ConfigurationFile::getValue(), INFO, Admin::isSuperAdmin(), IRCProtocol::joinChannel(), IRCProtocol::kick(), IRCProtocol::leaveChannel(), LogFile::log(), pPlugin::object, and BotKernel::send().

7.48.2.17 bool masskick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 682 of file moderation.cpp.

References BotKernel::getNick(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), BotKernel::getSysLog(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), IRCProtocol::kick(), LogFile::log(), and BotKernel::send().

7.48.2.18 bool modeHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 817 of file moderation.cpp.

References BotKernel::getCONFF(), Message::getHostSender(), Admin::getMaskLevel(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), Message::getSplit(), Admin::getUserLevel(), UsersInfos::getUsers(), ConfigurationFile::getValue(), Admin::isSuperAdmin(), IRCProtocol::kick(), Admin::maskIsSuperAdmin(), pPlugin::object, and BotKernel::send().

7.48.2.19 bool modeHandlerProtect (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 873 of file moderation.cpp.

References Moderation::checkAccess(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), UsersInfos::getPrefixes(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Tools::isInVector(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), and Tools::stringToVector().

7.48.2.20 bool op (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 585 of file moderation.cpp.

References Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), Moderation::hasOpPrivileges(), Message::isPublic(), IRCProtocol::op(), and BotKernel::send().

7.48.2.21 bool opall (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 698 of file moderation.cpp.

References Moderation::checkMode(), Moderation::getChanUsersList(), Message::getNickSender(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), LogFile::log(), IRCProtocol::op(), and BotKernel::send().

7.48.2.22 bool partHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 942 of file moderation.cpp.

References Moderation::checkMode(), BotKernel::getNick(), BotKernel::getPlugin(), Message::getSource(), UsersInfos::getUsers(), IRCProtocol::joinChannel(), IRCProtocol::leaveChannel(), pPlugin::object, and BotKernel::send().

7.48.2.23 bool protectmodes (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1150 of file moderation.cpp.

References Moderation::checkAccess(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Tools::isInVector(), Message::isPublic(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), and Tools::stringToVector().

7.48.2.24 bool protecttopic (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1106 of file moderation.cpp.

References Moderation::checkAccess(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Tools::isInVector(), Message::isPublic(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), and Tools::stringToVector().

7.48.2.25 bool quitHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 964 of file moderation.cpp.

References Moderation::checkMode(), BotKernel::getNick(), BotKernel::getPlugin(), UsersInfos::getUsers(), IRCProtocol::joinChannel(), IRCProtocol::leaveChannel(), pPlugin::object, and BotKernel::send().

7.48.2.26 bool randomKick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 794 of file moderation.cpp.

References Moderation::getChanUsersList(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), IRCProtocol::kick(), LogFile::log(), nick(), Tools::random(), BotKernel::send(), and IRCProtocol::sendMsg().

7.48.2.27 bool rejoinChan (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1231 of file moderation.cpp.

References Moderation::bumpRejoinAttempts(), BotKernel::getCONFF(), Message::getMessage(), Plugin::getName(), Moderation::getRejoinAttempts(), BotKernel::getSysLog(), ConfigurationFile::getValue(), IRCProtocol::joinChannel(), LogFile::log(), BotKernel::send(), Tools::strToUnsignedInt(), and WARNING.

Referenced by bannedHandler().

7.48.2.28 bool topic (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 659 of file moderation.cpp.

References IRCProtocol::changeTopic(), Message::getNickSender(), Message::getSender(), Message::getSource(), Message::getSplit(), Moderation::hasOpPrivileges(), Message::isPublic(), BotKernel::send(), and Tools::vectorToString().

7.48.2.29 bool topicHandler (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1271 of file moderation.cpp.

References IRCProtocol::changeTopic(), Moderation::checkAccess(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getNick(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), Message::getSplit(), UsersInfos::getUsers(), ConfigurationFile::getValue(), Tools::isInVector(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), Tools::stringToVector(), and Tools::vectorToString().

7.48.2.30 bool topicJoin (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1256 of file moderation.cpp.

References Message::getPart(), BotKernel::getPlugin(), Message::getSplit(), UsersInfos::getUsers(), pPlugin::object, and Tools::vectorToString().

7.48.2.31 bool unautoop (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1049 of file moderation.cpp.

References Tools::delStrFromVector(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), Tools::isInVector(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), Tools::stringToVector(), and Tools::vectorToString().

7.48.2.32 bool unautovoice (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1086 of file moderation.cpp.

References Tools::delStrFromVector(), BotKernel::getCONFF(), Plugin::getName(), Message::getNickSender(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Moderation::hasOpPrivileges(), Tools::isInVector(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), Tools::stringToVector(), and Tools::vectorToString().

7.48.2.33 bool unbanall (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 480 of file moderation.cpp.

References IRCProtocol::applyModes(), Moderation::clearList(), Message::getNickSender(), Message::getSender(), Message::getSource(), Moderation::hasOpPrivileges(), Message::isPublic(), and BotKernel::send().

7.48.2.34 bool unop (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 603 of file moderation.cpp.

References BotKernel::getNick(), Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), Moderation::hasOpPrivileges(), Message::isPublic(), BotKernel::send(), and IRCProtocol::unop().

7.48.2.35 bool unopall (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 717 of file moderation.cpp.

References Moderation::checkMode(), Moderation::getChanUsersList(), BotKernel::getNick(), Message::getNickSender(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), LogFile::log(), BotKernel::send(), and IRCProtocol::unop().

7.48.2.36 bool unprotectmodes (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1170 of file moderation.cpp.

References Moderation::checkAccess(), Tools::delStrFromVector(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Tools::isInVector(), Message::isPublic(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), Tools::stringToVector(), and Tools::vectorToString().

7.48.2.37 bool unprotecttopic (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 1126 of file moderation.cpp.

References Moderation::checkAccess(), Tools::delStrFromVector(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), ConfigurationFile::getValue(), Tools::isInVector(), Message::isPublic(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), IRCProtocol::sendMsg(), ConfigurationFile::setValue(), Tools::stringToVector(), and Tools::vectorToString().

7.48.2.38 bool unvoice (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 641 of file moderation.cpp.

References Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), Moderation::hasOpPrivileges(), Message::isPublic(), BotKernel::send(), and IRCProtocol::unvoice().

7.48.2.39 bool unvoiceall (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 756 of file moderation.cpp.

References Moderation::checkMode(), Moderation::getChanUsersList(), Message::getNickSender(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), LogFile::log(), BotKernel::send(), and IRCProtocol::unvoice().

7.48.2.40 bool voice (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 623 of file moderation.cpp.

References Message::getNickSender(), Message::getPart(), Message::getSender(), Message::getSource(), Message::getSplit(), Moderation::hasOpPrivileges(), Message::isPublic(), BotKernel::send(), and IRCProtocol::voice().

7.48.2.41 bool voiceall (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 737 of file moderation.cpp.

References Moderation::checkMode(), Moderation::getChanUsersList(), Message::getNickSender(), Message::getSender(), Message::getSource(), BotKernel::getSysLog(), Moderation::hasOpPrivileges(), INFO, Message::isPublic(), LogFile::log(), BotKernel::send(), and IRCProtocol::voice().

7.49 src/plugins/moderation.h File Reference

Moderation header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../tinyxml/tinyxml.h"
#include "admin.h"
#include "usersinfos.h"
#include <time.h>
#include <map>
```

Classes

- class Moderation

  Channel moderation.

7.49.1 Detailed Description

Moderation header file.

Definition in file moderation.h.

7.50 src/plugins/module.cpp File Reference

Module implementation file.

```
#include "module.h"
```

Functions

- Plugin ∗ contruct_module (BotKernel ∗b)
- void destroy_module (Plugin ∗p)
- bool load (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unload (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool loadnocheck (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool unloadnocheck (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool listmodules (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool listlibs (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool moduleinfos (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.50.1 Detailed Description

Module implementation file.

Definition in file module.cpp.

7.50.2 Function Documentation

7.50.2.1 Plugin∗ contruct_module (BotKernel ∗ *b*)

Definition at line 52 of file module.cpp.

7.50.2.2 void destroy_module (Plugin ∗ *p*)

Definition at line 56 of file module.cpp.

7.50.2.3 bool listlibs (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 149 of file module.cpp.

References Tools::gatherVectorElements(), BotKernel::getDatasDir(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotices().

7.50.2.4 bool listmodules (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 136 of file module.cpp.

References Tools::gatherVectorElements(), Message::getNickSender(), BotKernel::getPlugin(), BotKernel::getPluginsList(), Message::getSender(), Message::isPrivate(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotices().

7.50.2.5 bool load (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 60 of file module.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::loadPlugin(), LogFile::log(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and WARNING.

7.50.2.6 bool loadnocheck (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 98 of file module.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), BotKernel::loadPlugin(), LogFile::log(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and WARNING.

7.50.2.7 bool moduleinfos (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 180 of file module.cpp.

References Plugin::getAuthor(), Plugin::getDescription(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Plugin::getVersion(), Message::isPrivate(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), and IRCProtocol::sendNotice().

7.50.2.8 bool unload (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 79 of file module.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), BotKernel::unloadPlugin(), and WARNING.

7.50.2.9 bool unloadnocheck (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 117 of file module.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), BotKernel::getSysLog(), INFO, Message::isPrivate(), Admin::isSuperAdmin(), LogFile::log(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), BotKernel::unloadPlugin(), and WARNING.

7.51 src/plugins/module.h File Reference

Module header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "admin.h"
#include <sys/types.h>
#include <dirent.h>
#include <iostream>
```

Classes

- class Module

  Modules management.

7.51.1 Detailed Description

Module header file.

Definition in file module.h.

7.52 src/plugins/ping.cpp File Reference

Ping implementation file.

```
#include "ping.h"
```

Functions

- Plugin ∗ contruct_ping (BotKernel ∗b)
- void destroy_ping (Plugin ∗p)
- bool pinged (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool checkConnection (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool pongMe (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.52.1 Detailed Description

Ping implementation file.

Definition in file ping.cpp.

7.52.2 Function Documentation

7.52.2.1 bool checkConnection (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 81 of file ping.cpp.

References BotKernel::getNick(), Ping::getPonged(), BotKernel::getSysLog(), LogFile::log(), IRCProtocol::ping(), BotKernel::send(), BotKernel::setConnected(), Ping::setPonged(), and WARNING.

7.52.2.2 Plugin∗ contruct_ping (BotKernel ∗ *b*)

Definition at line 68 of file ping.cpp.

7.52.2.3 void destroy_ping (Plugin ∗ *p*)

Definition at line 72 of file ping.cpp.

7.52.2.4 bool pinged (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 76 of file ping.cpp.

References Message::getPart(), IRCProtocol::pong(), and BotKernel::send().

7.52.2.5 bool pongMe (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 96 of file ping.cpp.

References Ping::setPonged().

7.53 src/plugins/ping.h File Reference

Ping header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "time.h"
#include <iostream>
```

Classes

- class Ping

  Manage ping events.

7.53.1 Detailed Description

Ping header file.

Definition in file ping.h.

7.54 src/plugins/pluginsample.cpp File Reference

PluginSample implementation file.

```
#include "pluginsample.h"
```

Functions

- Plugin ∗ contruct_pluginsample (BotKernel ∗b)
- void destroy_pluginsample (Plugin ∗p)
- bool myFunction (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.54.1 Detailed Description

PluginSample implementation file.

Definition in file pluginsample.cpp.

7.54.2 Function Documentation

7.54.2.1 Plugin∗ contruct_pluginsample (BotKernel ∗ *b*)

Definition at line 45 of file pluginsample.cpp.

7.54.2.2 void destroy_pluginsample (Plugin ∗ *p*)

Definition at line 49 of file pluginsample.cpp.

7.54.2.3 bool myFunction (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 53 of file pluginsample.cpp.

References Message::getNickSender(), Message::getSource(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendMsg().

7.55 src/plugins/pluginsample.h File Reference

PluginSample header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class PluginSample

  Plugin class example.

7.55.1 Detailed Description

PluginSample header file.

Definition in file pluginsample.h.

7.56 src/plugins/postconnect.cpp File Reference

PostConnect implementation file.

```
#include "postconnect.h"
```

Functions

- Plugin ∗ contruct_postconnect (BotKernel ∗b)
- void destroy_postconnect (Plugin ∗p)
- bool nick_changed (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool getMyFirstNick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool secondaryNick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool onEndOfMOTD (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.56.1 Detailed Description

PostConnect implementation file.

Definition in file postconnect.cpp.

7.56.2 Function Documentation

7.56.2.1 Plugin∗ contruct_postconnect (BotKernel ∗ *b*)

Definition at line 74 of file postconnect.cpp.

7.56.2.2 void destroy_postconnect (Plugin ∗ *p*)

Definition at line 78 of file postconnect.cpp.

7.56.2.3 bool getMyFirstNick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 90 of file postconnect.cpp.

References PostConnect::bumpNickRetreiveAttempts(), IRCProtocol::changeNick(), BotKernel::getCONFF(), Plugin::getName(), PostConnect::getNickRetreiveAttempts(), ConfigurationFile::getValue(), BotKernel::send(), BotKernel::setNick(), and Tools::strToUnsignedInt().

Referenced by secondaryNick().

7.56.2.4 bool nick_changed (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 82 of file postconnect.cpp.

References BotKernel::getNick(), Message::getPart(), and PostConnect::resetNickRetreiveAttempts().

7.56.2.5 bool onEndOfMOTD (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 118 of file postconnect.cpp.

References Tools::gatherVectorElements(), BotKernel::getCONFF(), Plugin::getName(), BotKernel::getNick(), BotKernel::getSysLog(), ConfigurationFile::getValue(), INFO, IRCProtocol::joinChannel(), LogFile::log(), BotKernel::send(), Tools::stringToVector(), Tools::strToInt(), and WARNING.

7.56.2.6 bool secondaryNick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 105 of file postconnect.cpp.

References BotKernel::addCountDown(), IRCProtocol::changeNick(), BotKernel::getCONFF(), getMyFirstNick(), Plugin::getName(), BotKernel::getSysLog(), ConfigurationFile::getValue(), INFO, LogFile::log(), BotKernel::send(), BotKernel::setNick(), and Tools::strToUnsignedInt().

7.57 src/plugins/postconnect.h File Reference

PostConnect header file.

```
#include "../plugin.h"
#include "../botkernel.h"
```

Classes

- class PostConnect

  Afer connect plugin.

7.57.1 Detailed Description

PostConnect header file.

Definition in file postconnect.h.

7.58 src/plugins/quotes.cpp File Reference

Quotes implementation file.

```
#include "quotes.h"
```

Functions

- Plugin ∗ contruct_quotes (BotKernel ∗b)
- void destroy_quotes (Plugin ∗p)
- bool quote (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool addQuote (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool delQuote (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool searchQuote (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool quoteInfos (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool lastQuote (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.58.1 Detailed Description

Quotes implementation file.

Definition in file quotes.cpp.

7.58.2 Function Documentation

7.58.2.1 bool addQuote (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 246 of file quotes.cpp.

References Quotes::addQuote(), Message::getNickSender(), Message::getSender(), Message::getSplit(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendNotice(), and Tools::vectorToString().

7.58.2.2 Plugin∗ contruct_quotes (BotKernel ∗ *b*)

Definition at line 225 of file quotes.cpp.

7.58.2.3 bool delQuote (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 255 of file quotes.cpp.

References Quotes::delQuote(), Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::isPublic(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.58.2.4 void destroy_quotes (Plugin ∗ *p*)

Definition at line 229 of file quotes.cpp.

7.58.2.5 bool lastQuote (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 294 of file quotes.cpp.

References Quotes::getLastQuote(), Message::getSource(), Message::isPublic(), BotKernel::send(), and IRCProtocol::sendMsg().

7.58.2.6 bool quote (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 233 of file quotes.cpp.

References Message::getPart(), Quotes::getQuote(), Quotes::getRandomQuote(), Message::getSource(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::strToInt().

7.58.2.7 bool quoteInfos (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 281 of file quotes.cpp.

References Message::getNickSender(), Message::getPart(), BotKernel::getPlugin(), Message::getSender(), Message::isPublic(), Admin::isSuperAdmin(), Message::nbParts(), pPlugin::object, Quotes::quoteInfos(), BotKernel::send(), IRCProtocol::sendNotice(), and Tools::strToInt().

7.58.2.8 bool searchQuote (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 273 of file quotes.cpp.

References Message::getSource(), Message::getSplit(), Message::isPublic(), Message::nbParts(), Quotes::searchQuote(), BotKernel::send(), IRCProtocol::sendMsg(), and Tools::vectorToString().

7.59 src/plugins/quotes.h File Reference

Quotes header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../tinyxml/tinyxml.h"
#include "admin.h"
#include <iostream>
```

Classes

- class Quotes

    Quotes management (storage and access).

7.59.1 Detailed Description

Quotes header file.

Definition in file quotes.h.

7.60 src/plugins/remotecontrol.cpp File Reference

RemoteControl implementation file.

`#include "remotecontrol.h"`

Functions

- void ∗ myThread (void ∗)
- Plugin ∗ contruct_remotecontrol (BotKernel ∗b)
- void destroy_remotecontrol (Plugin ∗p)
- bool myUselessFunction (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.60.1 Detailed Description

RemoteControl implementation file.

Definition in file remotecontrol.cpp.

7.60.2 Function Documentation

7.60.2.1 Plugin∗ contruct_remotecontrol (BotKernel ∗ *b*)

Definition at line 194 of file remotecontrol.cpp.

7.60.2.2 void destroy_remotecontrol (Plugin ∗ *p*)

Definition at line 198 of file remotecontrol.cpp.

7.60.2.3 void ∗ myThread (void ∗ *arg*)

Definition at line 202 of file remotecontrol.cpp.

References BotKernel::getPlugin(), pPlugin::object, and RemoteControl::tcpServer().

Referenced by RemoteControl::RemoteControl().

7.60.2.4 bool myUselessFunction (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 213 of file remotecontrol.cpp.

References BotKernel::unregisterFunction().

7.61 src/plugins/remotecontrol.h File Reference

RemoteControl header file.

```
#include "../cppthread.h"
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class RemoteControl

  Plugin that allow remote TCP control.

7.61.1 Detailed Description

RemoteControl header file.

Definition in file remotecontrol.h.

7.62 src/plugins/slapme.cpp File Reference

Slapme implementation file.

```
#include "slapme.h"
```

Functions

- Plugin ∗ contruct_slapme (BotKernel ∗b)
- void destroy_slapme (Plugin ∗p)
- bool slapUser (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool slapme (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.62.1 Detailed Description

Slapme implementation file.

Definition in file slapme.cpp.

7.62.2 Function Documentation

7.62.2.1 Plugin∗ contruct_slapme (BotKernel ∗ *b*)

Definition at line 45 of file slapme.cpp.

7.62.2.2 void destroy_slapme (Plugin ∗ *p*)

Definition at line 49 of file slapme.cpp.

7.62.2.3 bool slapme (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 58 of file slapme.cpp.

References BotKernel::addCountDown(), Message::getNickSender(), Message::getPart(), Tools::intToStr(), Message::isPublic(), Message::nbParts(), BotKernel::send(), IRCProtocol::sendNotice(), slapUser(), and Tools::strtimeToSeconds().

7.62.2.4 bool slapUser (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 53 of file slapme.cpp.

References Message::getNickSender(), Message::getSource(), Message::getSplit(), BotKernel::send(), IRCProtocol::sendAction(), and Tools::vectorToString().

Referenced by slapme().

7.63 src/plugins/slapme.h File Reference

Slapme header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class Slapme

  Plugin used to slap users.

7.63.1 Detailed Description

Slapme header file.

Definition in file slapme.h.

7.64 src/plugins/survey.cpp File Reference

Survey implementation file.

`#include "survey.h"`

Functions

- Plugin ∗ contruct_survey (BotKernel ∗b)
- void destroy_survey (Plugin ∗p)
- bool stopSurvey (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool endSurvey (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool vote (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool launchSurvey (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.64.1 Detailed Description

Survey implementation file.

Definition in file survey.cpp.

7.64.2 Function Documentation

7.64.2.1 Plugin∗ contruct_survey (BotKernel ∗ *b*)

Definition at line 238 of file survey.cpp.

7.64.2.2 void destroy_survey (Plugin ∗ *p*)

Definition at line 242 of file survey.cpp.

7.64.2.3 bool endSurvey (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 269 of file survey.cpp.

References Survey::finishSurvey(), Message::getSource(), Survey::getSurveyFunctions(), BotKernel::send(), IRCProtocol::sendMsg(), and BotKernel::unregisterFunction().

Referenced by launchSurvey().

7.64.2.4 bool launchSurvey (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 290 of file survey.cpp.

References BotKernel::addCountDown(), endSurvey(), BotKernel::getCONFF(), Message::getMessage(), Plugin::getName(), Message::getNickSender(), Message::getSource(), ConfigurationFile::getValue(), IN_COMMAND_HANDLER, Tools::intToStr(), Tools::isInVector(), Message::isPublic(), Survey::launchSurvey(), prefix(), BotKernel::registerFunction(), BotKernel::send(), IRCProtocol::sendMsg(), IRCProtocol::sendNotice(), Survey::setCountDown(), Survey::setSurveyFunctions(), Survey::stopSurvey(), Tools::stringToVector(), Tools::strtimeToSeconds(), Tools::strToUnsignedInt(), Tools::vectorToString(), and vote().

7.64.2.5 bool stopSurvey (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 246 of file survey.cpp.

References Survey::getCountDown(), Message::getNickSender(), BotKernel::getPlugin(), Message::getSender(), Message::getSource(), Survey::getSurveyFunctions(), Message::isPublic(), Admin::isSuperAdmin(), pPlugin::object, BotKernel::send(), IRCProtocol::sendMsg(), IRCProtocol::sendNotice(), Survey::stopSurvey(), and BotKernel::unregisterFunction().

7.64.2.6 bool vote (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 280 of file survey.cpp.

References BotKernel::getCONFF(), Message::getNickSender(), Message::getPart(), Message::getSource(), ConfigurationFile::getValue(), Message::isPublic(), BotKernel::send(), IRCProtocol::sendNotice(), and Survey::vote().

Referenced by launchSurvey().

7.65 src/plugins/survey.h File Reference

Survey header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "admin.h"
#include <iostream>
```

Classes

- struct struct_survey

  Plugin object and header storage.

- class Survey

  This plugin manages surveys.

Defines

- #define CLASS_H

7.65.1 Detailed Description

Survey header file.

Definition in file survey.h.

7.65.2 Define Documentation

7.65.2.1 #define CLASS_H

Definition at line 30 of file survey.h.

7.66 src/plugins/tele.cpp File Reference

Tele implementation file.

```
#include "tele.h"
```

Functions

- Plugin ∗ contruct_tele (BotKernel ∗b)
- void destroy_tele (Plugin ∗p)
- bool tele (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.66.1 Detailed Description

Tele implementation file.

Definition in file tele.cpp.

7.66.2 Function Documentation

7.66.2.1 Plugin∗ contruct_tele (BotKernel ∗ *b*)

Definition at line 46 of file tele.cpp.

7.66.2.2 void destroy_tele (Plugin ∗ *p*)

Definition at line 50 of file tele.cpp.

7.66.2.3 bool tele (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 54 of file tele.cpp.

References Tools::cleanHTML(), Tools::clearAccents(), Socket::connectSock(), Message::getSource(), Message::isPublic(), Socket::receive(), BotKernel::send(), IRCProtocol::sendMsg(), and Socket::sendStr().

7.67 src/plugins/tele.h File Reference

Tele header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class Tele

  Display french TV program.

7.67.1 Detailed Description

Tele header file.

Definition in file tele.h.

7.68 src/plugins/trad.cpp File Reference

Trad implementation file.

```
#include "trad.h"
```

Functions

- Plugin ∗ contruct_trad (BotKernel ∗b)
- void destroy_trad (Plugin ∗p)
- bool trad (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.68.1 Detailed Description

Trad implementation file.

Definition in file trad.cpp.

7.68.2 Function Documentation

7.68.2.1 Plugin∗ contruct_trad (BotKernel ∗ *b*)

Definition at line 45 of file trad.cpp.

7.68.2.2 void destroy_trad (Plugin ∗ *p*)

Definition at line 49 of file trad.cpp.

7.68.2.3 bool trad (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 53 of file trad.cpp.

References Tools::cleanHTML(), Socket::connectSock(), Message::getPart(), Message::getSource(), Message::getSplit(), Message::isPublic(), Message::nbParts(), Socket::receive(), BotKernel::send(), IRCProtocol::sendMsg(), Socket::sendStr(), Tools::urlencode(), and Tools::vectorToString().

7.69 src/plugins/trad.h File Reference

Trad header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include <iostream>
```

Classes

- class Trad

  Provides a command to translate a sentence from a language to an other using translate.google.com.

7.69.1 Detailed Description

Trad header file.

Definition in file trad.h.

7.70 src/plugins/usersinfos.cpp File Reference

UsersInfos implementation file.

```
#include "usersinfos.h"
```

Functions

- Plugin ∗ contruct_usersinfos (BotKernel ∗b)
- void destroy_usersinfos (Plugin ∗p)
- bool reloadUsers (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool onJoin (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool onPart (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool onQuit (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool onKick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool mode (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool nick (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool event352 (Message ∗m, Plugin ∗p, BotKernel ∗b)
- bool event005 (Message ∗m, Plugin ∗p, BotKernel ∗b)

7.70.1 Detailed Description

UsersInfos implementation file.

Definition in file usersinfos.cpp.

7.70.2 Function Documentation

7.70.2.1 Plugin∗ contruct_usersinfos (BotKernel ∗ *b*)

Definition at line 159 of file usersinfos.cpp.

7.70.2.2 void destroy_usersinfos (Plugin ∗ *p*)

Definition at line 163 of file usersinfos.cpp.

7.70.2.3 bool event005 (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 321 of file usersinfos.cpp.

References UsersInfos::addPrefixe(), and Message::getSplit().

7.70.2.4 bool event352 (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 311 of file usersinfos.cpp.

References Message::getPart().

7.70.2.5 bool mode (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 273 of file usersinfos.cpp.

References Message::getPart(), UsersInfos::getPrefixe(), Message::getSource(), and Message::getSplit().

7.70.2.6 bool nick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 301 of file usersinfos.cpp.

References Message::getNickSender(), and Message::getSource().

Referenced by joinHandler(), lamoule(), and randomKick().

7.70.2.7 bool onJoin (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 191 of file usersinfos.cpp.

References Message::getHostSender(), Message::getIdentSender(), BotKernel::getNick(), Message::getNickSender(), Message::getSource(), BotKernel::send(), and IRCProtocol::who().

7.70.2.8 bool onKick (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 255 of file usersinfos.cpp.

References BotKernel::getNick(), Message::getPart(), and Message::getSource().

7.70.2.9 bool onPart (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 223 of file usersinfos.cpp.

References BotKernel::getNick(), Message::getNickSender(), and Message::getSource().

7.70.2.10 bool onQuit (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 241 of file usersinfos.cpp.

References UsersInfos::getLastQuitChannels(), and Message::getNickSender().

7.70.2.11 bool reloadUsers (Message ∗ *m*, Plugin ∗ *p*, BotKernel ∗ *b*)

Definition at line 167 of file usersinfos.cpp.

References BotKernel::send(), and IRCProtocol::who().

7.71 src/plugins/usersinfos.h File Reference

UsersInfos header file.

```
#include "../plugin.h"
#include "../botkernel.h"
#include "../channel.h"
#include <string>
#include <map>
```

Classes

- class UsersInfos

  Follow users modes on channels.

7.71.1 Detailed Description

UsersInfos header file.

Definition in file usersinfos.h.

7.72 src/socket.cpp File Reference

Socket implementation file.

```
#include "socket.h"
```

7.72.1 Detailed Description

Socket implementation file.

Definition in file socket.cpp.

7.73 src/socket.h File Reference

Socket header file.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <errno.h>
#include <string.h>
#include <iostream>
#include <string>
#include "tools.h"
```

Classes

- class Socket

  Class that manage the connection with the server.

7.73.1 Detailed Description

Socket header file.

Definition in file socket.h.

7.74 src/tools.cpp File Reference

Tools implementation file.

```
#include "tools.h"
#include <iostream>
```

Functions

- bool copyFile (string source, string destination)

7.74.1 Detailed Description

Tools implementation file.

Definition in file tools.cpp.

7.74.2 Function Documentation

7.74.2.1 bool copyFile (string *source*, string *destination*)

Copy a file from a source to a destination

Parameters:

source Source file (to copy)

destination Destination file

Returns:

true if copy ok, else false

Definition at line 565 of file tools.cpp.

7.75 src/tools.h File Reference

Tools header file.

```
#include <time.h>
#include <sstream>
#include <fstream>
#include <vector>
#include <string>
#include <stdlib.h>
#include <fnmatch.h>
```

Classes

- class Tools

  Class that provides tools for programmation.

7.75.1 Detailed Description

Tools header file.

Definition in file tools.h.

Index