Whatsapp bulk messanger

Version1.0

Generated by Doxygen 1.8.5

Wed Oct 23 2013 12:48:43

Contents

1 PHP

Routing

Routing is defined within index.php file

To modify routing refer to `http://docs.slimframework.com/#Routing-Overview`

Authentication

Authentication is enable within index.php file with Slim middleware and Strong package

It uses standard Slim::Extras::Middleware::StrongAuth / PDO mechanism

User data is stored in users table using UserDAO class

Routes that requires authentication are:

- /groups
- /senders
- /messages
- /users

To modify login/logout controller use GitGis::Auth::AuthController class

Main program

Messenger is done using MVC architecture

- Controlers are stored in GitGis::Whatsapp package
- DAO and entites are stored in GitGis::Whatsapp::Model package

2 Database

Table senders

Stores sender information

- id - Id of sender
- username - MSISDN/mobile number (UID) of group
- identity - Whatsapp identity
- nickname - Human friendly name
- password - Generated during registration
- user_id - Owner of the group

Table groups

Stores group information

- id - Id of group
- nickname - Human friendly name
- user_id - Owner of the group

Table numbers

Stores group's receivers numbers

- group_id - Id of group
- target - number in format 44123123123 (only numbers)
- nickname - Human friendly name
- synced - 1 if number exists in whatsapp

Table messages

Stores messages

- id - Id of message
- group_id - Id of group
- sender_id - Id of sender
- user_id - Id of user last modifing message
- target - Comma separated numbers taken from group
- data - Message or path to file (multimedia messages)
- ctime - Creation timestamp
- stime - Scheduled timestamp

Table statuses

- id - Id of status
- message_id - Id of message
- mtime - Modification timestamp
- status -

  See Also

  GitGis::Whatsapp::Model::Message
- target - Receiver's number associated with status
- debug - Debug info

Table users

Stores users for authentication

- id - Id of user
- username - Login
- password - Password in MD5
- email - Empty, field required by Slim
- roles - Comma seperated roles (ADMIN)
- credits - Credits left
- ctime - Creation timestamp
- dtime - Delete timestamp

3 Protocol

WhatsApp uses some sort of customized XMPP server, named internally as FunXMPP, which is basically some extended proprietary version.

Authentication

WhatsApp uses JID (jabber id) and password to successfully login to the service. The password is generated by the server and received upon registration. The JID is a concatenation between your country's code and mobile number.

Multimedia Message sending

Photos, Videos and Audio files shared with WhatsApp contacts are HTTP-uploaded to a server before being sent to the recipient(s) along with Base64 thumbnail of media file (if applicable) along with the generated HTTP link as the message body.

Files are stored within /upload directory which should be public accessible

Each file is stored in /upload/{messageId % 100}/{messageId % 100}.{realFileName}

Links

- https://github.com/perezdidac/WhatsAPINet
- FunXMPP Protocol - https://github.com/TheKirk/WhatsAPINet/wiki/FunXMPP
- Main protocol: XMPP (http://xmpp.org/rfcs/rfc6120.html)
- Authentification: SASL, digest auth (http://tools.ietf.org/html/rfc2831)
- Blocking: XMPP privacy lists? (http://xmpp.org/extensions/xep-0016.html)
- Ping (http://xmpp.org/extensions/xep-0199.html)

4 Updating dependencies

Dependencies are managed by:

- Composer for PHP packages
- Git Submodule for WhatsAPI code

To upgrade PHP packages run:

```
./composer.phar update
```

In case of any problems remove all files/dirs from /vendor except WhatsAPI
To upgrade WhatsAPI run:

```
git submodule foreach git pull
```

5 Namespace Index

5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

6 Hierarchical Index

6.1 Class Hierarchy

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

7 Class Index

7.1 Class List

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

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

9 Namespace Documentation

9.1 GitGis Namespace Reference

Namespaces

- Auth
- Whatsapp

Classes

- class Pager

  Pagination class.

9.2 GitGis\Auth Namespace Reference

Classes

- class AuthController

  Authetication controller.
- class GitGisMiddleware

9.3 GitGis\Whatsapp Namespace Reference

Namespaces

- Model

Classes

- class GroupsController

  Groups controller.
- class InboxController

  Message controller.
- class MainController

  Main controller.
- class MessagesController

  Message controller.
- class SendersController

  Senders controller.
- class UsersController

  User management controller.

9.4 GitGis\Whatsapp\Model Namespace Reference

Classes

- class Chat

  Inbox message.
- class ChatDAO

  Inbox management class.
- class DBConnection

Overrided default PDO class with config taken from config.php.

- class Group

  Group entity.
- class GroupDAO

  Group management class.
- class Message

  Message entity.
- class MessageDAO

  Message management class.
- class Sender

  Sender entity.
- class SenderDAO

  Sender management class.
- class User

  User entity used during authentication.
- class UserDAO

  Authentication/user management class.
- class WhatsappDAO

  DAO responsible for communicating WhatsApp server.
- class WhatsProt

  Overrides standard WhatsProt class.

10 Class Documentation

10.1 GitGis\Auth\AuthController Class Reference

Authetication controller.

Static Public Member Functions

- static getLoginPage ()

  Shows login form.
- static postLoginPage ()

  Processes login form.
- static getLogoutPage ()

  Logout user and redirects to Main Page.

10.1.1 Detailed Description

Authetication controller.

Definition at line 9 of file AuthController.php.

10.1.2 Member Function Documentation

10.1.2.1 static GitGis\Auth\AuthController::getLoginPage ( ) [static]

Shows login form.

Definition at line 14 of file AuthController.php.

```
14                                              {
15                  $app = \Slim\Slim::getInstance();
16                  $app->render('auth/form.twig.html');
17          }
```

10.1.2.2 static GitGis\Auth\AuthController::getLogoutPage ( ) [static]

Logout user and redirects to Main Page.

Definition at line 45 of file AuthController.php.

```
45                                              {
46                  $strong = \Strong\Strong::getInstance();
47                  $strong->logout(true);
48                  $app = \Slim\Slim::getInstance();
49                  if (MAINURL != '') {
50                          $app->redirect(MAINURL);
51                  } else {
52                          $app->redirect('/');
53                  }
54          }
```

10.1.2.3 static GitGis\Auth\AuthController::postLoginPage ( ) [static]

Processes login form.

Credantials are taken from: $_POST['username'] $_POST['password']

Definition at line 27 of file AuthController.php.

```
27                                              {
28                  $strong = \Strong\Strong::getInstance();
29                  $isLogged = $strong->login($_POST['username'], $_POST['password'], !empty($_POST['remember'
    ]));
30                  if (!$isLogged) {
31                          self::getLoginPage();
32                  } else {
33                          $app = \Slim\Slim::getInstance();
34                          if (MAINURL != '') {
35                                  $app->redirect(MAINURL);
36                          } else {
37                                  $app->redirect('/');
38                          }
39                  }
40          }
```

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

- src/GitGis/Auth/AuthController.php

10.2 GitGis\Whatsapp\Model\Chat Class Reference

Inbox message.

Public Member Functions

- getId ()
  Id of message.
- setId ($id)
- getData ()
  Text of message or link to audio/photo/video file.
- setData ($data)
- getCtime ()
  Creation timestamp.
- setCtime ($ctime)
- getFrom ()
- setFrom ($from)
- getFromNickname ()
- setFromNickname ($from_nickname)
- getTo ()
- setTo ($to)
- getToNickname ()
- setToNickname ($to_nickname)
- getWhatsappId ()
- setWhatsappId ($whatsapp_id)

10.2.1 Detailed Description

Inbox message.

Definition at line 8 of file Chat.php.

10.2.2 Member Function Documentation

10.2.2.1 GitGis\Whatsapp\Model\Chat::getCtime ( )

Creation timestamp.

Definition at line 91 of file Chat.php.

```
91                                  {
92                 return $this->ctime;
93         }
```

10.2.2.2 GitGis\Whatsapp\Model\Chat::getData ( )

Text of message or link to audio/photo/video file.

Definition at line 80 of file Chat.php.

```
80                                  {
81                 return $this->data;
82         }
```

10.2.2.3 GitGis\Whatsapp\Model\Chat::getFrom ( )

Definition at line 104 of file Chat.php.

```
104                                 {
105                 return $this->from;
106         }
```

10.2.2.4 GitGis\Whatsapp\Model\Chat::getFromNickname ( )

Definition at line 117 of file Chat.php.

```
117                                         {
118                 return $this->from_nickname;
119         }
```

10.2.2.5 GitGis\Whatsapp\Model\Chat::getId ( )

Id of message.

Definition at line 69 of file Chat.php.

```
69                                  {
70                 return $this->id;
71         }
```

10.2.2.6 GitGis\Whatsapp\Model\Chat::getTo ( )

Definition at line 130 of file Chat.php.

```
130                                  {
131                 return $this->to;
132         }
```

10.2.2.7 GitGis\Whatsapp\Model\Chat::getToNickname ( )

Definition at line 143 of file Chat.php.

```
143                                         {
144                 return $this->to_nickname;
145         }
```

10.2.2.8 GitGis\Whatsapp\Model\Chat::getWhatsappId ( )

Definition at line 156 of file Chat.php.

```
156                                         {
157                 return $this->whatsapp_id;
158         }
```

10.2.2.9 GitGis\Whatsapp\Model\Chat::setCtime (*$ctime*)

Definition at line 95 of file Chat.php.

```
95                                          {
96                 $this->ctime = $ctime;
97         }
```

10.2.2.10 GitGis\Whatsapp\Model\Chat::setData (*$data*)

Definition at line 84 of file Chat.php.

```
84                                          {
85                 $this->data = $data;
86         }
```

10.2.2.11 GitGis\Whatsapp\Model\Chat::setFrom (*$from*)

Definition at line 108 of file Chat.php.

```
108                                     {
109                 $this->from = $from;
110         }
```

10.2.2.12 GitGis\Whatsapp\Model\Chat::setFromNickname (*$from_nickname*)

Definition at line 121 of file Chat.php.

```
121                                                     {
122                 $this->from_nickname = $from_nickname;
123         }
```

10.2.2.13 GitGis\Whatsapp\Model\Chat::setId (*$id*)

Definition at line 73 of file Chat.php.

```
73                                      {
74                 $this->id = $id;
75         }
```

10.2.2.14 GitGis\Whatsapp\Model\Chat::setTo (*$to*)

Definition at line 134 of file Chat.php.

```
134                                     {
135                 $this->to = $to;
136         }
```

10.2.2.15 GitGis\Whatsapp\Model\Chat::setToNickname (*$to_nickname*)

Definition at line 147 of file Chat.php.

```
147                                                     {
148                 $this->to_nickname = $to_nickname;
149         }
```

10.2.2.16 GitGis\Whatsapp\Model\Chat::setWhatsappId (*$whatsapp_id*)

Definition at line 160 of file Chat.php.

```
160                                                     {
161                 $this->whatsapp_id = $whatsapp_id;
162         }
```

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

- src/GitGis/Whatsapp/Model/Chat.php

10.3 GitGis\Whatsapp\Model\ChatDAO Class Reference

Inbox management class.

Public Member Functions

- getList ($params=array())

  Get list of messages based on query.
- fetch ($id)

  Fetches specified message.
- save (Chat $item)
- delete ($id)

  Deletes message from DB.

10.3.1 Detailed Description

Inbox management class.

Definition at line 12 of file ChatDAO.php.

10.3.2 Member Function Documentation

10.3.2.1 GitGis\Whatsapp\Model\ChatDAO::delete (*$id*)

Deletes message from DB.

Parameters

number	*$id*	

Definition at line 192 of file ChatDAO.php.

```
192                                     {
193                 $db = DBConnection::getInstance();
194
195                 $query = $db->prepare("DELETE FROM chat WHERE id=:message_id ");
196                 $query->bindParam('message_id', $id);
197                 $query->execute();
198         }
```

10.3.2.2 GitGis\Whatsapp\Model\ChatDAO::fetch (*$id*)

Fetches specified message.

Parameters

number	*$id*	

Returns

>

Definition at line 90 of file ChatDAO.php.

```
90                                      {
91                 $db = DBConnection::getInstance();
92                 $item = null;
93
94                 if (empty($id)) {
95                         $item = new Chat();
96                         return $item;
97                 }
98
99                 $query = $db->prepare("SELECT * FROM chat WHERE id=:id ");
```

```
100                 $query->bindParam('id', $id);
101                 $query->execute();
102                 $row = $query->fetch();
103                 if (!empty($row)) {
104                         $item = new Chat();
105                         $item->setId($row['id']);
106                         $item->setData($row['data']);
107                         $item->setFrom($row['from']);
108                         $item->setFromNickname($row['from_nickname']);
109                         $item->setTo($row['to']);
110                         $item->setToNickname($row['to_nickname']);
111                 }
112
113                 return $item;
114         }
```

10.3.2.3 GitGis\Whatsapp\Model\ChatDAO::getList (*$params* = `array()`)

Get list of messages based on query.

Params could contain:

- start
- limit
- from - fetches only messages from

Parameters

array	*$params*	

Returns

multitype:Ambigous <number, unknown> number multitype: mixed

Definition at line 25 of file ChatDAO.php.

```
25                                                  {
26                 if (empty($params['start'])) $params['start'] = 0;
27                 if (empty($params['limit'])) $params['limit'] = 100;
28
29                 $db = DBConnection::getInstance();
30                 $list = array();
31
32                 $where = " WHERE (1=1) ";
33                 if (!empty($params['from'])) {
34                         $where .= " AND `from` = :from ";
35                 }
36                 if (!empty($params['user_id'])) {
37                         $where .= " AND EXISTS (SELECT * FROM senders WHERE senders.username=chat.to AND
      senders.`user_id` = :user_id) ";
38                 }
39
40                 $sql = "SELECT COUNT(chat.id) as total FROM chat ".$where;
41                 $query = $db->prepare($sql);
42                 if (!empty($params['from'])) {
43                         $query->bindParam('from', $params['from']);
44                 }
45                 if (!empty($params['user_id'])) {
46                         $query->bindParam('user_id', $params['user_id']);
47                 }
48                 $query->execute();
49                 $row = $query->fetch();
50                 $total = $row['total'];
51
52                 $sql = "SELECT chat.*
53                                 FROM chat ".$where." ORDER BY ctime DESC LIMIT :start, :limit ";
54                 $query = $db->prepare($sql);
55                 $query->bindParam('start', $params['start'], \PDO::PARAM_INT);
56                 $query->bindParam('limit', $params['limit'], \PDO::PARAM_INT);
57                 if (!empty($params['from'])) {
```

```
58                      $query->bindParam('from', $params['from']);
59              }
60              if (!empty($params['user_id'])) {
61                      $query->bindParam('user_id', $params['user_id']);
62              }
63              $query->execute();
64              while ($row = $query->fetch()) {
65                      $item = new Chat();
66                      $item->setId($row['id']);
67                      $item->setData($row['data']);
68                      $item->setFrom($row['from']);
69                      $item->setFromNickname($row['from_nickname']);
70                      $item->setTo($row['to']);
71                      $item->setToNickname($row['to_nickname']);
72
73                      $list[] = $item;
74              }
75
76              return array(
77                              'start' => !empty($params['start']) ? $params['start'] : 0,
78                              'limit' => !empty($params['limit']) ? $params['limit'] : 0,
79                              'total' => $total,
80                              'list' => $list
81              );
82      }
```

10.3.2.4 GitGis\Whatsapp\Model\ChatDAO::save (Chat *$item*)

Definition at line 116 of file ChatDAO.php.

```
116                                     {
117              if (empty($item)) {
118                      throw new \Exception('Empty item');
119              }
120
121              $db = DBConnection::getInstance();
122
123              if ($item->getFromNickname() == '') {
124                      $sql = "SELECT nickname FROM number WHERE nickname<>'' AND target=:target ";
125                      $query = $db->prepare($sql);
126                      $query->bindParam('target', $item->getFrom());
127                      $query->execute();
128                      $row = $query->fetch();
129                      if (!empty($row['nickname'])) {
130                              $nickname = $row['nickname'];
131                              $item->setFromNickname($nickname);
132                      }
133              }
134              if ($item->getToNickname() == '') {
135                      $sql = "SELECT nickname FROM senders WHERE nickname<>'' AND username=:username ";
136                      $query = $db->prepare($sql);
137                      $query->bindParam('username', $item->getTo());
138                      $query->execute();
139                      $row = $query->fetch();
140                      if (!empty($row['nickname'])) {
141                              $nickname = $row['nickname'];
142                              $item->setToNickname($nickname);
143                      }
144              }
145
146              if ($item->getId() == 0) {
147                      $query = $db->prepare("INSERT INTO chat
148                                      (`from`, `to`, from_nickname, to_nickname, data, ctime,
    whatsapp_id)
149                                      VALUES
150                                      (:from, :to, :from_nickname, :to_nickname, :data, :ctime,
    :whatsapp_id)");
151                      $query->bindParam('from', $item->getFrom());
152                      $query->bindParam('to', $item->getTo());
153                      $query->bindParam('from_nickname', $item->getFromNickname());
154                      $query->bindParam('to_nickname', $item->getToNickname());
155                      $query->bindParam('data', $item->getData());
156                      $query->bindParam('ctime', $item->getCtime(), \PDO::PARAM_INT);
157                      $query->bindParam('whatsapp_id', $item->getWhatsappId() );
158                      $query->execute();
159                      $item->setId($db->lastInsertId());
160              } else {
161                      $sql = ' UPDATE chat SET ';
162                      $sql.= ' whatsapp_id = :whatsapp_id, ';
```

```
163                         $sql.= ' `from` = :from, ';
164                         $sql.= ' `to` = :to, ';
165                         $sql.= ' from_nickname = :from_nickname, ';
166                         $sql.= ' to_nickname= :to_nickname, ';
167                         $sql.= ' data = :data, ';
168                         $sql.= ' ctime = :ctime ';
169                         $sql.= ' WHERE id = :id ';
170 
171                         $query = $db->prepare($sql);
172                         $query->bindParam('id', $item->getId(), \PDO::PARAM_INT);
173                         $query->bindParam('from', $item->getFrom());
174                         $query->bindParam('to', $item->getTo());
175                         $query->bindParam('from_nickname', $item->getFromNickname());
176                         $query->bindParam('to_nickname', $item->getToNickname());
177                         $query->bindParam('data', $item->getData());
178                         $query->bindParam('ctime', $item->getCtime());
179                         $query->bindParam('whatsapp_id', $item->getWhatsappId() );
180                         $query->execute();
181 
182                 }
183 
184                 return $item;
185         }
```

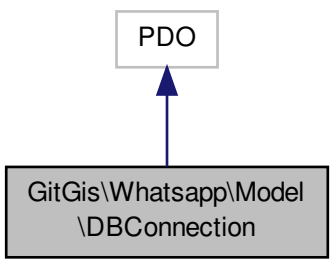
The documentation for this class was generated from the following file:

- src/GitGis/Whatsapp/Model/ChatDAO.php

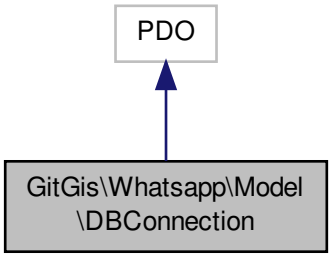
10.4 GitGis\Whatsapp\Model\DBConnection Class Reference

Overrided default PDO class with config taken from config.php.

Inheritance diagram for GitGis\Whatsapp\Model\DBConnection:



Collaboration diagram for GitGis\Whatsapp\Model\DBConnection:



Public Member Functions

- __construct ()
  Creates PDO connection with config taken from config.php.

Static Public Member Functions

- static getInstance ()
  Get instance of DBConnection - static constructor.

10.4.1 Detailed Description

Overrided default PDO class with config taken from config.php.

Definition at line 11 of file DBConnection.php.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 GitGis\Whatsapp\Model\DBConnection::__construct ( )

Creates PDO connection with config taken from config.php.

Definition at line 17 of file DBConnection.php.

```
17                                      {
18                 global $config;
19                 parent::__construct("mysql:host=".$config['db']['host'].";dbname=".$config['db']['name'].""
      , $config['db']['user'], $config['db']['pass']);
20                 $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
21         }
```

10.4.3 Member Function Documentation

10.4.3.1 static GitGis\Whatsapp\Model\DBConnection::getInstance ( ) [static]

Get instance of DBConnection - static constructor.

Returns

Definition at line 28 of file DBConnection.php.

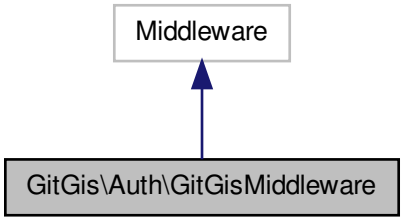
```
28                                      {
29                 return new DBConnection();
30         }
```

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

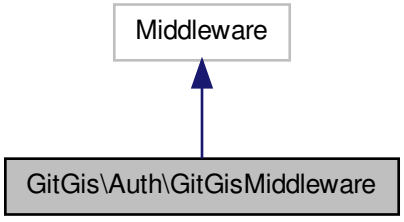
- src/GitGis/Whatsapp/Model/DBConnection.php

10.5 GitGis\Auth\GitGisMiddleware Class Reference

Inheritance diagram for GitGis\Auth\GitGisMiddleware:



Collaboration diagram for GitGis\Auth\GitGisMiddleware:



Public Member Functions

- call ()

10.5.1 Detailed Description

Definition at line 6 of file GitGisMiddleware.php.

10.5.2 Member Function Documentation

10.5.2.1 GitGis\Auth\GitGisMiddleware::call ( )

Definition at line 7 of file GitGisMiddleware.php.

```
7                                   {
8                   $app = $this->app;
9
10                  if ($app->view instanceof \Slim\Views\Twig) {
11                          $strong = \Strong\Strong::getInstance();
12                          $user = $strong->getUser();
13
14                          $userDAO = new \GitGis\Whatsapp\Model\UserDAO();
15                          $user = $userDAO->fetch($user['id']);
16
17                          $app->view->set('user', $user);
18
19                          $twig = $app->view->getInstance();
20                          $twig->addFunction(new \Twig_SimpleFunction('hasRole', function ($role) use ($user)
        {
21                                  return in_array($role, explode(',', $user->getRoles()));
22                          }));
23                  }
24
25                  $this->next->call();
26          }
```

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

- src/GitGis/Auth/GitGisMiddleware.php

10.6 GitGis\Whatsapp\Model\Group Class Reference

Group entity.

Public Member Functions

- getId ()

  Id of group.
- setId ($id)
- getNickname ()

  Human friendly name of group.
- setNickname ($nickname)
- getUserId ()
- setUserId ($user_id)

10.6.1 Detailed Description

Group entity.

Definition at line 7 of file Group.php.

10.6.2 Member Function Documentation

10.6.2.1 GitGis\Whatsapp\Model\Group::getId ( )

Id of group.

Definition at line 33 of file Group.php.

```
33                                  {
34                 return $this->id;
35         }
```

10.6.2.2 GitGis\Whatsapp\Model\Group::getNickname ( )

Human friendly name of group.

Definition at line 44 of file Group.php.

```
44                                          {
45                 return $this->nickname;
46         }
```

10.6.2.3 GitGis\Whatsapp\Model\Group::getUserId ( )

Definition at line 57 of file Group.php.

```
57                                      {
58                 return $this->user_id;
59         }
```

10.6.2.4 GitGis\Whatsapp\Model\Group::setId (*$id*)

Definition at line 37 of file Group.php.

```
37                                      {
38                 $this->id = $id;
39         }
```

10.6.2.5 GitGis\Whatsapp\Model\Group::setNickname (*$nickname*)

Definition at line 48 of file Group.php.

```
48                                                  {
49                 $this->nickname = $nickname;
50         }
```

10.6.2.6 GitGis\Whatsapp\Model\Group::setUserId (*$user_id*)

Definition at line 61 of file Group.php.

```
61                                              {
62                 $this->user_id = $user_id;
63         }
```

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

- src/GitGis/Whatsapp/Model/Group.php

10.7 GitGis\Whatsapp\Model\GroupDAO Class Reference

Group management class.

Public Member Functions

- getList ($params=array())
  Get list of groups based on query.
- fetch ($id)
  Fetches specified group.
- save (Group $item)
  Saves group to DB.
- delete ($id)
  Deletes group.
- addNumbers ($userId, $groupId, $toImport=array())
  Adds numbers to specified group id.
- deleteNumber ($id, $target)
  Removes number from group.
- getNumbers ($id)
  Get numbers for specified group id.
- getDuplicates ($id)
  Checks if in specified group are number duplicated by other groups.
- markSynced ($numbers)

10.7.1 Detailed Description

Group management class.

Definition at line 11 of file GroupDAO.php.

10.7.2 Member Function Documentation

10.7.2.1 GitGis\Whatsapp\Model\GroupDAO::addNumbers (*$userId, $groupId, $toImport* = `array()`)

Adds numbers to specified group id.

`@param number $userId`

Parameters

number	*$groupId*	
array	*$toImport*	

Definition at line 192 of file GroupDAO.php.

```
192                                                                     {
193                 if (!empty($toImport)) {
194                         $db = DBConnection::getInstance();
195                         $query = $db->prepare("REPLACE INTO numbers (group_id, target, nickname) VALUES
        (:group_id, :target, :nickname) ");
196
197                         foreach ($toImport as $target => $nickname) {
198                                 $query->bindParam('group_id', $groupId, \PDO::PARAM_INT);
199                                 $query->bindParam('target', $target);
```

```
200                                 $query->bindParam('nickname', $nickname);
201                                 $query->execute();
202                             }
203                 }
204
205                 $dao = new WhatsappDAO();
206                 $dao->syncContacts($groupId, $userId);
207         }
```

10.7.2.2 GitGis\Whatsapp\Model\GroupDAO::delete (*$id*)

Deletes group.

Parameters

number	*$id*	

Definition at line 167 of file GroupDAO.php.

```
167                                 {
168                 $group = $this->fetch($id);
169
170                 $db = DBConnection::getInstance();
171
172                 $query = $db->prepare("DELETE FROM messages WHERE group_id = :group_id");
173                 $query->bindParam('group_id', $id, \PDO::PARAM_INT);
174                 $query->execute();
175
176                 $query = $db->prepare("DELETE FROM numbers WHERE group_id = :group_id");
177                 $query->bindParam('group_id', $id, \PDO::PARAM_INT);
178                 $query->execute();
179
180                 $query = $db->prepare("DELETE FROM groups WHERE id = :group_id");
181                 $query->bindParam('group_id', $id, \PDO::PARAM_INT);
182                 $query->execute();
183         }
```

10.7.2.3 GitGis\Whatsapp\Model\GroupDAO::deleteNumber (*$id, $target*)

Removes number from group.

Parameters

number	*$id*	
string	*$target*	

Definition at line 215 of file GroupDAO.php.

```
215                                                 {
216                 $db = DBConnection::getInstance();
217                 $query = $db->prepare("DELETE FROM numbers WHERE group_id = :group_id AND target = :target
        ");
218                 $query->bindParam('group_id', $id, \PDO::PARAM_INT);
219                 $query->bindParam('target', $target);
220
221                 $query->execute();
222         }
```

10.7.2.4 GitGis\Whatsapp\Model\GroupDAO::fetch (*$id*)

Fetches specified group.

Parameters

number	*$id*	

Returns

Definition at line 100 of file GroupDAO.php.

```
100                                     {
101                 $db = DBConnection::getInstance();
102         $item = new Group();
103
104                 if (empty($id)) {
105                         return $item;
106                 }
107
108                 $query = $db->prepare("SELECT * FROM groups WHERE id=:id ");
109                 $query->bindParam('id', $id);
110                 $query->execute();
111                 $row = $query->fetch();
112                 if (!empty($row)) {
113                         $item->setId($row['id']);
114                         $item->setNickname($row['nickname']);
115                         $item->setUserId($row['user_id']);
116                 }
117
118                 return $item;
119         }
```

10.7.2.5 GitGis\Whatsapp\Model\GroupDAO::getDuplicates (*$id*)

Checks if in specified group are number duplicated by other groups.

Parameters

number	*$id*	

Returns

multitype:mixed

Definition at line 251 of file GroupDAO.php.

```
251                                             {
252                 $db = DBConnection::getInstance();
253
254                 $numbers = array();
255
256                 $query = $db->prepare("SELECT numbers.target,
257                                         (SELECT COUNT(groups.id) FROM groups JOIN numbers n2 ON
       (n2.group_id=groups.id) WHERE n2.target=numbers.target) AS count
258                                         FROM numbers
259                                         WHERE numbers.group_id=:id
260                                         ORDER BY numbers.target ");
261                 $query->bindParam('id', $id);
262                 $query->execute();
263                 while ($row = $query->fetch()) {
264                         $numbers[$row['target']] = $row['count'];
265                 }
266
267                 return $numbers;
268         }
```

10.7.2.6 GitGis\Whatsapp\Model\GroupDAO::getList (*$params* = `array()`)

Get list of groups based on query.

Params could contain:

- start
- limit
- search - fetches only groups containing specified MSISDN number
- username - fetches only specific group

Parameters

unknown	*$params*	

Returns

multitype:Ambigous <number, unknown> number multitype: mixed

Definition at line 25 of file GroupDAO.php.

```
25                                                  {
26                 if (empty($params['start'])) $params['start'] = 0;
27                 if (empty($params['limit'])) $params['limit'] = 100;
28
29                 $db = DBConnection::getInstance();
30                 $list = array();
31
32         $params['searchNumeric'] = preg_replace('![^0-9]*!', '', $params['search']);
33
34                 $where = " WHERE (1=1) ";
35                 if (!empty($params['searchNumeric'])) {
36                         $where .= " AND (EXISTS (SELECT target FROM numbers WHERE group_id=groups.id
37                             AND target=:searchNumeric)
38                             OR groups.nickname LIKE '%'||:searchNumeric||'%'
39                             ) ";
40                 } else
41         if (!empty($params['search'])) {
42             $where .= " AND groups.nickname LIKE CONCAT('%', :search, '%') ";
43         }
44                 if (!empty($params['user_id'])) {
45                         $where .= ' AND user_id=:user_id ';
46                 }
47
48                 $sql = "SELECT COUNT(id) as total FROM groups ".$where;
49                 $query = $db->prepare($sql);
50         if (!empty($params['search'])) {
51             $query->bindParam('search', $params['search']);
52         }
53         if (!empty($params['searchNumeric'])) {
54             $query->bindParam('searchNumeric', $params['searchNumeric']);
55         }
56                 if (!empty($params['user_id'])) {
57                         $query->bindParam('user_id', $params['user_id']);
58                 }
59                 $query->execute();
60                 $row = $query->fetch();
61                 $total = $row['total'];
62
63                 $sql = "SELECT * FROM groups ".$where." ORDER BY nickname LIMIT :start, :limit ";
64                 $query = $db->prepare($sql);
65                 $query->bindParam('start', $params['start'], \PDO::PARAM_INT);
66                 $query->bindParam('limit', $params['limit'], \PDO::PARAM_INT);
67         if (!empty($params['search'])) {
68             $query->bindParam('search', $params['search']);
69         }
70         if (!empty($params['searchNumeric'])) {
71             $query->bindParam('searchNumeric', $params['searchNumeric']);
72         }
73                 if (!empty($params['user_id'])) {
74                         $query->bindParam('user_id', $params['user_id']);
75                 }
76                 $query->execute();
77                 while ($row = $query->fetch()) {
78                         $item = new Group();
79                         $item->setId($row['id']);
80                         $item->setNickname($row['nickname']);
81                         $item->setUserId($row['user_id']);
```

```
82
83                      $list[$row['id']] = $item;
84              }
85
86              return array(
87                      'start' => !empty($params['start']) ? $params['start'] : 0,
88                      'limit' => !empty($params['limit']) ? $params['limit'] : 0,
89                      'total' => $total,
90                      'list' => $list
91              );
92      }
```

10.7.2.7 GitGis\Whatsapp\Model\GroupDAO::getNumbers (*$id*)

Get numbers for specified group id.

Parameters

number	*$id*	

Returns

multitype:mixed

Definition at line 230 of file GroupDAO.php.

```
230                                     {
231              $db = DBConnection::getInstance();
232
233              $numbers = array();
234
235              $query = $db->prepare("SELECT * FROM numbers WHERE group_id=:id ORDER BY target ");
236              $query->bindParam('id', $id);
237              $query->execute();
238              while ($row = $query->fetch()) {
239                      $numbers[$row['target']] = $row;
240              }
241
242              return $numbers;
243      }
```

10.7.2.8 GitGis\Whatsapp\Model\GroupDAO::markSynced (*$numbers*)

Definition at line 270 of file GroupDAO.php.

```
270                                             {
271              if (empty($numbers)) return;
272              $db = DBConnection::getInstance();
273
274              $ids = '';
275              foreach ($numbers as $number) {
276                      $ids .= "'".$number."',";
277              }
278              $ids = substr($ids, 0, -1);
279
280              $query = $db->prepare("UPDATE numbers SET synced=1 WHERE target IN (".$ids.") ");
281              $query->execute();
282      }
```

10.7.2.9 GitGis\Whatsapp\Model\GroupDAO::save (Group *$item*)

Saves group to DB.

Parameters

Group	*$item*	

Exceptions

\Exception	

Returns

Group

Definition at line 128 of file GroupDAO.php.

```
128                                     {
129                 if (empty($item)) {
130                         throw new \Exception('Empty item');
131                 }
132
133                 $db = DBConnection::getInstance();
134                 if ($item->getId() == 0) {
135                         $query = $db->prepare("INSERT INTO groups
136                                         (nickname, user_id)
137                                         VALUES
138                                         (:nickname, :user_id)");
139                         $query->bindParam('nickname', $item->getNickname());
140                         $query->bindParam('user_id', $item->getUserId());
141                         $query->execute();
142                         $item->setId($db->lastInsertId());
143                 } else {
144                         $sql = ' UPDATE groups SET ';
145                         $sql.= ' nickname = :nickname, ';
146                         $sql.= ' user_id = :user_id ';
147                         $sql.= ' WHERE id = :id ';
148
149                         $query = $db->prepare($sql);
150                         $query->bindParam('id', $item->getId(), \PDO::PARAM_INT);
151                         $query->bindParam('nickname', $item->getNickname());
152                         $query->bindParam('user_id', $item->getUserId());
153                         $query->execute();
154
155                 }
156
157         $this->addNumbers($item->getUserId(), $item->getId());
158
159                 return $item;
160         }
```

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

- src/GitGis/Whatsapp/Model/GroupDAO.php

10.8 GitGis\Whatsapp\GroupsController Class Reference

Groups controller.

Static Public Member Functions

- static getPage ($page=0)

  Get list of groups.
- static getEditPage ($id)

  Get edit group form.
- static postEditPage ($id)

  Process edit group form, validate, save to DB.

- static postUploadNumber ($id)

  Processes file with numbers uploaded for specied group, puts them to DB.
- static getDeleteNumber ($id)

  Deletes number for group, MSISDN is specified in $_GET['number'].
- static deletePage ($id)

  Show delete confirmation.
- static postDeletePage ($id)

  Deletes group.

10.8.1 Detailed Description

Groups controller.

Definition at line 14 of file GroupsController.php.

10.8.2 Member Function Documentation

10.8.2.1 static GitGis\Whatsapp\GroupsController::deletePage (*$id*) [static]

Show delete confirmation.

Definition at line 182 of file GroupsController.php.

```
182                                             {
183                 $app = \Slim\Slim::getInstance();
184                 $userDAO = new UserDAO();
185                 if (!$userDAO->hasRole('ADMIN')) {
186                         return $app->status(403);
187                 }
188
189                 $app->view->set('msg', 'It will delete group, its messages and numbers.');
190                 $app->render('confirm_delete.twig.html');
191         }
```

10.8.2.2 static GitGis\Whatsapp\GroupsController::getDeleteNumber (*$id*) [static]

Deletes number for group, MSISDN is specified in $_GET['number'].

Parameters

number	*$id*	

Definition at line 169 of file GroupsController.php.

```
169                                                 {
170                 $app = \Slim\Slim::getInstance();
171                 $dao = new GroupDAO();
172
173                 $target = $app->request->get('number');
174                 $dao->deleteNumber($id, $target);
175
176                 $app->redirect(MAINURL.'/groups/edit/'.$id);
177         }
```

10.8.2.3 static GitGis\Whatsapp\GroupsController::getEditPage (*$id*) [static]

Get edit group form.

Parameters

number	*$id*	

Definition at line 54 of file GroupsController.php.

```
54                                                  {
55                  $app = \Slim\Slim::getInstance();
56                  $dao = new GroupDAO();
57                  $userDAO = new UserDAO();
58
59                  $app->expires(time());
60
61                  $item = $dao->fetch($id);
62                  if (empty($item)) {
63                          return $app->notFound();
64                  }
65
66                  $app->view->set('menu', 'groups');
67                  $app->view->set('id', $id);
68                  $app->view->set('item', $item);
69                  $app->view->set('users', $userDAO->getList());
70                  $app->view->set('numbers', $dao->getNumbers($item->getId()));
71                  $app->view->set('duplicates', $dao->getDuplicates($item->getId()));
72
73                  $app->render('groups/edit.twig.html');
74          }
```

10.8.2.4 static GitGis\Whatsapp\GroupsController::getPage (*$page* = 0) [static]

Get list of groups.

Parameters

number	*$page*	

Definition at line 21 of file GroupsController.php.

```
21                                                  {
22                  $app = \Slim\Slim::getInstance();
23                  $dao = new GroupDAO();
24                  $userDAO = new UserDAO();
25
26                  $app->expires(time());
27
28                  $query = $_GET;
29                  if (!$userDAO->hasRole('ADMIN')) {
30                          $strong = \Strong\Strong::getInstance();
31                          $user = $strong->getUser();
32                          $query['user_id'] = $user['id'];
33                  }
34                  $pager = new Pager(MAINURL.'/messages/', 25);
35                  $pager->setPage($page);
36                  $query = $pager->getQueryArray($query);
37                  $list = $dao->getList($query);
38                  $pager->setCount(count($list['list']));
39                  if (isset($list['total'])) $pager->setTotal($list['total']);
40
41                  $app->view->set('menu', 'groups');
42                  $app->view->set('query', $query);
43                  $app->view->set('result', $list);
44                  $app->view->set('pager', $pager);
45
46                  $app->render('groups/list.twig.html');
47          }
```

10.8.2.5 static GitGis\Whatsapp\GroupsController::postDeletePage (*$id*) [static]

Deletes group.

Definition at line 196 of file GroupsController.php.

```
196                                                 {
```

```
197                 $app = \Slim\Slim::getInstance();
198                 $userDAO = new UserDAO();
199                 if (!$userDAO->hasRole('ADMIN')) {
200                         return $app->status(403);
201                 }
202
203                 if (!empty($_POST['yes'])) {
204                         $dao = new GroupDAO();
205                         $dao->delete($id);
206                 }
207
208                 return $app->redirect(MAINURL.'/groups');
209         }
```

10.8.2.6 static GitGis\Whatsapp\GroupsController::postEditPage (*$id*) [static]

Process edit group form, validate, save to DB.

Parameters

unknown	*$id*	

Returns

boolean

Definition at line 82 of file GroupsController.php.

```
82                                                  {
83                 $app = \Slim\Slim::getInstance();
84                 $dao = new GroupDAO();
85                 $userDAO = new UserDAO();
86
87                 $item = $dao->fetch($id);
88                 if (empty($item)) {
89                         return $app->notFound();
90                 }
91
92                 $item->setNickname($_POST['nickname']);
93
94                 if ($userDAO->hasRole('ADMIN')) {
95                         $item->setUserId($_POST['user_id']);
96                 } else {
97                         if (empty($id)) {
98                                 $strong = \Strong\Strong::getInstance();
99                                 $user = $strong->getUser();
100                                 $item->setUserId($user['id']);
101                         }
102                 }
103
104                 $validator = new \Valitron\Validator($_POST);
105                 $validator->rule('required', 'nickname');
106
107                 if($validator->validate()) {
108                         $item = $dao->save($item);
109             if (empty($id)) {
110                 $app->flash('info', 'Group '.$item->getNickname().' has been created successfully');
111             }
112             $app->redirect(MAINURL.'/groups/edit/'.$item->getId());
113                 } else {
114                         $app->view->set('menu', 'groups');
115                         $app->view->set('id', $id);
116                         $app->view->set('users', $userDAO->getList());
117                         $app->view->set('item', $item);
118                         $app->view->set('numbers', $dao->getNumbers($item->getId()));
119                         $app->view->set('errors', $validator->errors());
120
121                         $app->render('groups/edit.twig.html');
122                 }
123         }
```

10.8.2.7 static GitGis\Whatsapp\GroupsController::postUploadNumber (*$id*) [static]

Processes file with numbers uploaded for specied group, puts them to DB.

Parameters

number	*$id*	

Definition at line 130 of file GroupsController.php.

```
130                                                 {
131                 $app = \Slim\Slim::getInstance();
132                 $dao = new GroupDAO();
133 
134                 if (!empty($_FILES['file'])) {
135                         $app->response->headers->set('Content-type', 'text/plain');
136 
137                         $numbers = file_get_contents($_FILES['file']['tmp_name']);
138                         $numbers = explode("\n", $numbers);
139                         $toImport = array();
140                         foreach ($numbers as $number) {
141                                 $idx = strpos($number, ',');
142                                 $idx2 = strpos($number, ';');
143                                 if (($idx2 !== false && $idx2 < $idx) || ($idx === false)) {
144                                         $idx = $idx2;
145                                 }
146                                 if ($idx == 0) {
147                                         $idx = strlen($number);
148                                 }
149                                 $msisdn = substr($number, 0, $idx);
150                                 $msisdn = preg_replace('![^0-9]*!', '', $msisdn);
151                                 $nickname = trim(substr($number, $idx+1));
152 
153                                 if (empty($msisdn)) continue;
154                                 echo "$msisdn, $nickname\n";
155 
156                                 $toImport[$msisdn] = $nickname;
157                         }
158 
159             $group = $dao->fetch($id);
160                         $dao->addNumbers($group->getUserId(), $id, $toImport);
161                 }
162         }
```

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

- src/GitGis/Whatsapp/GroupsController.php

10.9 GitGis\Whatsapp\InboxController Class Reference

Message controller.

Static Public Member Functions

- static getPage ($page=0)

  Get list of messages.

10.9.1 Detailed Description

Message controller.

Definition at line 14 of file InboxController.php.

10.9.2 Member Function Documentation

10.9.2.1 static GitGis\Whatsapp\InboxController::getPage (*$page* = 0) [static]

Get list of messages.

Definition at line 19 of file InboxController.php.

```
19                                                  {
20                 $app = \Slim\Slim::getInstance();
21
22                 $app->expires(time());
23
24                 $userDAO = new UserDAO();
25                 $groupDAO = new GroupDAO();
26                 $groupsQuery = array();
27                 if (!$userDAO->hasRole('ADMIN')) {
28                         $strong = \Strong\Strong::getInstance();
29                         $user = $strong->getUser();
30                         $groupsQuery['user_id'] = $user['id'];
31                 }
32                 $groups = $groupDAO->getList($groupsQuery);
33                 if (0 == $groups['total']) {
34                         return $app->redirect(MAINURL.'/groups');
35                 }
36
37                 $dao = new ChatDAO();
38
39                 $query = $_GET;
40                 $query['from'] = preg_replace('![^0-9]*!', '', $query['search']);
41                 if (!$userDAO->hasRole('ADMIN')) {
42                         $strong = \Strong\Strong::getInstance();
43                         $user = $strong->getUser();
44                         $query['user_id'] = $user['id'];
45                 }
46                 $pager = new Pager(MAINURL.'/inbox/', 25);
47                 $pager->setPage($page);
48                 $query = $pager->getQueryArray($query);
49                 $list = $dao->getList($query);
50                 $pager->setCount(count($list['list']));
51                 if (isset($list['total'])) $pager->setTotal($list['total']);
52
53                 $app->view->set('menu', 'inbox');
54                 $app->view->set('query', $query);
55                 $app->view->set('result', $list);
56                 $app->view->set('pager', $pager);
57
58                 $app->render('inbox/list.twig.html');
59         }
```

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

- src/GitGis/Whatsapp/InboxController.php

10.10 GitGis\Whatsapp\MainController Class Reference

Main controller.

Static Public Member Functions

- static getPage ()

  Redirects to /messages link.

10.10.1 Detailed Description

Main controller.

Definition at line 8 of file MainController.php.

10.10.2 Member Function Documentation

10.10.2.1 static GitGis\Whatsapp\MainController::getPage ( ) [static]

Redirects to /messages link.

Definition at line 13 of file MainController.php.

```
13                                      {
14                 $app = \Slim\Slim::getInstance();
15                 $app->redirect(MAINURL.'/messages');
16         }
```

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

- src/GitGis/Whatsapp/MainController.php

10.11 GitGis\Whatsapp\Model\Message Class Reference

Message entity.

Public Member Functions

- getId ()
  Id of message.
- setId ($id)
- getGroupId ()
  Id of group.
- setGroupId ($group_id)
- getSenderId ()
  Id of sender.
- setSenderId ($sender_id)
- getUserId ()
  Id of user who last modifed the message.
- setUserId ($user_id)
- getKind ()
  Kind of message.
- setKind ($kind)
- getTarget ()
  Target is a comma separated list of MSISDNs of receivers.
- setTarget ($target)
- getData ()
  Text of message or link to audio/photo/video file.
- setData ($data)
- getCtime ()
  Creation timestamp.
- setCtime ($ctime)
- getStime ()
  Scheduled timestamp.
- setStime ($stime)

Public Attributes

- const KIND_TEXT_MSG = 1
- const KIND_PHOTO_MSG = 2
- const KIND_AUDIO_MSG = 4
- const KIND_VIDEO_MSG = 8
- const MESSAGE_STATUS_TO_SEND = 1
- const MESSAGE_STATUS_TO_RETRY = 2
- const MESSAGE_STATUS_SENT = 4
- const MESSAGE_STATUS_RECEIVED_BY_SERVER = 5
- const MESSAGE_STATUS_ERROR = 8
- const MESSAGE_STATUS_RETRIED = 16

10.11.1 Detailed Description

Message entity.

Definition at line 9 of file Message.php.

10.11.2 Member Function Documentation

10.11.2.1 GitGis\Whatsapp\Model\Message::getCtime ( )

Creation timestamp.

Definition at line 225 of file Message.php.

```
225                                     {
226                 return $this->ctime;
227         }
```

10.11.2.2 GitGis\Whatsapp\Model\Message::getData ( )

Text of message or link to audio/photo/video file.

Definition at line 214 of file Message.php.

```
214                                     {
215                 return $this->data;
216         }
```

10.11.2.3 GitGis\Whatsapp\Model\Message::getGroupId ( )

Id of group.

Definition at line 157 of file Message.php.

```
157                                     {
158         return $this->group_id;
159     }
```

10.11.2.4 GitGis\Whatsapp\Model\Message::getId ( )

Id of message.

Definition at line 146 of file Message.php.

```
146                                     {
147                 return $this->id;
148         }
```

10.11.2.5 GitGis\Whatsapp\Model\Message::getKind ( )

Kind of message.

See KIND_* consts

Definition at line 192 of file Message.php.

```
192                                 {
193                 return $this->kind;
194         }
```

10.11.2.6 GitGis\Whatsapp\Model\Message::getSenderId ( )

Id of sender.

Definition at line 168 of file Message.php.

```
168                                 {
169         return $this->sender_id;
170     }
```

10.11.2.7 GitGis\Whatsapp\Model\Message::getStime ( )

Scheduled timestamp.

Definition at line 236 of file Message.php.

```
236                                   {
237                 return $this->stime;
238         }
```

10.11.2.8 GitGis\Whatsapp\Model\Message::getTarget ( )

Target is a comma separated list of MSISDNs of receivers.

Definition at line 203 of file Message.php.

```
203                                    {
204                 return $this->target;
205         }
```

10.11.2.9 GitGis\Whatsapp\Model\Message::getUserId ( )

Id of user who last modifed the message.

Definition at line 179 of file Message.php.

```
179                                    {
180                 return $this->user_id;
181         }
```

10.11.2.10 GitGis\Whatsapp\Model\Message::setCtime (*$ctime*)

Definition at line 229 of file Message.php.

```
229                                                {
230                 $this->ctime = $ctime;
231         }
```

10.11.2.11 GitGis\Whatsapp\Model\Message::setData (*$data*)

Definition at line 218 of file Message.php.

```
218                                     {
219                 $this->data = $data;
220         }
```

10.11.2.12 GitGis\Whatsapp\Model\Message::setGroupId (*$group_id*)

Definition at line 161 of file Message.php.

```
161                                         {
162         $this->group_id = $group_id;
163     }
```

10.11.2.13 GitGis\Whatsapp\Model\Message::setId (*$id*)

Definition at line 150 of file Message.php.

```
150                                     {
151                 $this->id = $id;
152         }
```

10.11.2.14 GitGis\Whatsapp\Model\Message::setKind (*$kind*)

Definition at line 196 of file Message.php.

```
196                                     {
197                 $this->kind = $kind;
198         }
```

10.11.2.15 GitGis\Whatsapp\Model\Message::setSenderId (*$sender_id*)

Definition at line 172 of file Message.php.

```
172                                         {
173         $this->sender_id = $sender_id;
174     }
```

10.11.2.16 GitGis\Whatsapp\Model\Message::setStime (*$stime*)

Definition at line 240 of file Message.php.

```
240                                       {
241                 $this->stime = $stime;
242         }
```

10.11.2.17 GitGis\Whatsapp\Model\Message::setTarget (*$target*)

Definition at line 207 of file Message.php.

```
207                                       {
208                 $this->target = $target;
209         }
```

10.11.2.18 GitGis\Whatsapp\Model\Message::setUserId (*$user_id*)

Definition at line 183 of file Message.php.

```
183                                         {
184                 $this->user_id = $user_id;
185         }
```

10.11.3 Member Data Documentation

10.11.3.1 const GitGis\Whatsapp\Model\Message::KIND_AUDIO_MSG = 4

Definition at line 28 of file Message.php.

10.11.3.2 const GitGis\Whatsapp\Model\Message::KIND_PHOTO_MSG = 2

Definition at line 22 of file Message.php.

10.11.3.3 const GitGis\Whatsapp\Model\Message::KIND_TEXT_MSG = 1

Definition at line 16 of file Message.php.

10.11.3.4 const GitGis\Whatsapp\Model\Message::KIND_VIDEO_MSG = 8

Definition at line 34 of file Message.php.

10.11.3.5 const GitGis\Whatsapp\Model\Message::MESSAGE_STATUS_ERROR = 8

Definition at line 69 of file Message.php.

10.11.3.6 const GitGis\Whatsapp\Model\Message::MESSAGE_STATUS_RECEIVED_BY_SERVER = 5

Definition at line 62 of file Message.php.

10.11.3.7 const GitGis\Whatsapp\Model\Message::MESSAGE_STATUS_RETRIED = 16

Definition at line 76 of file Message.php.

10.11.3.8 const GitGis\Whatsapp\Model\Message::MESSAGE_STATUS_SENT = 4

Definition at line 55 of file Message.php.

10.11.3.9 const GitGis\Whatsapp\Model\Message::MESSAGE_STATUS_TO_RETRY = 2

Definition at line 48 of file Message.php.

10.11.3.10 const GitGis\Whatsapp\Model\Message::MESSAGE_STATUS_TO_SEND = 1

Definition at line 41 of file Message.php.

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

- src/GitGis/Whatsapp/Model/Message.php

10.12 GitGis\Whatsapp\Model\MessageDAO Class Reference

Message management class.

Public Member Functions

- getList ($params=array())

  Get list of messages based on query.
- fetch ($id)

  Fetches specified message.
- save (Message $item)

  Saves message to DB.
- delete ($id)

  Deletes message from DB.
- getStatusesByWhatsappId ($whatsapp_id)

  Gets statuses for whatsapp_id (id of sent message returned by whatsapp)
- getStatuses (Message $item)

  Gets statuses for message.
- addStatus (Message $item, $status, $debugMsg= ”, $target= ”, $whatsapp_id= ”)

  Add status to message.
- clearRetry (Message $item)

  Changes force resend status to resent.

10.12.1 Detailed Description

Message management class.

Definition at line 13 of file MessageDAO.php.

10.12.2 Member Function Documentation

10.12.2.1 GitGis\Whatsapp\Model\MessageDAO::addStatus (Message *$item*, *$status*, *$debugMsg = ”*, *$target = ”*, *$whatsapp_id = ”*)

Add status to message.

Parameters

Message	*$item*	
number	*$status*	
string	*$debug*	
string	*$target*	
string	*$whatsapp_id*	

Definition at line 279 of file MessageDAO.php.

```
279
    {
280             $db = DBConnection::getInstance();
281
282             $query = $db->prepare("INSERT INTO statuses
283                                    (message_id, mtime, status, debug, target, whatsapp_id)
284                                    VALUES
285                                    (:message_id, :mtime, :status, :debug, :target, :whatsapp_id)");
286             $query->bindParam('message_id', $item->getId());
287             $query->bindParam('mtime', time());
288             $query->bindParam('status', $status);
289             $query->bindParam('debug', $debugMsg);
290             $query->bindParam('target', $target);
291             $query->bindParam('whatsapp_id', $whatsapp_id);
292             $query->execute();
293         }
```

10.12.2.2 GitGis\Whatsapp\Model\MessageDAO::clearRetry (Message *$item*)

Changes force resend status to resent.

Parameters

Message	*$item*	

Definition at line 300 of file MessageDAO.php.

```
300                                             {
301         $db = DBConnection::getInstance();
302
303         $query = $db->prepare("UPDATE statuses
304                                     SET status = 16
305                                     WHERE status = 2 AND message_id = :message_id ");
306         $query->bindParam('message_id', $item->getId());
307         $query->execute();
308     }
```

10.12.2.3 GitGis\Whatsapp\Model\MessageDAO::delete (*$id*)

Deletes message from DB.

Parameters

number	*$id*	

Definition at line 212 of file MessageDAO.php.

```
212                                 {
213                 $db = DBConnection::getInstance();
214
215                 $item = $this->fetch($id);
216                 $statuses = $this->getStatuses($item);
217                 if (!empty($statuses)) {
218                         return;
219                 }
220
221                 $query = $db->prepare("DELETE FROM statuses WHERE message_id=:message_id ");
222                 $query->bindParam('message_id', $id);
223                 $query->execute();
224
225                 $query = $db->prepare("DELETE FROM messages WHERE id=:message_id ");
226                 $query->bindParam('message_id', $id);
227                 $query->execute();
228         }
```

10.12.2.4 GitGis\Whatsapp\Model\MessageDAO::fetch (*$id*)

Fetches specified message.

Parameters

number	*$id*	

Returns

>

Definition at line 117 of file MessageDAO.php.

```
117                                 {
118                 $db = DBConnection::getInstance();
119                 $item = null;
120
121                 if (empty($id)) {
122                         $item = new Message();
123                         return $item;
124                 }
125
126                 $query = $db->prepare("SELECT * FROM messages WHERE id=:id ");
```

```
127                 $query->bindParam('id', $id);
128                 $query->execute();
129                 $row = $query->fetch();
130                 if (!empty($row)) {
131                         $item = new Message();
132                         $item->setId($row['id']);
133                         $item->setGroupId($row['group_id']);
134             $item->setSenderId($row['sender_id']);
135                         $item->setUserId($row['user_id']);
136                         $item->setkind($row['kind']);
137                         $item->setTarget($row['target']);
138                         $item->setData($row['data']);
139                         $item->setCtime($row['ctime']);
140                         $item->setStime($row['stime']);
141                 }
142
143                 return $item;
144         }
```

10.12.2.5 GitGis\Whatsapp\Model\MessageDAO::getList (*$params* = `array()`)

Get list of messages based on query.

Params could contain:

- start
- limit
- toSend - gets only messages scheduled after specified timestamp
- username - fetches only specific message

Parameters

array	*$params*	

Returns

multitype:Ambigous <number, unknown> number multitype: mixed

Definition at line 27 of file MessageDAO.php.

```
27                                                  {
28                 if (empty($params['start'])) $params['start'] = 0;
29                 if (empty($params['limit'])) $params['limit'] = 100;
30
31                 $db = DBConnection::getInstance();
32                 $list = array();
33
34                 $where = " WHERE (1=1) ";
35                 if (!empty($params['toSend'])) {
36             $ids = array(0 => 0);
37
38                         $sql = " SELECT statuses.message_id FROM statuses
39                                 JOIN messages ON (statuses.message_id = messages.id)
40                                 WHERE messages.stime < ".(intval($params['toSend'])).""
41                                 GROUP BY statuses.message_id
42                                 HAVING MAX(statuses.status) = 1 ";
43                         $query = $db->prepare($sql);
44                         $query->execute();
45                         while ($row = $query->fetch()) {
46                                 $ids[$row['message_id']] = $row['message_id'];
47                         }
48
49             $sql = " SELECT statuses.message_id FROM statuses
50                                 JOIN messages ON (statuses.message_id = messages.id)
51                                 WHERE
52                                     messages.stime < ".(intval($params['toSend'])).""
53                                     AND statuses.target = ''
54                                     AND statuses.status = 2
```

```
55                              GROUP BY statuses.message_id ";
56              $query = $db->prepare($sql);
57              $query->execute();
58              while ($row = $query->fetch()) {
59                  $ids[$row['message_id']] = $row['message_id'];
60              }
61
62              $where .= " AND id IN (".implode(',', array_values($ids)).") ";
63                  }
64                  if (!empty($params['user_id'])) {
65                          $where .= " AND EXISTS (SELECT * FROM groups WHERE messages.group_id=groups.id AND
    groups.user_id=:user_id) ";
66                  }
67
68                  $sql = "SELECT COUNT(messages.id) as total FROM messages ".$where;
69                  $query = $db->prepare($sql);
70                  if (!empty($params['user_id'])) {
71                          $query->bindParam('user_id', $params['user_id']);
72                  }
73                  $query->execute();
74                  $row = $query->fetch();
75                  $total = $row['total'];
76
77                  $sql = "SELECT messages.*,
78                                  (SELECT max(statuses.status) FROM statuses WHERE statuses.message_id =
    messages.id ) as max_status
79                                  FROM messages ".$where." ORDER BY ctime DESC LIMIT :start, :limit ";
80                  $query = $db->prepare($sql);
81                  $query->bindParam('start', $params['start'], \PDO::PARAM_INT);
82                  $query->bindParam('limit', $params['limit'], \PDO::PARAM_INT);
83                  if (!empty($params['user_id'])) {
84                          $query->bindParam('user_id', $params['user_id']);
85                  }
86                  $query->execute();
87                  while ($row = $query->fetch()) {
88                          $item = new Message();
89                          $item->setId($row['id']);
90              $item->setGroupId($row['group_id']);
91              $item->setSenderId($row['sender_id']);
92                          $item->setUserId($row['user_id']);
93                          $item->setkind($row['kind']);
94                          $item->setTarget($row['target']);
95                          $item->setData($row['data']);
96                          $item->setCtime($row['ctime']);
97                          $item->setStime($row['stime']);
98                          $item->max_status = $row['max_status'];
99
100                          $list[] = $item;
101                 }
102
103                 return array(
104                         'start' => !empty($params['start']) ? $params['start'] : 0,
105                         'limit' => !empty($params['limit']) ? $params['limit'] : 0,
106                         'total' => $total,
107                         'list' => $list
108                 );
109         }
```

10.12.2.6 GitGis\Whatsapp\Model\MessageDAO::getStatuses (Message *$item*)

Gets statuses for message.

Parameters

Message	*$item*	

Returns

multitype:mixed

Definition at line 256 of file MessageDAO.php.

```
256                                                 {
257                 $retVal = array();
258                 $db = DBConnection::getInstance();
```

```
259
260                 $query = $db->prepare("SELECT * FROM statuses WHERE message_id=:message_id ORDER BY id DESC
    ");
261                 $query->bindParam('message_id', $item->getId());
262                 $query->execute();
263                 while ($row = $query->fetch()) {
264                         $retVal[$row['id']] = $row;
265                 }
266
267                 return $retVal;
268         }
```

10.12.2.7 GitGis\Whatsapp\Model\MessageDAO::getStatusesByWhatsappId (*$whatsapp_id*)

Gets statuses for whatsapp_id (id of sent message returned by whatsapp)

Parameters

unknown	*$whatsapp_id*	

Returns

multitype:mixed

Definition at line 236 of file MessageDAO.php.

```
236                                                         {
237                 $retVal = array();
238                 $db = DBConnection::getInstance();
239
240                 $query = $db->prepare("SELECT * FROM statuses WHERE whatsapp_id=:whatsapp_id ORDER BY id
    DESC ");
241                 $query->bindParam('whatsapp_id', $whatsapp_id);
242                 $query->execute();
243                 while ($row = $query->fetch()) {
244                         $retVal[$row['id']] = $row;
245                 }
246
247                 return $retVal;
248         }
```

10.12.2.8 GitGis\Whatsapp\Model\MessageDAO::save (Message *$item*)

Saves message to DB.

Parameters

Message	*$item*	

Exceptions

\Exception	

Returns

Message

Definition at line 153 of file MessageDAO.php.

```
153                                         {
154                 if (empty($item)) {
155                         throw new \Exception('Empty item');
156                 }
157                 $groupDao = new GroupDAO();
158                 $item->setTarget(implode(',', array_keys($groupDao->getNumbers($item->getGroupId()))));
159
```

```
160 //              if (!$item->getTarget()) $item->setTarget('');
161                 if (!$item->getData()) $item->setData('');
162                 if (!$item->getStime()) $item->setStime(0);
163 
164                 $db = DBConnection::getInstance();
165                 if ($item->getId() == 0) {
166                         $query = $db->prepare("INSERT INTO messages
167                                         (group_id, sender_id, user_id, kind, target, data, ctime, stime)
168                                         VALUES
169                                         (:group_id, :sender_id, :user_id, :kind, :target, :data, :ctime,
    :stime)");
170             $query->bindParam('group_id', $item->getGroupId());
171             $query->bindParam('sender_id', $item->getSenderId());
172                         $query->bindParam('user_id', $item->getUserId());
173                         $query->bindParam('kind', $item->getKind());
174                         $query->bindParam('target', $item->getTarget());
175                         $query->bindParam('data', $item->getData());
176                         $query->bindParam('ctime', time() );
177                         $query->bindParam('stime', $item->getStime());
178                         $query->execute();
179                         $item->setId($db->lastInsertId());
180                 } else {
181                         $sql = ' UPDATE messages SET ';
182             $sql.= ' group_id = :group_id, ';
183             $sql.= ' sender_id = :sender_id, ';
184                         $sql.= ' user_id = :user_id, ';
185                         $sql.= ' kind = :kind, ';
186                         $sql.= ' target = :target, ';
187                         $sql.= ' data = :data, ';
188                         $sql.= ' stime = :stime ';
189                         $sql.= ' WHERE id = :id ';
190 
191                         $query = $db->prepare($sql);
192                         $query->bindParam('id', $item->getId(), \PDO::PARAM_INT);
193                         $query->bindParam('group_id', $item->getGroupId(), \PDO::PARAM_INT);
194             $query->bindParam('sender_id', $item->getSenderId(), \PDO::PARAM_INT);
195                         $query->bindParam('user_id', $item->getUserId(), \PDO::PARAM_INT);
196                         $query->bindParam('kind', $item->getKind());
197                         $query->bindParam('target', $item->getTarget());
198                         $query->bindParam('data', $item->getData());
199                         $query->bindParam('stime', $item->getStime());
200                         $query->execute();
201 
202                 }
203 
204                 return $item;
205         }
```

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

- src/GitGis/Whatsapp/Model/MessageDAO.php

10.13 GitGis\Whatsapp\MessagesController Class Reference

Message controller.

Static Public Member Functions

- static getPage ($page=0)

  Get list of messages.
- static getHumanUrl (Message $item)

  Get abbreviated link to file connected to multimedia message.
- static deletePage ($id)

  Deletes message.
- static getEditPage ($id)

  Get message edit form.
- static postUploadPage ($id)

Process upload multimedia file.

- static postEditPage ($id)

  Process message form.

- static getSendText ()

  Creates text message and redirects to edit.

- static getSendPhoto ()

  Creates photo message and redirects to edit.

- static getSendAudio ()

  Creates audio message and redirects to edit.

- static getSendVideo ()

  Creates video message and redirects to edit.

10.13.1 Detailed Description

Message controller.

Definition at line 15 of file MessagesController.php.

10.13.2 Member Function Documentation

10.13.2.1 static GitGis\Whatsapp\MessagesController::deletePage (*$id*) [static]

Deletes message.

Definition at line 92 of file MessagesController.php.

```
92                                                  {
93                 $app = \Slim\Slim::getInstance();
94                 $dao = new MessageDAO();
95                 $dao->delete($id);
96
97                 return $app->redirect(MAINURL.'/messages');
98         }
```

10.13.2.2 static GitGis\Whatsapp\MessagesController::getEditPage (*$id*) [static]

Get message edit form.

Definition at line 103 of file MessagesController.php.

```
103                                                  {
104                 $app = \Slim\Slim::getInstance();
105                 $dao = new MessageDAO();
106                 $userDAO = new UserDAO();
107         $groupDAO = new GroupDAO();
108         $groupsQuery = array();
109         $senderDAO = new SenderDAO();
110         $sendersQuery = array();
111         if (!$userDAO->hasRole('ADMIN')) {
112             $strong = \Strong\Strong::getInstance();
113             $user = $strong->getUser();
114             $groupsQuery['user_id'] = $user['id'];
115             $sendersQuery['user_id'] = $user['id'];
116         }
117
118         $groups = $groupDAO->getList($groupsQuery);
119         if (0 == $groups['total']) {
120             return $app->redirect(MAINURL.'/groups');
121         }
122         $senders = $senderDAO->getList($sendersQuery);
123         if (0 == $senders['total']) {
124             return $app->redirect(MAINURL.'/senders');
125         }
```

```
126
127                 if (!$userDAO->hasRole('ADMIN')) {
128 //                      return $app->status(403);
129                 }
130
131                 $app->expires(time());
132
133                 $item = $dao->fetch($id);
134                 if (empty($item)) {
135                     return $app->notFound();
136                 } else {
137                     if (!$userDAO->hasRole('ADMIN') && $item->getGroupId()>0 && !in_array($item->
    getGroupId(), array_keys($groups['list']))) {
138                         return $app->status(403);
139                     }
140                 }
141
142                 $statuses = $dao->getStatuses($item);
143         $hasErrors = false;
144         $targetSent = array();
145         $targetReceived = array();
146         if (!empty($statuses)) {
147             foreach ($statuses as $status) {
148
149                 if (!empty($status['target'])) {
150                     if ($status['status'] ==
    Message::MESSAGE_STATUS_RECEIVED_BY_SERVER) {
151                         $targetReceived[$status['target']] = $status['target'];
152                     }
153                     if ($status['status'] == Message::MESSAGE_STATUS_SENT) {
154                         $targetSent[$status['target']] = $status['target'];
155                     }
156                 }
157                 if ($status['status'] == Message::MESSAGE_STATUS_ERROR) {
158                     $hasErrors = true;
159                 }
160             }
161         }
162
163         if ($item->getStime() == 0) {
164             $item->setStime(time());
165         }
166
167         $item->dataHuman = self::getHumanUrl($item);
168                 $mime = '*/*';
169                 if (Message::KIND_PHOTO_MSG == $item->getKind()) {
170             $mime = 'image/*';
171         }
172                 if (Message::KIND_AUDIO_MSG == $item->getKind()) {
173             $mime = 'audio/*';
174         }
175                 if (Message::KIND_VIDEO_MSG == $item->getKind()) {
176             $mime = 'video/*';
177         }
178
179                 $app->view->set('KIND_TEXT_MSG', Message::KIND_TEXT_MSG);
180                 $app->view->set('KIND_PHOTO_MSG', Message::KIND_PHOTO_MSG);
181                 $app->view->set('KIND_AUDIO_MSG', Message::KIND_AUDIO_MSG);
182                 $app->view->set('KIND_VIDEO_MSG', Message::KIND_VIDEO_MSG);
183
184                 $app->view->set('menu', 'messages');
185                 $app->view->set('id', $id);
186                 $app->view->set('item', $item);
187                 $app->view->set('groups', $groups);
188         $app->view->set('senders', $senders);
189                 $app->view->set('statuses', $statuses);
190         $app->view->set('hasErrors', $hasErrors);
191         $app->view->set('noSent', count($targetSent));
192         $app->view->set('noReceived', count($targetReceived));
193                 $app->view->set('mime', $mime);
194
195                 $app->render('messages/edit.twig.html');
196         }
```

10.13.2.3 static GitGis\Whatsapp\MessagesController::getHumanUrl (Message *$item*) [static]

Get abbreviated link to file connected to multimedia message.

Parameters

Message	*$item*	

Returns

string

Definition at line 80 of file MessagesController.php.

```
80                                                      {
81                 $data = $item->getData();
82                 $prefix = ($item->getId() % 100).'/'.$item->getId().'.';
83                 if (substr($data, 0, strlen($prefix)) == $prefix) {
84                         $data = substr($data, strlen($prefix));
85                 }
86                 return $data;
87         }
```

10.13.2.4 static GitGis\Whatsapp\MessagesController::getPage (*$page* = 0) [static]

Get list of messages.

Definition at line 20 of file MessagesController.php.

```
20                                                  {
21                 $app = \Slim\Slim::getInstance();
22                 $dao = new MessageDAO();
23                 $userDAO = new UserDAO();
24                 $users = $userDAO->getList();
25                 $groupDAO = new GroupDAO();
26                 $groupsQuery = array();
27         $senderDAO = new SenderDAO();
28         $sendersQuery = array();
29                 if (!$userDAO->hasRole('ADMIN')) {
30                         $strong = \Strong\Strong::getInstance();
31                         $user = $strong->getUser();
32                         $groupsQuery['user_id'] = $user['id'];
33             $sendersQuery['user_id'] = $user['id'];
34         }
35
36         $groups = $groupDAO->getList($groupsQuery);
37         if (0 == $groups['total']) {
38             return $app->redirect(MAINURL.'/groups');
39         }
40         $senders = $senderDAO->getList($sendersQuery);
41         if (0 == $senders['total']) {
42             return $app->redirect(MAINURL.'/senders');
43         }
44
45                 $app->expires(time());
46
47                 $query = $_GET;
48                 if (!$userDAO->hasRole('ADMIN')) {
49                         $strong = \Strong\Strong::getInstance();
50                         $user = $strong->getUser();
51                         $query['user_id'] = $user['id'];
52                 }
53                 $pager = new Pager(MAINURL.'/messages/', 25);
54                 $pager->setPage($page);
55                 $query = $pager->getQueryArray($query);
56                 $list = $dao->getList($query);
57                 $pager->setCount(count($list['list']));
58                 if (isset($list['total'])) $pager->setTotal($list['total']);
59
60                 foreach ($list['list'] as $k => $v) {
61                         $list['list'][$k]->dataHuman = self::getHumanUrl($v);
62                 }
63
64                 $app->view->set('menu', 'messages');
65                 $app->view->set('result', $list);
66                 $app->view->set('pager', $pager);
67         $app->view->set('groups', $groups);
68         $app->view->set('senders', $senders);
```

```
69                 $app->view->set('users', $users);
70
71                 $app->render('messages/list.twig.html');
72         }
```

10.13.2.5 static GitGis\Whatsapp\MessagesController::getSendAudio ( ) [static]

Creates audio message and redirects to edit.

Definition at line 397 of file MessagesController.php.

```
397                                                 {
398                 $app = \Slim\Slim::getInstance();
399                 $dao = new MessageDAO();
400
401                 $item = new Message();
402
403                 $item->setKind(Message::KIND_AUDIO_MSG);
404                 $item->setCtime(time());
405                 $strong = \Strong\Strong::getInstance();
406                 $user = $strong->getUser();
407                 $item->setUserId($user['id']);
408
409                 $item = $dao->save($item);
410
411                 $app->redirect(MAINURL.'/messages/edit/'.$item->getId());
412         }
```

10.13.2.6 static GitGis\Whatsapp\MessagesController::getSendPhoto ( ) [static]

Creates photo message and redirects to edit.

Definition at line 377 of file MessagesController.php.

```
377                                                 {
378                 $app = \Slim\Slim::getInstance();
379                 $dao = new MessageDAO();
380
381                 $item = new Message();
382
383                 $item->setKind(Message::KIND_PHOTO_MSG);
384                 $item->setCtime(time());
385                 $strong = \Strong\Strong::getInstance();
386                 $user = $strong->getUser();
387                 $item->setUserId($user['id']);
388
389                 $item = $dao->save($item);
390
391                 $app->redirect(MAINURL.'/messages/edit/'.$item->getId());
392         }
```

10.13.2.7 static GitGis\Whatsapp\MessagesController::getSendText ( ) [static]

Creates text message and redirects to edit.

Definition at line 357 of file MessagesController.php.

```
357                                                 {
358                 $app = \Slim\Slim::getInstance();
359                 $dao = new MessageDAO();
360
361                 $item = new Message();
362
363                 $item->setKind(Message::KIND_TEXT_MSG);
364                 $item->setCtime(time());
365                 $strong = \Strong\Strong::getInstance();
366                 $user = $strong->getUser();
367                 $item->setUserId($user['id']);
368
369                 $item = $dao->save($item);
370
371                 $app->redirect(MAINURL.'/messages/edit/'.$item->getId());
372         }
```

10.13.2.8 static GitGis\Whatsapp\MessagesController::getSendVideo ( ) [static]

Creates video message and redirects to edit.

Definition at line 417 of file MessagesController.php.

```
417                                             {
418                 $app = \Slim\Slim::getInstance();
419                 $dao = new MessageDAO();
420
421                 $item = new Message();
422
423                 $item->setKind(Message::KIND_VIDEO_MSG);
424                 $item->setCtime(time());
425                 $strong = \Strong\Strong::getInstance();
426                 $user = $strong->getUser();
427                 $item->setUserId($user['id']);
428
429                 $item = $dao->save($item);
430
431                 $app->redirect(MAINURL.'/messages/edit/'.$item->getId());
432         }
```

10.13.2.9 static GitGis\Whatsapp\MessagesController::postEditPage (*$id*) [static]

Process message form.

Definition at line 227 of file MessagesController.php.

```
227                                                 {
228                 $app = \Slim\Slim::getInstance();
229                 $dao = new MessageDAO();
230                 $userDAO = new UserDAO();
231
232                 $strong = \Strong\Strong::getInstance();
233                 $user = $strong->getUser();
234                 $user = $userDAO->fetch($user['id']);
235
236         $groupDAO = new GroupDAO();
237         $groupsQuery = array();
238         $senderDAO = new SenderDAO();
239         $sendersQuery = array();
240         if (!$userDAO->hasRole('ADMIN')) {
241             $strong = \Strong\Strong::getInstance();
242             $user2 = $strong->getUser();
243             $groupsQuery['user_id'] = $user2['id'];
244             $sendersQuery['user_id'] = $user2['id'];
245         }
246
247         $groups = $groupDAO->getList($groupsQuery);
248         if (0 == $groups['total']) {
249             return $app->redirect(MAINURL.'/groups');
250         }
251         $senders = $senderDAO->getList($sendersQuery);
252         if (0 == $senders['total']) {
253             return $app->redirect(MAINURL.'/senders');
254         }
255
256                 $app->view->set('KIND_TEXT_MSG', Message::KIND_TEXT_MSG);
257                 $app->view->set('KIND_PHOTO_MSG', Message::KIND_PHOTO_MSG);
258                 $app->view->set('KIND_AUDIO_MSG', Message::KIND_AUDIO_MSG);
259                 $app->view->set('KIND_VIDEO_MSG', Message::KIND_VIDEO_MSG);
260
261                 $item = $dao->fetch($id);
262                 if (empty($item)) {
263                         return $app->notFound();
264                 } else {
265                         if (!$userDAO->hasRole('ADMIN') && $item->getGroupId()>0 && !in_array($item->
    getGroupId(), array_keys($groups['list']))) {
266                                 return $app->status(403);
267                         }
268                         if (empty($_POST['data'])) $_POST['data'] = $item->getData();
269                 }
270
271         $statuses = $dao->getStatuses($item);
272
273         if (!empty($_POST['resend'])) {
```

```
274             $dao->addStatus($item, Message::MESSAGE_STATUS_TO_RETRY);
275             $app->redirect(MAINURL.'/messages/edit/'.$item->getId());
276             return;
277         }
278 
279         $item->dataHuman = self::getHumanUrl($item);
280                 $mime = '*/*';
281                 if (Message::KIND_PHOTO_MSG == $item->getKind()) {
282                         $mime = 'image/*';
283                 }
284                 if (Message::KIND_AUDIO_MSG == $item->getKind()) {
285                         $mime = 'audio/*';
286                 }
287                 if (Message::KIND_VIDEO_MSG == $item->getKind()) {
288                         $mime = 'video/*';
289                 }
290 
291                 $item->setUserId($user->getId());
292 
293                 $dateParts = explode('-', $_POST['stime_date']);
294                 $timeParts = explode(':', $_POST['stime_time']);
295                 $stime = mktime($timeParts[0], $timeParts[1], $timeParts[2], $dateParts[1], $dateParts[2],
    $dateParts[0]);
296                 $item->setStime($stime);
297                 if ($item->getKind() == Message::KIND_TEXT_MSG) {
298                         $item->setData($_POST['data']);
299                 }
300         $item->setGroupId($_POST['group_id']);
301         $item->setSenderId($_POST['sender_id']);
302 
303                 $validator = new \Valitron\Validator($_POST);
304                 $validator->addRule('credits', function ($name, $value) use ($user, $userDAO) {
305                         if ($userDAO->hasRole('ADMIN')) return true;
306 
307                         return $user->getCredits() > 0;
308                 });
309                 $validator->addRule('time', function ($name, $value) {
310                         $value = explode(':', $value);
311                         if (count($value)!=3) return false;
312 
313                         return true;
314                 });
315         $validator->rule('required', 'group_id');
316         $validator->label('Group');
317         $validator->rule('required', 'sender_id');
318         $validator->label('Sender');
319                 $validator->rule('date', 'stime_date');
320                 $validator->label('Date');
321                 $validator->rule('time', 'stime_time');
322                 $validator->label('Time');
323                 $validator->rule('required', 'data');
324                 $validator->rule('credits', 'data');
325                 $validator->label('Credits');
326 
327                 if($validator->validate()) {
328                         $item = $dao->save($item);
329                         if (!empty($_POST['send']) && empty($statuses)) {
330                                 $dao->addStatus($item,
    Message::MESSAGE_STATUS_TO_SEND);
331                                 if (!$userDAO->hasRole('ADMIN')) {
332                     $numbers = $groupDAO->getNumbers($item->getGroupId());
333 
334                                         $user->setCredits($user->getCredits()-count($numbers));
335                                         $userDAO->save($user);
336                                 }
337                         }
338 
339                         $app->redirect(MAINURL.'/messages/edit/'.$item->getId());
340                 } else {
341                         $app->view->set('menu', 'messages');
342                         $app->view->set('id', $id);
343                         $app->view->set('item', $item);
344                         $app->view->set('groups', $groups);
345             $app->view->set('senders', $senders);
346                         $app->view->set('statuses', $statuses);
347                         $app->view->set('errors', $validator->errors());
348                         $app->view->set('mime', $mime);
349 
350                         $app->render('messages/edit.twig.html');
351                 }
352         }
```

10.13.2.10 static GitGis\Whatsapp\MessagesController::postUploadPage (*$id*) [static]

Process upload multimedia file.

Definition at line 201 of file MessagesController.php.

```
201                                                 {
202                 $app = \Slim\Slim::getInstance();
203                 $dao = new MessageDAO();
204 
205                 $item = $dao->fetch($id);
206                 if (empty($item)) {
207                         return $app->notFound();
208                 }
209                 $statuses = $dao->getStatuses($item);
210 
211                 if (!empty($_FILES['file'])) {
212                         $dirPrefix = $id % 100;
213                         if (!file_exists(MAINDIR.'/uploads/'.$dirPrefix)) {
214                                 mkdir(MAINDIR.'/uploads/'.$dirPrefix);
215                         }
216                         $fileName = $id.'.'.$_FILES['file']['name'];
217                         move_uploaded_file($_FILES['file']['tmp_name'], MAINDIR.'/uploads/'.$dirPrefix.'/'.
    $fileName);
218 
219                         $item->setData($dirPrefix.'/'.$fileName);
220                         $dao->save($item);
221                 }
222         }
```

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

- src/GitGis/Whatsapp/MessagesController.php

10.14 GitGis\Pager Class Reference

Pagination class.

Public Member Functions

- __construct ($url, $pageSize=50)
  Constructor.
- setUrl ($url)
  Set base URL for page links.
- getPageSize ()
  Returns page size (number of rows, 50 by default)
- setPageSize ($pageSize)
  Overrides page size (50 by default)
- getPage ()
  Get current page.
- setPage ($page)
  Set current page.
- getCount ()
  Get number of rows currently displayed on the list.
- setCount ($count)
  Set number of rows currently displayed on the list.
- getTotal ()
  Get total number of rows.

- setTotal ($total)

  Set total number of rows.
- createHref ($page)

  Creates link to specified page relative to url.
- getLastPage ()

  Returns number of last page.
- getStartPage ()

  Returns first page for pager: usually currentPage - MARGIN.
- getEndPage ()

  Returns last page for pager: usually currentPage + MARGIN.
- getQueryArray ($query)

  Query is an array of parameters for SQL.

Static Public Member Functions

- static unparse_url ($parsed_url)

  Reverse to `parse_url`

Public Attributes

- const MARGIN = 3

10.14.1 Detailed Description

Pagination class.

Usage: $pager = new Pager(MAINURL.'/messages/', 25); $pager->setPage($page); // set current page $query = $pager->getQueryArray($query); $queryResult = $dao->getList($query); // fetches list of rows $pager->setCount(count($queryResult['list'])); // set number of rows fetched if (isset($list['total'])) $pager->setTotal($queryResult['total']); // Optionally set total number of rows

Definition at line 17 of file Pager.php.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 GitGis\Pager::__construct (*$url*, *$pageSize* = `50`)

Constructor.

Initialize base URL for links and pageSize

Parameters

string	*$url*	
number	*$pageSize*	

Definition at line 76 of file Pager.php.

```
76                                                      {
77                 $this->url = $url;
78                 $this->pageSize = $pageSize;
79         }
```

10.14.3 Member Function Documentation

10.14.3.1 GitGis\Pager::createHref (*$page*)

Creates link to specified page relative to url.

Parameters

number	*$page*	

Returns

string

Definition at line 174 of file Pager.php.

```
174                                     {
175                 if ($page <= 0) {
176                         return $this->url;
177                 }
178
179                 $retVal = parse_url($this->url);
180                 $retVal['path'] .= $page."/";
181
182                 return Pager::unparse_url($retVal);
183         }
```

10.14.3.2 GitGis\Pager::getCount ( )

Get number of rows currently displayed on the list.

Usually same as pageSize except last page when can be fewer rows

Returns

number

Definition at line 133 of file Pager.php.

```
133                                     {
134                 return $this->count;
135         }
```

10.14.3.3 GitGis\Pager::getEndPage ( )

Returns last page for pager: usually currentPage + MARGIN.

Returns

number

Definition at line 218 of file Pager.php.

```
218                                     {
219                 $endPage = $this->getPage() + Pager::MARGIN;
220                 if ($endPage > $this->getLastPage()) {
221                         $endPage = $this->getLastPage();
222                 }
223                 return $endPage;
224         }
```

10.14.3.4 GitGis\Pager::getLastPage ( )

Returns number of last page.

Returns

number

Definition at line 190 of file Pager.php.

```
190                                     {
191                 if ($this->total == -1) return 0;
192
193                 $lastPage = floor(($this->total-1) / $this->pageSize);
194                 if ($lastPage < 0) {
195                         $lastPage = 0;
196                 }
197                 return $lastPage;
198         }
```

10.14.3.5 GitGis\Pager::getPage ( )

Get current page.

Returns

number

Definition at line 113 of file Pager.php.

```
113                                     {
114                 return $this->page;
115         }
```

10.14.3.6 GitGis\Pager::getPageSize ( )

Returns page size (number of rows, 50 by default)

Returns

number

Definition at line 95 of file Pager.php.

```
95                                      {
96                 return $this->pageSize;
97         }
```

10.14.3.7 GitGis\Pager::getQueryArray (*$query*)

Query is an array of parameters for SQL.

This function adds start and limit variables to the array based on current page and pagesize

Parameters

array	*$query*	

Returns

number

Definition at line 233 of file Pager.php.

```
233                                             {
234                 $query['limit'] = $this->pageSize;
235                 $query['start'] = $this->pageSize * $this->page;
236
237                 return $query;
238         }
```

10.14.3.8 GitGis\Pager::getStartPage ( )

Returns first page for pager: usually currentPage - MARGIN.

Returns

number

Definition at line 205 of file Pager.php.

```
205                                     {
206                 $startPage = 0;
207                 if ($this->getPage() > (Pager::MARGIN+1)) {
208                         $startPage = $this->getPage() - Pager::MARGIN;
209                 }
210                 return startPage;
211         }
```

10.14.3.9 GitGis\Pager::getTotal ( )

Get total number of rows.

Returns

number

Definition at line 153 of file Pager.php.

```
153                                     {
154                 return $this->total;
155         }
```

10.14.3.10 GitGis\Pager::setCount (*$count*)

Set number of rows currently displayed on the list.

Usually same as pageSize except last page when can be fewer rows

Parameters

number	*$count*	

Definition at line 144 of file Pager.php.

```
144                                     {
145                 $this->count = $count;
146         }
```

10.14.3.11 GitGis\Pager::setPage (*$page*)

Set current page.

Parameters

unknown	*$page*	

Definition at line 122 of file Pager.php.

```
122                                     {
123                 $this->page = $page;
124         }
```

10.14.3.12 GitGis\Pager::setPageSize (*$pageSize*)

Overrides page size (50 by default)

Parameters

number	*$pageSize*	

Definition at line 104 of file Pager.php.

```
104                                             {
105                 $this->pageSize = $pageSize;
106         }
```

10.14.3.13 GitGis\Pager::setTotal (*$total*)

Set total number of rows.

If set to -1 total is unknown (no link to last page in pager)

Returns

number

Definition at line 164 of file Pager.php.

```
164                                         {
165                 $this->total = $total;
166         }
```

10.14.3.14 GitGis\Pager::setUrl (*$url*)

Set base URL for page links.

Parameters

unknown	*$url*	

Definition at line 86 of file Pager.php.

```
86                                      {
87                 $this->url = $url;
88         }
```

10.14.3.15 static GitGis\Pager::unparse_url (*$parsed_url*) [static]

Reverse to `parse_url`

Parameters

array	*$parsed_url*	

Returns

string

Definition at line 246 of file Pager.php.

```
246                                                     {
247                 $scheme   = isset($parsed_url['scheme']) ? $parsed_url['scheme'] . '://' : '';
248                 $host     = isset($parsed_url['host']) ? $parsed_url['host'] : '';
249                 $port     = isset($parsed_url['port']) ? ':' . $parsed_url['port'] : '';
250                 $user     = isset($parsed_url['user']) ? $parsed_url['user'] : '';
251                 $pass     = isset($parsed_url['pass']) ? ':' . $parsed_url['pass']  : '';
252                 $pass     = ($user || $pass) ? "$pass@" : '';
253                 $path     = isset($parsed_url['path']) ? $parsed_url['path'] : '';
254                 $query    = isset($parsed_url['query']) ? '?' . $parsed_url['query'] : '';
255                 $fragment = isset($parsed_url['fragment']) ? '#' . $parsed_url['fragment'] : '';
256                 return "$scheme$user$pass$host$port$path$query$fragment";
257         }
```

10.14.4 Member Data Documentation

10.14.4.1 const GitGis\Pager::MARGIN = 3

Definition at line 27 of file Pager.php.

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

- src/GitGis/Pager.php

10.15 GitGis\Whatsapp\Model\Sender Class Reference

Sender entity.

Public Member Functions

- getId ()
  Id of sender.
- setId ($id)
- getUsername ()
  MSISDN (phone number) associated with sender.
- setUsername ($username)
- getIdentity ()
  String generated during registration (login)
- setIdentity ($identity)
- getNickname ()
  Human friendly name of sender.
- setNickname ($nickname)
- getPassword ()
  Password generated during registration.
- setPassword ($password)
- getUserId ()
- setUserId ($user_id)

10.15.1 Detailed Description

Sender entity.

Definition at line 7 of file Sender.php.

10.15.2 Member Function Documentation

10.15.2.1 GitGis\Whatsapp\Model\Sender::getId ( )

Id of sender.

Definition at line 54 of file Sender.php.

```
54                          {
55         return $this->id;
56     }
```

10.15.2.2 GitGis\Whatsapp\Model\Sender::getIdentity ( )

String generated during registration (login)

Definition at line 76 of file Sender.php.

```
76                                  {
77          return $this->identity;
78      }
```

10.15.2.3 GitGis\Whatsapp\Model\Sender::getNickname ( )

Human friendly name of sender.

Definition at line 87 of file Sender.php.

```
87                                  {
88          return $this->nickname;
89      }
```

10.15.2.4 GitGis\Whatsapp\Model\Sender::getPassword ( )

Password generated during registration.

Definition at line 98 of file Sender.php.

```
98                                  {
99          return $this->password;
100     }
```

10.15.2.5 GitGis\Whatsapp\Model\Sender::getUserId ( )

Definition at line 111 of file Sender.php.

```
111                                 {
112          return $this->user_id;
113     }
```

10.15.2.6 GitGis\Whatsapp\Model\Sender::getUsername ( )

MSISDN (phone number) associated with sender.

Definition at line 65 of file Sender.php.

```
65                                  {
66          return $this->username;
67      }
```

10.15.2.7 GitGis\Whatsapp\Model\Sender::setId (*$id*)

Definition at line 58 of file Sender.php.

```
58                                  {
59          $this->id = $id;
60      }
```

10.15.2.8 GitGis\Whatsapp\Model\Sender::setIdentity (*$identity*)

Definition at line 80 of file Sender.php.

```
80                                          {
81          $this->identity = $identity;
82      }
```

10.15.2.9 GitGis\Whatsapp\Model\Sender::setNickname (*$nickname*)

Definition at line 91 of file Sender.php.

```
91                                          {
92          $this->nickname = $nickname;
93      }
```

10.15.2.10 GitGis\Whatsapp\Model\Sender::setPassword (*$password*)

Definition at line 102 of file Sender.php.

```
102                                          {
103         $this->password = $password;
104     }
```

10.15.2.11 GitGis\Whatsapp\Model\Sender::setUserId (*$user_id*)

Definition at line 115 of file Sender.php.

```
115                                         {
116         $this->user_id = $user_id;
117     }
```

10.15.2.12 GitGis\Whatsapp\Model\Sender::setUsername (*$username*)

Definition at line 69 of file Sender.php.

```
69                                          {
70          $this->username = $username;
71      }
```

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

- src/GitGis/Whatsapp/Model/Sender.php

10.16 GitGis\Whatsapp\Model\SenderDAO Class Reference

Sender management class.

Public Member Functions

- getList ($params=array())
  Get list of senders based on query.
- fetch ($id)
  Fetches specified sender.
- save (Sender $item)
  Saves sender to DB.
- delete ($id)
  Deletes sender.

10.16.1 Detailed Description

Sender management class.

Definition at line 11 of file SenderDAO.php.

10.16.2 Member Function Documentation

10.16.2.1 GitGis\Whatsapp\Model\SenderDAO::delete (*$id*)

Deletes sender.

Parameters

number	*$id*	

Definition at line 191 of file SenderDAO.php.

```
191                                 {
192         $sender = $this->fetch($id);
193
194         $db = DBConnection::getInstance();
195
196         $query = $db->prepare("DELETE FROM chat WHERE `to` = :to");
197         $query->bindParam('to', $sender->getUsername());
198         $query->execute();
199
200         $query = $db->prepare("DELETE FROM messages WHERE sender_id = :sender_id");
201         $query->bindParam('sender_id', $id, \PDO::PARAM_INT);
202         $query->execute();
203
204         $query = $db->prepare("DELETE FROM senders WHERE id = :sender_id");
205         $query->bindParam('sender_id', $id, \PDO::PARAM_INT);
206         $query->execute();
207     }
```

10.16.2.2 GitGis\Whatsapp\Model\SenderDAO::fetch (*$id*)

Fetches specified sender.

Parameters

number	*$id*	

Returns

Definition at line 111 of file SenderDAO.php.

```
111                                 {
112         $db = DBConnection::getInstance();
113         $item = new Sender();
114
115         if (empty($id)) {
116             return $item;
117         }
118
119         $query = $db->prepare("SELECT * FROM senders WHERE id=:id ");
120         $query->bindParam('id', $id);
121         $query->execute();
122         $row = $query->fetch();
123         if (!empty($row)) {
124             $item->setId($row['id']);
125             $item->setUsername($row['username']);
126             $item->setIdentity($row['identity']);
127             $item->setNickname($row['nickname']);
128             $item->setPassword($row['password']);
129             $item->setUserId($row['user_id']);
130         }
131
132         return $item;
133     }
```

10.16.2.3 GitGis\Whatsapp\Model\SenderDAO::getList (*$params* = `array()`)

Get list of senders based on query.

Params could contain:

- start
- limit
- search - fetches only senders containing specified MSISDN number
- username - fetches only specific sender

Parameters

unknown	*$params*	

Returns

multitype:Ambigous <number, unknown> number multitype: mixed

Definition at line 25 of file SenderDAO.php.

```
25                                                  {
26         if (empty($params['start'])) $params['start'] = 0;
27         if (empty($params['limit'])) $params['limit'] = 100;
28
29         $db = DBConnection::getInstance();
30         $list = array();
31
32         $params['searchNumeric'] = preg_replace('![^0-9]*!', '', $params['search']);
33
34         $where = " WHERE (1=1) ";
35         if (!empty($params['searchNumeric'])) {
36             $where .= " AND (senders.username=:searchNumeric
37                         OR senders.nickname LIKE '%'||:searchNumeric||'%'
38                         ) ";
39         } else
40             if (!empty($params['search'])) {
41                 $where .= " AND senders.nickname LIKE CONCAT('%', :search, '%') ";
42             }
43         if (!empty($params['username'])) {
44             $where .= ' AND username=:username ';
45         }
46         if (!empty($params['user_id'])) {
47             $where .= ' AND user_id=:user_id ';
48         }
49
50         $sql = "SELECT COUNT(id) as total FROM senders ".$where;
51         $query = $db->prepare($sql);
52         if (!empty($params['search'])) {
53             $query->bindParam('search', $params['search']);
54         }
55         if (!empty($params['searchNumeric'])) {
56             $query->bindParam('searchNumeric', $params['searchNumeric']);
57         }
58         if (!empty($params['username'])) {
59             $query->bindParam('username', $params['username']);
60         }
61         if (!empty($params['user_id'])) {
62             $query->bindParam('user_id', $params['user_id']);
63         }
64         $query->execute();
65         $row = $query->fetch();
66         $total = $row['total'];
67
68         $sql = "SELECT * FROM senders ".$where." ORDER BY username LIMIT :start, :limit ";
69         $query = $db->prepare($sql);
70         $query->bindParam('start', $params['start'], \PDO::PARAM_INT);
71         $query->bindParam('limit', $params['limit'], \PDO::PARAM_INT);
72         if (!empty($params['search'])) {
73             $query->bindParam('search', $params['search']);
```

```
74          }
75          if (!empty($params['searchNumeric'])) {
76              $query->bindParam('searchNumeric', $params['searchNumeric']);
77          }
78          if (!empty($params['username'])) {
79              $query->bindParam('username', $params['username']);
80          }
81          if (!empty($params['user_id'])) {
82              $query->bindParam('user_id', $params['user_id']);
83          }
84          $query->execute();
85          while ($row = $query->fetch()) {
86              $item = new Sender();
87              $item->setId($row['id']);
88              $item->setUsername($row['username']);
89              $item->setIdentity($row['identity']);
90              $item->setNickname($row['nickname']);
91              $item->setPassword($row['password']);
92              $item->setUserId($row['user_id']);
93
94              $list[$row['id']] = $item;
95          }
96
97          return array(
98              'start' => !empty($params['start']) ? $params['start'] : 0,
99              'limit' => !empty($params['limit']) ? $params['limit'] : 0,
100              'total' => $total,
101              'list' => $list
102          );
103      }
```

10.16.2.4 GitGis\Whatsapp\Model\SenderDAO::save (Sender *$item*)

Saves sender to DB.

Parameters

Sender	*$item*	

Exceptions

\Exception	

Returns

Sender

Definition at line 142 of file SenderDAO.php.

```
142                                     {
143          if (empty($item)) {
144              throw new \Exception('Empty item');
145          }
146
147          if (!$item->getIdentity()) $item->setIdentity('');
148          if (!$item->getPassword()) $item->setPassword('');
149
150          $db = DBConnection::getInstance();
151          if ($item->getId() == 0) {
152              $query = $db->prepare("INSERT INTO senders
153                                     (username, identity, nickname, password, user_id)
154                                     VALUES
155                                     (:username, :identity, :nickname, :password, :user_id)");
156              $query->bindParam('username', $item->getUsername());
157              $query->bindParam('nickname', $item->getNickname());
158              $query->bindParam('identity', $item->getIdentity());
159              $query->bindParam('password', $item->getPassword());
160              $query->bindParam('user_id', $item->getUserId());
161              $query->execute();
162              $item->setId($db->lastInsertId());
163          } else {
164              $sql = ' UPDATE senders SET ';
165              if ($item->getIdentity() !== null) $sql.= ' identity = :identity, ';
166              if ($item->getPassword() !== null) $sql.= ' password = :password, ';
```

```
167             $sql.= ' nickname = :nickname, ';
168             $sql.= ' username = :username, ';
169             $sql.= ' user_id = :user_id ';
170             $sql.= ' WHERE id = :id ';
171 
172             $query = $db->prepare($sql);
173             $query->bindParam('id', $item->getId(), \PDO::PARAM_INT);
174             $query->bindParam('username', $item->getUsername());
175             $query->bindParam('identity', $item->getIdentity());
176             $query->bindParam('nickname', $item->getNickname());
177             $query->bindParam('password', $item->getPassword());
178             $query->bindParam('user_id', $item->getUserId());
179             $query->execute();
180 
181         }
182 
183         return $item;
184     }
```

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

- src/GitGis/Whatsapp/Model/SenderDAO.php

10.17 GitGis\Whatsapp\SendersController Class Reference

Senders controller.

Static Public Member Functions

- static getPage ($page=0)

  Get list of senders.
- static getEditPage ($id)

  Get edit sender form.
- static postEditPage ($id)

  Process edit sender form, validate, save to DB.
- static getConfirmPage ($id)

  Get confirmation/registration form.
- static postConfirmPage ($id)

  Process confirmation SMS code.
- static deletePage ($id)

  Show delete confirmation.
- static postDeletePage ($id)

  Deletes sender.

10.17.1 Detailed Description

Senders controller.

Definition at line 14 of file SendersController.php.

10.17.2 Member Function Documentation

10.17.2.1 static GitGis\Whatsapp\SendersController::deletePage (*$id*) [static]

Show delete confirmation.

Definition at line 214 of file SendersController.php.

```
214                                         {
215         $app = \Slim\Slim::getInstance();
216         $userDAO = new UserDAO();
217         if (!$userDAO->hasRole('ADMIN')) {
218             return $app->status(403);
219         }
220
221         $app->view->set('msg', 'It will delete sender and its messages.');
222         $app->render('confirm_delete.twig.html');
223     }
```

10.17.2.2 static GitGis\Whatsapp\SendersController::getConfirmPage (*$id*) [static]

Get confirmation/registration form.

Parameters

number	*$id*	

Definition at line 143 of file SendersController.php.

```
143                                             {
144         $app = \Slim\Slim::getInstance();
145         $dao = new SenderDAO();
146         $wdao = new WhatsappDAO();
147
148         $item = $dao->fetch($id);
149         if (empty($item) || empty($id)) {
150             return $app->notFound();
151         }
152
153         $errors = array();
154         $res = $wdao->sendSmsCode($item);
155         if (!empty($res)) {
156             $errors['debug'] = $res;
157         }
158         if ($item->getPassword() != '') {
159             $app->flash('info', 'Your confirmation code has been successfully verified');
160             $app->redirect(MAINURL.'/senders/edit/'.$id);
161             return ;
162         }
163
164         if ($res['status'] == 'fail') {
165             $app->view->set('errors', array('sms' => array( json_encode($res) )));
166         }
167
168         $app->view->set('menu', 'senders');
169         $app->view->set('id', $id);
170         $app->view->set('item', $item);
171         $app->view->set('errors', $errors);
172
173         $app->render('senders/confirm.twig.html');
174     }
```

10.17.2.3 static GitGis\Whatsapp\SendersController::getEditPage (*$id*) [static]

Get edit sender form.

Parameters

number	*$id*	

Definition at line 54 of file SendersController.php.

```
54                                          {
55         $app = \Slim\Slim::getInstance();
56         $dao = new SenderDAO();
57         $userDAO = new UserDAO();
58
59         $app->expires(time());
60
61         $item = $dao->fetch($id);
62         if (empty($item)) {
```

```
63              return $app->notFound();
64          }
65
66          $app->view->set('menu', 'senders');
67          $app->view->set('id', $id);
68          $app->view->set('item', $item);
69          $app->view->set('users', $userDAO->getList());
70
71          $app->render('senders/edit.twig.html');
72      }
```

10.17.2.4 static GitGis\Whatsapp\SendersController::getPage (*$page* = 0) [static]

Get list of senders.

Parameters

number	*$page*	

Definition at line 21 of file SendersController.php.

```
21                                              {
22          $app = \Slim\Slim::getInstance();
23          $dao = new SenderDAO();
24          $userDAO = new UserDAO();
25
26          $app->expires(time());
27
28          $query = $_GET;
29          if (!$userDAO->hasRole('ADMIN')) {
30              $strong = \Strong\Strong::getInstance();
31              $user = $strong->getUser();
32              $query['user_id'] = $user['id'];
33          }
34          $pager = new Pager(MAINURL.'/messages/', 25);
35          $pager->setPage($page);
36          $query = $pager->getQueryArray($query);
37          $list = $dao->getList($query);
38          $pager->setCount(count($list['list']));
39          if (isset($list['total'])) $pager->setTotal($list['total']);
40
41          $app->view->set('menu', 'senders');
42          $app->view->set('query', $query);
43          $app->view->set('result', $list);
44          $app->view->set('pager', $pager);
45
46          $app->render('senders/list.twig.html');
47      }
```

10.17.2.5 static GitGis\Whatsapp\SendersController::postConfirmPage (*$id*) [static]

Process confirmation SMS code.

Parameters

number	*$id*	

Definition at line 181 of file SendersController.php.

```
181                                             {
182         $app = \Slim\Slim::getInstance();
183         $dao = new SenderDAO();
184         $wdao = new WhatsappDAO();
185
186         $item = $dao->fetch($id);
187         if (empty($item) || empty($id)) {
188             return $app->notFound();
189         }
190
191         $smscode = $_POST['smscode'];
192         $smscode = preg_replace('![^0-9]*!', '', $smscode);
193
194         $errors = array();
```

```
195         $res = $wdao->confirmSmsCode($item, $smscode);
196         if (!empty($res)) {
197             $errors['debug'] = $res;
198 
199             $app->view->set('menu', 'senders');
200             $app->view->set('id', $id);
201             $app->view->set('item', $item);
202             $app->view->set('errors', $errors);
203 
204             $app->render('senders/confirm.twig.html');
205         } else {
206             $app->flash('info', 'Your confirmation code has been successfully verified');
207             $app->redirect(MAINURL.'/senders/edit/'.$id);
208         }
209     }
```

10.17.2.6 static GitGis\Whatsapp\SendersController::postDeletePage (*$id*) [static]

Deletes sender.

Definition at line 228 of file SendersController.php.

```
228                                             {
229         $app = \Slim\Slim::getInstance();
230         $userDAO = new UserDAO();
231         if (!$userDAO->hasRole('ADMIN')) {
232             return $app->status(403);
233         }
234 
235         if (!empty($_POST['yes'])) {
236             $dao = new SenderDAO();
237             $dao->delete($id);
238         }
239 
240         return $app->redirect(MAINURL.'/senders');
241     }
```

10.17.2.7 static GitGis\Whatsapp\SendersController::postEditPage (*$id*) [static]

Process edit sender form, validate, save to DB.

Parameters

unknown	*$id*	

Returns

boolean

Definition at line 80 of file SendersController.php.

```
80                                              {
81         $app = \Slim\Slim::getInstance();
82         $dao = new SenderDAO();
83         $userDAO = new UserDAO();
84 
85         $item = $dao->fetch($id);
86         if (empty($item)) {
87             return $app->notFound();
88         }
89 
90         $_POST['username'] = preg_replace('![^0-9]*!', '', $_POST['username']);
91         $item->setNickname($_POST['nickname']);
92         if (empty($id)) {
93             $item->setUsername($_POST['username']);
94         }
95 
96         if ($userDAO->hasRole('ADMIN')) {
97             $item->setUserId($_POST['user_id']);
98         } else {
99             if (empty($id)) {
```

```
100                 $strong = \Strong\Strong::getInstance();
101                 $user = $strong->getUser();
102                 $item->setUserId($user['id']);
103             }
104         }
105 
106         $validator = new \Valitron\Validator($_POST);
107         $validator->addRule('unique_username', function ($name, $value) use ($id, $dao) {
108             $list = $dao->getList(array('username' => $value));
109             if (!empty($list['list'])) {
110                 foreach ($list['list'] as $item) {
111                     if ($item->getId() != $id) return false;
112                 }
113             }
114 
115             return true;
116         }, 'is not unique');
117 
118         $validator->rule('unique_username', 'username');
119         $validator->rule('required', 'nickname');
120         $validator->rule('required', 'username');
121         $validator->label('MSISDN');
122 
123         if($validator->validate()) {
124             $item = $dao->save($item);
125             $app->flash('info', 'Sender '.$item->getNickname().' has been created successfully');
126             $app->redirect(MAINURL.'/senders/edit/'.$item->getId());
127         } else {
128             $app->view->set('menu', 'senders');
129             $app->view->set('id', $id);
130             $app->view->set('users', $userDAO->getList());
131             $app->view->set('item', $item);
132             $app->view->set('errors', $validator->errors());
133 
134             $app->render('senders/edit.twig.html');
135         }
136     }
```

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

- src/GitGis/Whatsapp/SendersController.php

10.18 GitGis\Whatsapp\Model\User Class Reference

User entity used during authentication.

Public Member Functions

- getId ()

  Id of user.
- setId ($id)
- getUsername ()

  Username / login.
- setUsername ($username)
- getPassword ()

  Password for authentication stored with MD5.
- setPassword ($password)
- getRoles ()
- setRoles ($roles)
- getCredits ()
- setCredits ($credits)
- getCtime ()

  Creation timestamp.

- setCtime ($ctime)
- getDtime ()

  Delete timestamp.
- setDtime ($dtime)

10.18.1 Detailed Description

User entity used during authentication.

Definition at line 8 of file User.php.

10.18.2 Member Function Documentation

10.18.2.1 GitGis\Whatsapp\Model\User::getCredits ( )

Definition at line 110 of file User.php.

```
110                                     {
111                 return $this->credits;
112         }
```

10.18.2.2 GitGis\Whatsapp\Model\User::getCtime ( )

Creation timestamp.

Definition at line 121 of file User.php.

```
121                                     {
122                 return $this->ctime;
123         }
```

10.18.2.3 GitGis\Whatsapp\Model\User::getDtime ( )

Delete timestamp.

Definition at line 132 of file User.php.

```
132                                     {
133                 return $this->dtime;
134         }
```

10.18.2.4 GitGis\Whatsapp\Model\User::getId ( )

Id of user.

Definition at line 62 of file User.php.

```
62                                      {
63                 return $this->id;
64         }
```

10.18.2.5 GitGis\Whatsapp\Model\User::getPassword ( )

Password for authentication stored with MD5.

Definition at line 84 of file User.php.

```
84                                      {
85                 return $this->password;
86         }
```

10.18.2.6 GitGis\Whatsapp\Model\User::getRoles ( )

Definition at line 97 of file User.php.

```
97                                  {
98                 return $this->roles;
99         }
```

10.18.2.7 GitGis\Whatsapp\Model\User::getUsername ( )

Username / login.

Definition at line 73 of file User.php.

```
73                                  {
74                 return $this->username;
75         }
```

10.18.2.8 GitGis\Whatsapp\Model\User::setCredits (*$credits*)

Definition at line 114 of file User.php.

```
114                                         {
115                 $this->credits = $credits;
116         }
```

10.18.2.9 GitGis\Whatsapp\Model\User::setCtime (*$ctime*)

Definition at line 125 of file User.php.

```
125                                         {
126                 $this->ctime = $ctime;
127         }
```

10.18.2.10 GitGis\Whatsapp\Model\User::setDtime (*$dtime*)

Definition at line 136 of file User.php.

```
136                                         {
137                 $this->dtime = $dtime;
138         }
```

10.18.2.11 GitGis\Whatsapp\Model\User::setId (*$id*)

Definition at line 66 of file User.php.

```
66                                  {
67                 $this->id = $id;
68         }
```

10.18.2.12 GitGis\Whatsapp\Model\User::setPassword (*$password*)

Definition at line 88 of file User.php.

```
88                                          {
89                 $this->password = $password;
90         }
```

10.18.2.13 GitGis\Whatsapp\Model\User::setRoles (*$roles*)

Definition at line 101 of file User.php.

```
101                                     {
102                 $this->roles = $roles;
103         }
```

10.18.2.14 GitGis\Whatsapp\Model\User::setUsername (*$username*)

Definition at line 77 of file User.php.

```
77                                              {
78                 $this->username = $username;
79         }
```

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

- src/GitGis/Whatsapp/Model/User.php

10.19 GitGis\Whatsapp\Model\UserDAO Class Reference

Authentication/user management class.

Public Member Functions

- getList ($params=array())

  Get list of users based on query.
- fetch ($id)

  Fetches user with specific id.
- save (User $item)

  Stores user into db.
- delete ($id)

  Deletes user.
- hasRole ($role, User $user=null)

10.19.1 Detailed Description

Authentication/user management class.

Definition at line 11 of file UserDAO.php.

10.19.2 Member Function Documentation

10.19.2.1 GitGis\Whatsapp\Model\UserDAO::delete (*$id*)

Deletes user.

Definition at line 186 of file UserDAO.php.

```
186                                 {
187                 $db = DBConnection::getInstance();
188
189                 $sql = " UPDATE users SET password='', dtime=:dtime WHERE id=:id ";
190
191                 $query = $db->prepare($sql);
192                 $query->bindParam('id', $id, \PDO::PARAM_INT);
193                 $query->bindParam('dtime', time());
194                 $query->execute();
195
196         }
```

10.19.2.2 GitGis\Whatsapp\Model\UserDAO::fetch (*$id*)

Fetches user with specific id.

Parameters

number	*$id*	

Returns

|Ambigous <NULL, >

Definition at line 96 of file UserDAO.php.

```
96                                  {
97                 $db = DBConnection::getInstance();
98                 $item = null;
99
100                 if (empty($id)) {
101                         $item = new User();
102                         return $item;
103                 }
104
105                 $query = $db->prepare("SELECT * FROM users WHERE id=:id ");
106                 $query->bindParam('id', $id);
107                 $query->execute();
108                 $row = $query->fetch();
109                 if (!empty($row)) {
110                         $item = new User();
111                         $item->setId($row['id']);
112                         $item->setUsername($row['username']);
113                         $item->setPassword('');
114                         $item->setRoles($row['roles']);
115                         $item->setCredits((int) $row['credits']);
116                         $item->setCtime($row['ctime']);
117                         $item->setDtime($row['dtime']);
118                 }
119
120                 return $item;
121         }
```

10.19.2.3 GitGis\Whatsapp\Model\UserDAO::getList (*$params* = `array()`)

Get list of users based on query.

Params could contain:

- start
- limit
- username - fetches only specific user

Parameters

array	*$params*	

Returns

multitype:Ambigous <number, unknown> number multitype: mixed

Definition at line 24 of file UserDAO.php.

```
24                                                  {
25                 if (empty($params['start'])) $params['start'] = 0;
26                 if (empty($params['limit'])) $params['limit'] = 100;
27
28                 $db = DBConnection::getInstance();
29                 $list = array();
30
31                 $where = ' WHERE (1=1) ';
32         if (!empty($params['search'])) {
33             $where .= " AND username like CONCAT ('%', :search, '%') ";
34         }
35                 if (!empty($params['username'])) {
36                         $where .= ' AND username=:username ';
37                 }
38                 if (isset($params['deleted'])) {
39                         if ($params['deleted'] > 0) {
40                                 $where .= ' AND dtime > 0 ';
41                         } else {
42                                 $where .= ' AND NOT dtime > 0 ';
43                         }
44                 }
45
46                 $sql = "SELECT COUNT(id) as total FROM users ".$where;
47                 $query = $db->prepare($sql);
48                 if (!empty($params['username'])) {
49                         $query->bindParam('username', $params['username']);
50                 }
51         if (!empty($params['search'])) {
52             $query->bindParam('search', $params['search']);
53         }
54                 $query->execute();
55                 $row = $query->fetch();
56                 $total = $row['total'];
57
58                 $sql = "SELECT * FROM users ".$where." ORDER BY username LIMIT :start, :limit ";
59                 $query = $db->prepare($sql);
60                 $query->bindParam('start', $params['start'], \PDO::PARAM_INT);
61                 $query->bindParam('limit', $params['limit'], \PDO::PARAM_INT);
62                 if (!empty($params['username'])) {
63                         $query->bindParam('username', $params['username']);
64                 }
65         if (!empty($params['search'])) {
66             $query->bindParam('search', $params['search']);
67         }
68                 $query->execute();
69                 while ($row = $query->fetch()) {
70                         $item = new User();
71                         $item->setId($row['id']);
72                         $item->setUsername($row['username']);
73                         $item->setPassword('');
74                         $item->setRoles($row['roles']);
75                         $item->setCredits((int) $row['credits']);
76                         $item->setCtime($row['ctime']);
77                         $item->setDtime($row['dtime']);
78
79                         $list[$row['id']] = $item;
80                 }
81
82                 return array(
83                         'start' => !empty($params['start']) ? $params['start'] : 0,
84                         'limit' => !empty($params['limit']) ? $params['limit'] : 0,
85                         'total' => $total,
86                         'list' => $list
87                 );
88         }
```

10.19.2.4 GitGis\Whatsapp\Model\UserDAO::hasRole (*$role,* User *$user* = `null`)

Definition at line 198 of file UserDAO.php.

```
198                                                     {
199                 if (empty($user)) {
200                         $strong = \Strong\Strong::getInstance();
201                         $user = $strong->getUser();
202                 }
203
204                 if (is_array($user)) {
205                         $user = $this->fetch($user['id']);
206                 }
207
208                 return in_array($role, explode(',', $user->getRoles()));
209         }
```

10.19.2.5 GitGis\Whatsapp\Model\UserDAO::save (User *$item*)

Stores user into db.

Parameters

User	*$item*	

Exceptions

\Exception	

Returns

unknown

Definition at line 130 of file UserDAO.php.

```
130                                     {
131                 if (empty($item)) {
132                         throw new \Exception('Empty item');
133                 }
134
135                 if ($item->getCtime() == 0) {
136                         $item->setCtime(time());
137                 }
138
139                 $db = DBConnection::getInstance();
140                 if ($item->getId() == 0) {
141                         $query = $db->prepare("INSERT INTO users
142                                         (username, password, roles, credits, ctime, dtime)
143                                         VALUES
144                                         (:username, :password, :roles, :credits, :ctime, :dtime)");
145                         $query->bindParam('username', $item->getUsername());
146                         $query->bindParam('password', $item->getPassword());
147                         $query->bindParam('roles', $item->getRoles());
148                         $query->bindParam('credits', $item->getCredits());
149                         $query->bindParam('ctime', $item->getCtime());
150                         $query->bindParam('dtime', $item->getDtime());
151                         $query->execute();
152                         $item->setId($db->lastInsertId());
153                 } else {
154                         $sql = ' UPDATE users SET ';
155                         if ($item->getPassword() != '') $sql.= ' password = :password, ';
156                         if ($item->getCredits() !== null) $sql.= ' credits = :credits, ';
157                         $sql.= ' roles = :roles, ';
158                         $sql.= ' ctime = :ctime, ';
159                         $sql.= ' dtime = :dtime, ';
160                         $sql.= ' username = :username ';
161                         $sql.= ' WHERE id = :id ';
162
163                         $query = $db->prepare($sql);
164                         $query->bindParam('id', $item->getId(), \PDO::PARAM_INT);
165                         $query->bindParam('username', $item->getUsername());
166                         if ($item->getPassword() != '') {
```

```
167                                 $query->bindParam('password', $item->getPassword());
168                         }
169                         if ($item->getCredits() !== null) {
170                                 $query->bindParam('credits', $item->getCredits());
171                         }
172                         $query->bindParam('roles', $item->getRoles());
173                         $query->bindParam('ctime', $item->getCtime());
174                         $query->bindParam('dtime', $item->getDtime());
175                         $query->execute();
176
177                 }
178
179                 return $item;
180         }
```

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

- src/GitGis/Whatsapp/Model/UserDAO.php

10.20 GitGis\Whatsapp\UsersController Class Reference

User management controller.

Static Public Member Functions

- static getPage ($page=0)
  Get list of users.
- static getEditPage ($id)
  Get user edit form.
- static postEditPage ($id)
  Process user edit form.
- static deletePage ($id)
  Deletes message.

10.20.1 Detailed Description

User management controller.

Definition at line 11 of file UsersController.php.

10.20.2 Member Function Documentation

10.20.2.1 static GitGis\Whatsapp\UsersController::deletePage (*$id*) [static]

Deletes message.

Definition at line 152 of file UsersController.php.

```
152                                                 {
153                 $app = \Slim\Slim::getInstance();
154                 $dao = new UserDAO();
155                 $dao->delete($id);
156
157                 return $app->redirect(MAINURL.'/users');
158         }
```

10.20.2.2 static GitGis\Whatsapp\UsersController::getEditPage (*$id*) [static]

Get user edit form.

Parameters

number	*$id*	

Definition at line 49 of file UsersController.php.

```
49                                                  {
50                 $app = \Slim\Slim::getInstance();
51                 $dao = new UserDAO();
52
53                 if (!$dao->hasRole('ADMIN')) {
54                         return $app->status(403);
55                 }
56
57                 $app->expires(time());
58
59                 $item = $dao->fetch($id);
60                 if (empty($item)) {
61                         return $app->notFound();
62                 }
63
64                 $app->view->set('menu', 'users');
65                 $app->view->set('id', $id);
66                 $app->view->set('item', $item);
67                 $strong = \Strong\Strong::getInstance();
68                 $user = $strong->getUser();
69                 $app->view->set('user', $user);
70
71                 $app->render('users/edit.twig.html');
72         }
```

10.20.2.3 static GitGis\Whatsapp\UsersController::getPage (*$page* = 0) [static]

Get list of users.

Definition at line 16 of file UsersController.php.

```
16                                                  {
17                 $app = \Slim\Slim::getInstance();
18                 $dao = new UserDAO();
19
20                 if (!$dao->hasRole('ADMIN')) {
21                         return $app->status(403);
22                 }
23
24                 $app->expires(time());
25
26                 $query = $_GET;
27                 $query['deleted'] = (int) $query['deleted'];
28                 $pager = new Pager(MAINURL.'/users/', 25);
29                 $pager->setPage($page);
30                 $query = $pager->getQueryArray($query);
31                 $list = $dao->getList($query);
32                 $pager->setCount(count($list['list']));
33                 if (isset($list['total'])) $pager->setTotal($list['total']);
34
35                 $app->view->set('menu', 'users');
36         $app->view->set('query', $query);
37                 $app->view->set('result', $list);
38                 $app->view->set('pager', $pager);
39                 $app->view->set('app', $app);
40
41                 $app->render('users/list.twig.html');
42         }
```

10.20.2.4 static GitGis\Whatsapp\UsersController::postEditPage (*$id*) [static]

Process user edit form.

Parameters

number	*$id*	

Returns

boolean

Definition at line 80 of file UsersController.php.

```
80                                                  {
81                 $app = \Slim\Slim::getInstance();
82                 $dao = new UserDAO();
83
84                 if (!$dao->hasRole('ADMIN')) {
85                         return $app->status(403);
86                 }
87
88                 $item = $dao->fetch($id);
89                 if (empty($item)) {
90                         return $app->notFound();
91                 }
92
93                 $item->setUsername($_POST['username']);
94                 if ($_POST['credits'] > 0) {
95                         $item->setCredits($_POST['credits']);
96                 }
97                 if (is_array($_POST['roles'])) {
98                         $item->setRoles(implode(',', $_POST['roles']));
99                 } else {
100                         $item->setRoles('');
101                 }
102                 $item->setUsername($_POST['username']);
103                 if (!empty($_POST['password'])) {
104                         $item->setPassword(md5($_POST['password']));
105                 }
106
107                 $validator = new \Valitron\Validator($_POST);
108                 $validator->addRule('repeat', function ($name, $value) {
109                         if ($value != $_POST['password']) return false;
110
111                         return true;
112                 });
113                 $validator->addRule('unique_username', function ($name, $value) use ($id, $dao) {
114                         $list = $dao->getList(array('username' => $value));
115                         if (!empty($list['list'])) {
116                                 foreach ($list['list'] as $item) {
117                                         if ($item->getId() != $id) return false;
118                                 }
119                         }
120
121                         return true;
122                 }, 'is not unique');
123
124                 $validator->rule('unique_username', 'username');
125                 $validator->rule('repeat', 'repeat');
126                 $validator->label('Password repeat');
127                 if (empty($id)) {
128                         $validator->rule('required', 'password');
129                 }
130                 $validator->rule('required', 'username');
131                 $validator->label('Login');
132
133                 if ($validator->validate()) {
134                         $item = $dao->save($item);
135             if (empty($id)) {
136                 $app->flash('info', 'Account '.$item->getUsername().' has been created successfully');
137             }
138                         $app->redirect(MAINURL.'/users/edit/'.$item->getId());
139                 } else {
140                         $app->view->set('menu', 'users');
141                         $app->view->set('id', $id);
142                         $app->view->set('item', $item);
143                         $app->view->set('errors', $validator->errors());
144
145                         $app->render('users/edit.twig.html');
146                 }
147         }
```

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

- src/GitGis/Whatsapp/UsersController.php

10.21 GitGis\Whatsapp\Model\WhatsappDAO Class Reference

DAO responsible for communicating WhatsApp server.

Public Member Functions

- getDebug ()
  Is debugging enabled.
- setDebug ($debug)
  Set debugging mode.
- printDebug ($msg)
- sendSmsCode (Sender $item)
  Tells WhatsApp to send sms code for specified sender MSISDN.
- confirmSmsCode (Sender $item, $smscode)
  Based on sender's MSISDN and SMS code fetches identity and password for sender, then store it into DB.
- getWhatsProt (Sender $sender)
- sendMessage (Message $item)
  Send message to WhatsApp, then add statuses to DB.
- processPoll ()
  Polls for incomming messages for each sender.
- syncContacts ($groupId, $userId)
  Sync group contacts.

Protected Member Functions

- processIncomingMessages ($msgs, $debugMsg= '')
  Processes incomming messaged, stores statuses into DB.

10.21.1 Detailed Description

DAO responsible for communicating WhatsApp server.

Definition at line 21 of file WhatsappDAO.php.

10.21.2 Member Function Documentation

10.21.2.1 GitGis\Whatsapp\Model\WhatsappDAO::confirmSmsCode (Sender *$item*, *$smscode*)

Based on sender's MSISDN and SMS code fetches identity and password for sender, then store it into DB.

Parameters

Sender	*$item*	
string	*$smscode*	

Returns

array

Definition at line 107 of file WhatsappDAO.php.

```
107                                                                 {
108                 $dao = new SenderDAO();
109
110                 $debug = true;
111
112                 $username = $item->getUsername(); // Telephone number including the country code without '+' or '00'.
113                 $identity = $item->getIdentity(); // Obtained during registration with this API or using MissVenom (https://github.com/shirioko/MissVenom) to sniff from your phone.
114                 $nickname = $item->getNickname(); // This is the username displayed by WhatsApp clients.
115
116                 $oldDir = getcwd();
117                 chdir(MAINDIR.'/vendor/WhatsAPI/src/php');
118
119                 $res = array();
120
121                 ob_start();
122                 $w = new WhatsProt($username, $identity, $nickname, $debug);
123                 try {
124                         $w->eventManager()->bind('onCredentialsBad', function ($number) {
125                         });
126             $w->eventManager()->bind('onLoginFailed', function ($number) {
127             });
128
129                         $code = $w->codeRegister($smscode);
130                         ob_end_clean();
131
132                         if ($code->status == 'ok' && !empty($code->pw)) {
133                                 $item->setUsername($code->login);
134                                 $item->setPassword($code->pw);
135                                 $dao->save($item);
136                         }
137                 } catch (\Exception $ex) {
138                         $res[] = ob_get_contents();
139                         ob_end_clean();
140                 }
141
142                 chdir($oldDir);
143
144                 return $res;
145         }
```

10.21.2.2 GitGis\Whatsapp\Model\WhatsappDAO::getDebug ( )

Is debugging enabled.

Returns

boolean

Definition at line 30 of file WhatsappDAO.php.

```
30                                  {
31                 return $this->debug;
32         }
```

10.21.2.3 GitGis\Whatsapp\Model\WhatsappDAO::getWhatsProt (Sender *$sender*)

Definition at line 147 of file WhatsappDAO.php.

```
147                                                 {
148                 $debugMsg = '';
149
150                 $username = $sender->getUsername(); // Telephone number including the country code without
    '+' or '00'.
151                 $identity = $sender->getIdentity(); // Obtained during registration with this API or using
    MissVenom (https://github.com/shirioko/MissVenom) to sniff from your phone.
152                 $nickname = $sender->getNickname(); // This is the username displayed by WhatsApp clients.
153
154         if (!empty($this->whatsProts[$username])) {
155             return array($this->whatsProts[$username], $debugMsg);
156         }
157
158         $debug = true;
159
160                 try {
161                         $debugMsg = '';
162                         ob_start();
163                         $w = new WhatsProt($username, $identity, $nickname, $debug);
164
165                         $w->eventManager()->bind('onCredentialsBad', function ($number) use ($debugMsg) {
166                                 $debugMsg .= 'Bad credentials';
167                                 throw new \Exception($debugMsg, 403);
168                         });
169             $w->eventManager()->bind('onLoginFailed', function ($number) use ($debugMsg) {
170                 $debugMsg .= 'Bad credentials';
171                 throw new \Exception($debugMsg, 403);
172             });
173
174                         $w->eventManager()->bind('onGetMessage', function ($number, $from, $id, $type, $t,
    $name,
175                                         $data) use ($debugMsg) {
176                                 if ($type == 'chat') {
177                                         $chatDAO = new ChatDAO();
178
179                                         $from = preg_replace('!@.*!', '', $from);
180
181                                         $chat = new Chat();
182                                         $chat->setFrom($from);
183                                         $chat->setFromNickname($name);
184                                         $chat->setTo($number);
185                                         $chat->setData($data);
186                                         $chat->setCtime($t);
187                                         $chat->setWhatsappId($id);
188
189                                         $chat = $chatDAO->save($chat);
190 //                                      print_r(array($number, $from, $id, $type, $t, $name, $data));
191                                 }
192                         });
193
194 //                      $w->eventManager()->bind('onGetImage', function ($number, $from, $id, $type, $t,
    $name,
195 //                                      $size, $url, $file, $mimetype, $filehast, $width, $height, $data)
    use ($debugMsg) {
196 //                              print_r(array($number, $from, $id, $type, $t, $name, $size, $url, $file,
    $mimetype, $filehast, $width, $height, $data));
197 //                      });
198
199                         $w->connect();
200                         $w->loginWithPassword($sender->getPassword());
201
202                         $debugMsg .= ob_get_contents();
203                         ob_clean();
204
205                         if (!empty($w)) {
206                                 $this->whatsProts[$username] = $w;
207                         }
208
209                 } catch (\Exception $ex) {
210                         $debugMsg .= ob_get_contents();
211                         ob_end_clean();
212
213                         if ($ex->getCode() == 403) {
214                                 $senderDao = new SenderDAO();
215
216                                 $senderMod = $senderDao->fetch($sender->getId());
217                                 $senderMod->setPassword('');
218                                 $senderDao->save($senderMod);
219                         }
220                 }
221
222                 return array($this->whatsProts[$username], $debugMsg);
```

```
223         }
```

10.21.2.4 GitGis\Whatsapp\Model\WhatsappDAO::printDebug (*$msg*)

Definition at line 43 of file WhatsappDAO.php.

```
43                                      {
44                 if (!$this->debug) return;
45                 echo $msg;
46         }
```

10.21.2.5 GitGis\Whatsapp\Model\WhatsappDAO::processIncomingMessages (*$msgs, $debugMsg = ''*) [protected]

Processes incomming messaged, stores statuses into DB.

Parameters

array	*$msgs*	

Definition at line 342 of file WhatsappDAO.php.

```
342                                                                                 {
343                 $dao = new MessageDAO();
344
345                 foreach ($msgs as $msg) {
346                         $attrs = $msg->getAttributes();
347                         $whatsapp_id = $attrs['id'];
348                         $statuses = $dao->getStatusesByWhatsappId($whatsapp_id);
349
350                         foreach ($msg->getChildren() as $child) {
351                                 $attrs = $child->getAttributes();
352                                 if ($attrs['xmlns'] == 'urn:xmpp:receipts') {
353                                         foreach ($statuses as $status) {
354                                                 $item = $dao->fetch($status['message_id']);
355                                                 $dao->addStatus($item,
      Message::MESSAGE_STATUS_RECEIVED_BY_SERVER, $debugMsg, $status['
      target'], $whatsapp_id);
356                                         }
357                                 }
358                         }
359                 }
360         }
```

10.21.2.6 GitGis\Whatsapp\Model\WhatsappDAO::processPoll ( )

Polls for incomming messages for each sender.

Definition at line 365 of file WhatsappDAO.php.

```
365                                     {
366         $groupDao = new GroupDAO();
367         $senderDao = new SenderDAO();
368                 $dao = new MessageDAO();
369
370                 $groups = $groupDao->getList();
371         $senders = $senderDao->getList();
372                 foreach ($senders['list'] as $sender) {
373                         if ('' == $sender->getPassword()) continue;
374                         $username = $sender->getUsername(); // Telephone number including the country code
      without '+' or '00'.
375                         $identity = $sender->getIdentity(); // Obtained during registration with this API
      or using MissVenom (https://github.com/shirioko/MissVenom) to sniff from your phone.
376                         $nickname = $sender->getNickname(); // This is the username displayed by WhatsApp
      clients.
377
378                         $oldDir = getcwd();
379                         chdir(MAINDIR.'/vendor/WhatsAPI/src/php');
380
381                         $debugMsg = '';
```

```
382                         $msgs = array();
383
384                         try {
385                                 list($w, $debugMsg) = $this->getWhatsProt($sender);
386                                 $this->printDebug($debugMsg);
387                                 if (empty($w)) return;
388
389                                 ob_start();
390                                 $w->connect();
391                                 $w->loginWithPassword($sender->getPassword());
392
393                                 $w->pollMessages();
394                                 $msgs = $w->getMessages();
395
396                                 $debugMsg = ob_get_contents();
397                                 ob_end_clean();
398                         } catch (\Exception $ex) {
399                                 $debugMsg = ob_get_contents();
400                                 ob_end_clean();
401
402                                 echo ($ex);
403
404                                 $this->printDebug($debugMsg);
405 //                              $dao->addStatus($item, Message::MESSAGE_STATUS_ERROR, $debugMsg);
406                                 $debugMsg = '';
407                         }
408
409                         $this->printDebug($debugMsg);
410                         $this->processIncomingMessages($msgs, $debugMsg);
411
412                         chdir($oldDir);
413                 }
414         }
```

10.21.2.7 GitGis\Whatsapp\Model\WhatsappDAO::sendMessage (Message *$item*)

Send message to WhatsApp, then add statuses to DB.

Parameters

Message	*$item*	

Definition at line 230 of file WhatsappDAO.php.

```
230                                                     {
231         $groupDao = new GroupDAO();
232         $senderDao = new SenderDAO();
233                 $dao = new MessageDAO();
234
235         $group = $groupDao->fetch($item->getGroupId());
236         $sender = $senderDao->fetch($item->getSenderId());
237
238                 $username = $sender->getUsername(); // Telephone number including the country code without '+' or '00'.
239                 $identity = $sender->getIdentity(); // Obtained during registration with this API or using MissVenom (https://github.com/shirioko/MissVenom) to sniff from your phone.
240                 $nickname = $sender->getNickname(); // This is the username displayed by WhatsApp clients.
241                 if ('' == $sender->getPassword()) {
242                         $dao->addStatus($item, Message::MESSAGE_STATUS_ERROR, 'Sender not registered - go sender and use confirm SMS function');
243                         return;
244                 }
245
246                 $oldDir = getcwd();
247         chdir(MAINDIR.'/vendor/WhatsAPI/src/php');
248
249                 $debugMsg = '';
250                 $msgs = array();
251
252                 try {
253                         $debugMsg = '';
254
255                         list($w, $debugMsg) = $this->getWhatsProt($sender);
256                         $this->printDebug($debugMsg);
257                         if (empty($w)) return;
258
259                         $whatsapp_id = '';
260                         $targets = explode(',', $item->getTarget());
```

```
261             $targets = array_combine($targets, $targets);
262
263             $statuses = $dao->getStatuses($item);
264             foreach ($statuses as $status) {
265                 if ($status['status'] ==
    Message::MESSAGE_STATUS_RECEIVED_BY_SERVER) {
266                     unset($targets[$status['target']]);
267                 }
268             }
269
270             $targets = array_keys($targets);
271
272                         foreach ($targets as $target) {
273                                 $sendRetVal = null;
274                                 switch ($item->getKind()) {
275                                         case Message::KIND_TEXT_MSG:
276                                                 $sendRetVal = $w->sendMessage($target, $item->getData());
277                                                 break;
278
279                                         case Message::KIND_PHOTO_MSG:
280                         $sendRetVal = $w->sendMessageImage($target, MAINDIR.'/uploads/'.$item->getData(),
    false);
281                                                 break;
282
283                                         case Message::KIND_AUDIO_MSG:
284                                                 $sendRetVal = $w->sendMessageAudio($target, MAINDIR.'
    /uploads/'.$item->getData(), false);
285                                                 break;
286
287                                         case Message::KIND_VIDEO_MSG:
288                         $sendRetVal = $w->sendMessageVideo($target, MAINDIR.'/uploads/'.$item->getData(),
    false);
289                                                 break;
290
291                                         default:
292                                                 $sendRetVal = false;
293                                                 break;
294                                 }
295
296                                 $whatsapp_id = '';
297
298                                 $sentNodes = $w->getSentNodes();
299                                 $debugMsgReceiver = ob_get_contents();
300                                 ob_end_clean();
301
302                                 foreach ($sentNodes as $node) {
303                                         if ($node->getTag() == 'message') {
304                                                 foreach ($node->getChildren() as $child) {
305                                                         if ($child->getTag() == 'body') {
306                                                                 $attrs = $node->getAttributes();
307                                                                 $whatsapp_id = $attrs['id'];
308                                                                 break;
309                                                         }
310                                                 }
311                                         }
312                                 }
313
314                                 $this->printDebug($debugMsgReceiver);
315                                 $dao->addStatus($item,
    Message::MESSAGE_STATUS_SENT, $debugMsgReceiver, $target, $whatsapp_id);
316                         }
317
318                         $w->pollMessages();
319                         $msgs = $w->getMessages();
320
321                         $debugMsg .= ob_get_contents();
322                         ob_end_clean();
323
324                 } catch (\Exception $ex) {
325                         $debugMsg .= ob_get_contents();
326                         ob_end_clean();
327
328                         $this->printDebug($debugMsg);
329                         $dao->addStatus($item, Message::MESSAGE_STATUS_ERROR,
    $debugMsg);
330                         $debugMsg = '';
331                 }
332
333                 $this->printDebug($debugMsg);
334                 $this->processIncomingMessages($msgs, $debugMsg);
335         }
```

10.21.2.8 GitGis\Whatsapp\Model\WhatsappDAO::sendSmsCode (Sender *$item*)

Tells WhatsApp to send sms code for specified sender MSISDN.

Parameters

Sender	*$item*	

Returns

multitype:string

Definition at line 55 of file WhatsappDAO.php.

```
55                                                  {
56                 $dao = new SenderDAO();
57
58                 $debug = true;
59
60                 $username = $item->getUsername(); // Telephone number including the country code without
       '+' or '00'.
61                 $identity = urlencode(sha1('fakeimei'.$item->getId(), true)); // Obtained during
       registration with this API or using MissVenom (https://github.com/shirioko/MissVenom) to sniff from your phone.
62                 $nickname = $item->getNickname(); // This is the username displayed by WhatsApp clients.
63
64                 $item->setIdentity($identity);
65                 $item->setPassword('');
66
67                 $dao->save($item);
68
69                 $oldDir = getcwd();
70                 chdir(MAINDIR.'/vendor/WhatsAPI/src/php');
71
72                 $res = array();
73
74                 ob_start();
75                 $w = new WhatsProt($username, $identity, $nickname, $debug);
76                 try {
77                         $w->eventManager()->bind('onCredentialsBad', function ($number) {
78                         });
79             $w->eventManager()->bind('onLoginFailed', function ($number) {
80             });
81
82                         $code = $w->codeRequest('sms');
83                         ob_end_clean();
84
85                         if ($code->status == 'ok' && !empty($code->pw)) {
86                                 $item->setUsername($code->login);
87                                 $item->setPassword($code->pw);
88                                 $dao->save($item);
89                         }
90                 } catch (\Exception $ex) {
91                         $res[] = ob_get_contents();
92                         ob_end_clean();
93                 }
94
95                 chdir($oldDir);
96
97                 return $res;
98         }
```

10.21.2.9 GitGis\Whatsapp\Model\WhatsappDAO::setDebug (*$debug*)

Set debugging mode.

Parameters

boolean	*$debug*	

Definition at line 39 of file WhatsappDAO.php.

```
39                                          {
40                 $this->debug = $debug;
41         }
```

10.21.2.10 GitGis\Whatsapp\Model\WhatsappDAO::syncContacts (*$groupId*, *$userId*)

Sync group contacts.

Definition at line 419 of file WhatsappDAO.php.

```
419                                                     {
420         $groupDAO = new GroupDAO();
421         $senderDAO = new SenderDAO();
422         $sender = null;
423         $senderList = $senderDAO->getList(array('user_id' => $userId));
424         if (!empty($senderList['list'])) {
425             foreach ($senderList['list'] as $itemList) {
426                 $sender = $itemList;
427                 break;
428             }
429         }
430 
431         if (empty($sender)) {
432             return;
433         }
434 
435                 $contacts = array_keys($groupDAO->getNumbers($groupId));
436 
437                 $numbers = array();
438                 try {
439                         $sync = new \WhatsAppContactSync($sender->getUsername(), $sender->getPassword(),
    $contacts, true);
440                         $results = $sync->executeSync();
441 
442                         foreach ($results as $result) {
443                                 if (!empty($result['phonenumber'])) {
444                                         $numbers[] = $result['phonenumber'];
445                                 }
446                         }
447                         $groupDAO->markSynced($numbers);
448                 } catch (\Exception $ex) {
449                 }
450         }
```

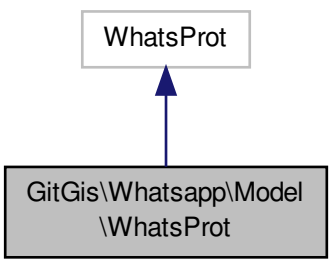
The documentation for this class was generated from the following file:

- src/GitGis/Whatsapp/Model/WhatsappDAO.php

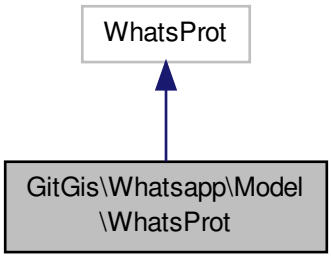
10.22 GitGis\Whatsapp\Model\WhatsProt Class Reference

Overrides standard WhatsProt class.

Inheritance diagram for GitGis\Whatsapp\Model\WhatsProt:



Collaboration diagram for GitGis\Whatsapp\Model\WhatsProt:



Public Member Functions

- getSentNodes ()
- sendMessageVideo ($to, $filepath, $storeURLmedia=false)

Protected Member Functions

- sendNode ($node)

Protected Attributes

- $sentNodes = array()

10.22.1 Detailed Description

Overrides standard WhatsProt class.

Its purpose is to store debug info

Definition at line 14 of file WhatsProt.php.

10.22.2 Member Function Documentation

10.22.2.1 GitGis\Whatsapp\Model\WhatsProt::getSentNodes ( )

Definition at line 23 of file WhatsProt.php.

```
23                                      {
24                 return $this->sentNodes;
25         }
```

10.22.2.2 GitGis\Whatsapp\Model\WhatsProt::sendMessageVideo (*$to, $filepath, $storeURLmedia* = `false`)

Definition at line 32 of file WhatsProt.php.

```
33     {
34         $allowedExtensions = array('mp4', 'mov', 'avi', '3gp');
35         $size = 20 * 1024 * 1024; // Easy way to set maximum file size for this media type.
36         return $this->sendCheckAndSendMedia($filepath, $size, $to, 'video', $allowedExtensions,
       $storeURLmedia);
37     }
```

10.22.2.3 GitGis\Whatsapp\Model\WhatsProt::sendNode (*$node*) `[protected]`

Definition at line 27 of file WhatsProt.php.

```
27                                          {
28                 parent::sendNode($node);
29                 $this->sentNodes[] = $node;
30         }
```

10.22.3 Member Data Documentation

10.22.3.1 GitGis\Whatsapp\Model\WhatsProt::$sentNodes = array() `[protected]`

Definition at line 21 of file WhatsProt.php.

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

- src/GitGis/Whatsapp/Model/WhatsProt.php

11 File Documentation

11.1 docs/dev/010_php.md File Reference

11.2 docs/dev/020_db.md File Reference

11.3 docs/dev/080_protocol.md File Reference

11.4 docs/dev/090_update.md File Reference

11.5 src/GitGis/Auth/AuthController.php File Reference

Classes

- class GitGis\Auth\AuthController

  Authetication controller.

Namespaces

- GitGis\Auth

11.6 src/GitGis/Auth/GitGisMiddleware.php File Reference

Classes

- class GitGis\Auth\GitGisMiddleware

Namespaces

- GitGis\Auth

11.7 src/GitGis/Pager.php File Reference

Classes

- class GitGis\Pager
  Pagination class.

Namespaces

- GitGis

11.8 src/GitGis/Whatsapp/GroupsController.php File Reference

Classes

- class GitGis\Whatsapp\GroupsController
  Groups controller.

Namespaces

- GitGis\Whatsapp

11.9 src/GitGis/Whatsapp/InboxController.php File Reference

Classes

- class GitGis\Whatsapp\InboxController
  Message controller.

Namespaces

- GitGis\Whatsapp

11.10 src/GitGis/Whatsapp/MainController.php File Reference

Classes

- class GitGis\Whatsapp\MainController
  Main controller.

Namespaces

- GitGis\Whatsapp

11.11 src/GitGis/Whatsapp/MessagesController.php File Reference

Classes

- class GitGis\Whatsapp\MessagesController

  Message controller.

Namespaces

- GitGis\Whatsapp

11.12 src/GitGis/Whatsapp/Model/Chat.php File Reference

Classes

- class GitGis\Whatsapp\Model\Chat

  Inbox message.

Namespaces

- GitGis\Whatsapp\Model

11.13 src/GitGis/Whatsapp/Model/ChatDAO.php File Reference

Classes

- class GitGis\Whatsapp\Model\ChatDAO

  Inbox management class.

Namespaces

- GitGis\Whatsapp\Model

11.14 src/GitGis/Whatsapp/Model/DBConnection.php File Reference

Classes

- class GitGis\Whatsapp\Model\DBConnection

  Overrided default PDO class with config taken from config.php.

Namespaces

- GitGis\Whatsapp\Model

11.15 src/GitGis/Whatsapp/Model/Group.php File Reference

Classes

- class GitGis\Whatsapp\Model\Group

  Group entity.

Namespaces

- GitGis\Whatsapp\Model

11.16 src/GitGis/Whatsapp/Model/GroupDAO.php File Reference

Classes

- class GitGis\Whatsapp\Model\GroupDAO

  Group management class.

Namespaces

- GitGis\Whatsapp\Model

11.17 src/GitGis/Whatsapp/Model/Message.php File Reference

Classes

- class GitGis\Whatsapp\Model\Message

  Message entity.

Namespaces

- GitGis\Whatsapp\Model

11.18 src/GitGis/Whatsapp/Model/MessageDAO.php File Reference

Classes

- class GitGis\Whatsapp\Model\MessageDAO

  Message management class.

Namespaces

- GitGis\Whatsapp\Model

11.19 src/GitGis/Whatsapp/Model/Sender.php File Reference

Classes

- class GitGis\Whatsapp\Model\Sender

  Sender entity.

Namespaces

- GitGis\Whatsapp\Model

11.20 src/GitGis/Whatsapp/Model/SenderDAO.php File Reference

Classes

- class GitGis\Whatsapp\Model\SenderDAO

  Sender management class.

Namespaces

- GitGis\Whatsapp\Model

11.21 src/GitGis/Whatsapp/Model/User.php File Reference

Classes

- class GitGis\Whatsapp\Model\User

  User entity used during authentication.

Namespaces

- GitGis\Whatsapp\Model

11.22 src/GitGis/Whatsapp/Model/UserDAO.php File Reference

Classes

- class GitGis\Whatsapp\Model\UserDAO

  Authentication/user management class.

Namespaces

- GitGis\Whatsapp\Model

11.23 src/GitGis/Whatsapp/Model/WhatsappDAO.php File Reference

Classes

- class GitGis\Whatsapp\Model\WhatsappDAO
  DAO responsible for communicating WhatsApp server.

Namespaces

- GitGis\Whatsapp\Model

11.24 src/GitGis/Whatsapp/Model/WhatsProt.php File Reference

Classes

- class GitGis\Whatsapp\Model\WhatsProt
  Overrides standard WhatsProt class.

Namespaces

- GitGis\Whatsapp\Model

11.25 src/GitGis/Whatsapp/SendersController.php File Reference

Classes

- class GitGis\Whatsapp\SendersController
  Senders controller.

Namespaces

- GitGis\Whatsapp

11.26 src/GitGis/Whatsapp/UsersController.php File Reference

Classes

- class GitGis\Whatsapp\UsersController
  User management controller.

Namespaces

- GitGis\Whatsapp

Index