Guide to use for Developers

Message Module V. 0.05.06

Moisés Alcocer, 2017

https://www.ironwoods.es

Index of contents

1. Installation

For using the module first time you must follow this ***Step by step installation***

1) Requires the main class: "messages.class.php", or adds it to your autoload as: "ironwoods\modules\messages\Messages".

2) Instances the main class:

```
$msgs = new ironwoods\modules\messages\Messages(
        $your_db_connection
);
```

3) Creates the settings file and the tables in the Database:

```
$arr_with_data = [ ... ]; //see below
$msgs->install( $arr_with_data );
```

You need some information about the table referred in your system, for composing the $arr_with_data. For example, if you have the table "xi_users": the *users* are who send messages to each other and "xi_" is the prefix for your tables.

```
$arr_with_data = [
        "table-prefix"      => "xi_",
        "table"             => "users",
        //Next are from your table ID:
        "id"                => "id",
        "id-type"           => "int",
        "id-length"         => 12,
        "id-unsigned"       => TRUE,
];
```

Now, you can check if the installation is ok: search for the new table in the DB: "xi_users" and for the *settings* file in the module configuration directory: "message-mod/config/files/ ". If both is OK the installation was successful!

2. Working

2.1. Instances main class

For using the module, after the installation, first of all, it is necessary to instance the main class:

```
$msgs = new ironwoods\modules\messages\Messages(
    $your_db_connection
);
```

2.2. Use the module logical functions

2.2.1. Available methods

There are several functional methods that you can use. It's necessary to understand that in the table "messages" messages and drafts are stored together, the difference between they are that the drafts don't have a receiver ID. From the logic point of view, or access to a row content, a draft owner is the user that wrote it (this row doesn't have a receiver ID). For a message the owner is the user that received it. **Sended messages aren't handled in this version**.

Methods to *get a number of messages* for the user:

- **countMsgs()** -> Counts all the received messages
- **countNewMsgs()** -> Counts the unread messages

Other methods to *handle messages*:

- **deleteOne()** -> Deletes a received message
- **getAll()** -> Gets all messages of the receiver
- **getOne()** -> Gets the message with the identifier
- **getUnread()** -> Gets all unread messages of the receiver
- **send()** -> Sets receiver ID in a message
- **setRead()** -> Sets the received message as read

Methods to *handle drafts*:

- **addOneDraft()** -> Inserts a new draft in the DB
- **changeDraftContent()** -> Updates the content in a draft
- **changeDraftSubject()** -> Updates the subject in a draft
- **deleteOneDraft()** -> Deletes a draft
- **getAllDrafts()** -> Gets all drafts for the author
- **getOneDraft()** -> Gets the draft with the identifier

2.2.2. Available methods and basic use

2.2.2.1. Count messages

1. Count the user's messages: received (all) and unread (new)

countNewMsgs() / countMsgs() need the "referer ID" as argument, f.e. the "user_id".

```
//User with ID: 5

//Number of received messages
$n_msgs = $msgs->countMsgs( 5 );

//Number of received unread messages
$n_msgs = $msgs->countNewMsgs( 5 );
```

2.2.2.2. Handle messages

1. Deleting messages (by receiver)

Method **deleteOne()** is used to delete a message from the DB. It accepts a "message ID" and "the receiver ID". A boolean result will be returned.

```
//Deletes message with ID 5, received by user 3
$res = $msgs->deleteOne( 5, 3 );

//Possible way to handle operation results:
$err = "Message hasn't been deleting";
$ok  = "Message has been deleting";
$success = ( $res )
    ? [ "status" => "ok",  "feedback" => $ok ]
    : [ "status" => "err", "feedback" => $err ];
```

2. Getting the received messages

Method **getAll()** is used to get all messages for a user (received) passing the "user ID" as argument.

```
//Gets all the messages for user with ID 25
$arr_msgs = $msgs->getAll( 25 );
```

3. Reading a message

Method **getOne()** is used to get a message from the Db, the passing the "message ID" and the "receiver ID"as argument.

```
//Gets the message with ID 1 received by user 2
$msg = $msgs->getOne( 1, 2 );
```

4. Getting the received unread messages

Method **getUnread()** gets all unread messages for a user (received). We pass the "user ID" as argument.

```
//Gets unread messages for user with ID 25
$arr_unread_msgs = $msgs->getUnread( 25 );
```

5. Sending messages

To send a message from the user with ID: 4 to the user with ID: 5 the method **send()** is used. Then are two arguments necessary: the "message (draft) ID" and the "destinatary ID". Then, there are two possible ways: you have a draft and send it or you write and send at the same time, ergo, you create a draft and then send it.

```
/* Creates a draft (See forward apdo. 2.2.2.3.
 * Handle drafts / Create news)
 * ...
 **/
$res_id = $msgs->addOneDraft( $msg );
//or: $res_id = $msgs->createDraft( $msg );

//Sends the message to user with ID: 5
$res = $msgs->send( $res_id, 5 );

//Possible way to handle operation results:
$err = "Message hasn't been sending";
$ok  = "Message has been sending";
$success = ( $res )
     ? [ "status" => "ok",  "feedback" => $ok ]
     : [ "status" => "err", "feedback" => $err ];
```

6. Marking a message as read

The method **setRead()** is used to set a message as read passing the "message ID" in firts place and the "user ID" from receiver as second argument.

```
//Sets message ID: 25 read by user with ID: 3
$res = $msgs-> setRead( 25, 3 );
```

2.2.2.3. Handle drafts

1. Creating news

addOneDraft() needs an instance of ***Message*** as argument. A Draft or *not sent message* will be created. Alternatively, you can use **createDraft()** in the same mode that the first one.

```
//Creating a draft for the user with ID: 4

//Way one (passing the data with chain methods):
$msg = new Message();
$msg->setSenderId( 4 )
     ->setSubject( "fake" )
     ->setContent( "lorem ipsum content" );

//or way two (passing the data in constructor):
$msg = new Message(
     4,
     "fake",
     "lorem ipsum content"
);

//Storing the draft into DB
$res_id = $msgs->addOneDraft( $msg );
//or: $res_id = $msgs->createDraft( $msg );
```

2. Changing the content in a draft

It's possible to change a draft content. You can be to change the subject and the content, using the methods: **changeDraftContent()** and **changeDraftSubject(),** that both accept, in this order, a "message (draft) ID", the "writer ID" and "a string with the new". Both operations return a boolean result.

```
//Changes draft with ID 1 of the user with ID 1
$res = $msgs->changeDraftSubject(
     1,
     "test-xxx"
);

//Possible way to handle operation results:
//...
```

```
//Changes draft with ID 1 of the user with ID 1
$res = $msgs->changeDraftContent(
     1,
     "test-zzz"
);

//Possible way to handle operation results:
//...
```

3. Deleting a draft (by author)

deleteOneDraft() is the method used to delete a draft from the DB. It accept a "message ID" and "the author ID". A boolean result will be returned.

```
//Deletes draft with ID 3, wrote by user 12
$res = $msgs->deleteOneDraft( 3, 12 );

//Possible way to handle operation results:
//...
```

4. Getting drafts from the user

Method **getAllDrafts()** gets the drafts belonging to a user, using the user's ID as argument.

```
//Gets drafts from user with ID 3
$drafts = $msgs-> getAllDrafts( 3 );
```

5. Getting a draft

getOneDraft() is used to get a draft from the Db, passing the "draft ID" and the "writer ID" as arguments.

```
//Gets the draft with ID 1 of the user with ID 1
$msg = $msgs->getOneDraft( 1, 1 );
```

2.2.3. Facade class: basic structure

```
<?php namespace ironwoods\modules\messages;
/**
 * Class and module descriptions
 *
 */

//requires

//Use of classes

class Messages {

/**********************************/
/*** Properties declaration *******/

     private $con = NULL;

/**********************************/
/*** Methods declaration **********/

     /**
      * Construct
      * To create an instance must be to inyect
      * a connection with the DB
      *
      * @param  Object      $db_con
      */
     public function __construct( $db_con ) {

          $this->con = $db_con;
     }
     /*** Public Methods ***************/

     ////////////////////////////////////////////////
     // Install the module
     //
     // The installation require some parameters about
     // your DB as:
     //   - DB tables Prefix (if exist), f.e. "appx_"
     //   - Name of reference table (without prefix).
     // This is those with the subjects that message
     // ones to others: "users", "owners",
"clients"...
     //   - Ref. Table ID, default "id"
     //   - Ref. Table ID type, f.e. "Int"
     //   - Ref. Table ID length, f.e. "12"
     //

          /**
           * Install the module in the App
           *
           * In the module installation is required
           * data for the table with entities that
           * send and receive the messages:
           * array [
           *       "table-prefix"    => "xxx",
           *       "table"       => "xxx",
           *       "id"          => "xxx"
           *       "id-type"         => "xxx",
           *       "id-length"   => 10,
           * ]
           * This data is used to create settings file
           * with an array
           *
           * @param     Array       $arr_data
           */
          public function install( $arr_data ) {}


     ////////////////////////////////////////////////
     // Uninstall the module
     //
     // This deletes the table with the messages from
     // This the DB and the settings file from the HD
     //
     /**
      * Unistalls the module from the App
      *
      */
     public function uninstall() {}


////////////////////////////////////////////////
// Methods in class "Count"
//
//
     /**
      * Counts all of messages received
      *
      * @param      int          $receiver_id
      * @return     int
      */
     public function countMsgs( $receiver_id ) {}

     /**
      * Counts unread messages
      *
      * @param      int          $receiver_id
      * @return     int
      */
     public function countNewMsgs( $receiver_id )
{}

////////////////////////////////////////////////
// Methods in class "XDraft"
//
//
     /**
      * Inserts a new message (draft) in the DB
      *
      * @param     Message       $x
      * @return    int           id
      */
     public function addOneDraft( Message $x ) {}

     /*** Alternative method for the same ***/
     public function createDraft( Message $x ) {}
```

```
        /**
         * Changes the content for a draft
         *
         * @param      int          $id
         * @param      string       $str
         * @return     boolean
         */
        public function changeDraftContent(
    $id, $str ) {}

        /**
         * Changes the subject for a draft
         *
         * @param      int          $id
         * @param      string       $str
         * @return     boolean
         */
        public function changeDraftSubject(
    $id, $str ) {}

        /**
         * Deletes a draft by the author
         *
         * @param     int            $id
         * @param     int            $writer_id
         * @return    boolean
         */
        public function deleteOneDraft(
    $id, $writer_id ) {}

        /**
         * Gets the drafts for the user with the ID
         *
         * @param     int            $writer_id
         * @return    array
         */
        public function getAllDrafts( $writer_id )
{}

        /**
         * Gets the draft with the ID of the author
         *
         * @param   int              $id
         * @param   int              $writer_id
         * @return    array
```

```
       */


      public function getOneDraft(
 $id, $writer_id ) {}


 ///////////////////////////////////////////////////
 // Methods in class "XMessages"
 //
 //
      /**
       * Deletes a message by receiver
       *
       * @param     int            $id
       * @param     int            $receiver_id
       * @return  boolean
       */
      public function deleteOne(
 $id, $receiver_id ) {}

      /**
       * Gets the messages for the user with ID
       *
       * @param     int            $receiver_id
       * @return    array
       */
      public function getAll( $receiver_id ) {}

      /**
       * Gets the message for the user with ID
       *
       * @param     int            $id
       * @param     int            $receiver_id
       * @return    object
       */
      public function getOne( $id, $receiver_id )
 {}

      /**
       * Gets the unread messages of the receiver
       *
       * @param     id             $user_id
       * @return    array
       */
      public function getUnread( $user_id ) {}
```

```
        /**
         * Sets the message sended ( put the ID for
         * the receiver )
         *
         * @param      int          $id
         * @param      int          $destinatary_id
         * @return     boolean
         */
        public function send( $id, $destinatary_id )
    {}

        /**
         * Sets the field read to 1 in the table
         *
         * @param      int          $id
         * @param      int          $receiver_id
         * @return     boolean
         */
        public function setRead(
    $id, $receiver_id ) {}


    /*** Private Methods **************/

    } //class
```