PHP OTServ Toolkit

Contents

POT

This is documenation of POT - official toolkit for OTServ AAC scripts.

PHP OTServ Toolkit

There are several reasons why POT was created:

- Just because it was needed - OTServ should have had that long time ago.
- To unify AAC scripts - there are tons of them, and you never know how to write even a single line of code to them as each of them are created different way.
- To provide reliable way of database accessing - most of people who create AAC scripts are (to be honest...) idiots - they don't know what PHP is, how to use it, they just "want to make own AAC script".
- To provide easy interface - people who write in PHP want to write in PHP, not using SQL, XML and many other languages. POT provides abstract PHP interface for data stored in database.

POT has been created for latest SVN release, it will work best with pure SVN servers. However it provides routines to access custom database structure elements. However it won't work with broken database - it ralies on database foreign key contraints, triggers etc.

System requirements

To use POT you need PHP version at least 5.0 with PDO extension installed (so it means you will mostly need PHP 5.1, but it is possible to download PDO as external libraries for PHP 5.0.x).

What POT is

POT is a toolkit/library for accessing OTServ database from PHP. It provides PHP classes that represents OTServ database inforation as an objects.

What POT is not

- It is not AAC script - this is a toolkit for making them, but you can't directly run it as website. It has only programming interface.
- It is not application/system framework - you won't create website with only POT. POT has only functionality connected with OTServ database, it doesn't contain for example templates engine. You also won't be able to use it as an ordinary database connection engine - it makes use of PDO so you can use PDO by itself, POT doesnt provide any additional universal functionality. All it's classes are strictly connected with OTServ database.

What about XML?

Sorry to say, XML guys - go out. OTServ will never leave XML - it is good to store some flat parts of database there. But not for main database which requires more advanced relationship between data. However of course maybe someone would want to create DB_XML driver for POT? If you realy are a masochist - you're welcome, we will be glad to contribute with you ;).

If you are interested in why XML so sux, and you with it, check out OTFans thread.

How to use

This is toolkit - set of classes/methods for OTServ database. It abstracts database mechanisms for you so you can work on "physical" PHP objects. But you must know how to use them. This documentation describes some basic steps and toolkit API, but you must know PHP in order to make use of them - the best place to get some knowledge is PHP manual.

Don't copy any of included examples, neither codes provided as examples - they probably won't work. Mainly it's because you have to put your database configuration into them and your script paths. But it's not enought. If you have your own __autoload() mechanism you won't be able to just inlude example codes - you would need to redefine __autoload() function, which PHP doesnt allow to (but you should know that very well). Example codes are examples - write your own (if you want them to work the best way for you).

Link

If you use POT in your script and want to show that you can put this image on your website:

You can use following code for that:

```
1  <a  href="http://www.otserv-aac.info/pot/"  >
2    <img  alt="This site was smoked"  src="http://www.otserv-aac.info/pot.png"  />
3  </a>
```

PHP 5.0

Some things that you should know if you use POT under PHP 5.0.x.

PHP 5.0

PHP5 was a huge step in PHP histroy. It is completly other language then PHP4 (and older versions). POT is written for PHP5 but currently most PHP5 installations are done with PHP 5.1 and higher versions. PHP 5.0 differs from next versions in few details (or even not details, but huge changes, but those mostly doesn't affect POT). There are some important things you should know if you use POT with PHP 5.0.

PDO

POT requires PDO extension. It is bundled with PHP since 5.1 version. If you use PHP 5.0 you still can install PDO, but you need to do that using PECL extensions. Detailed information about how to do that are in PHP manual PDO page.

Sub package "compat"

If you use PHP 5.0 you should include special compatibility assurance library. POT uses some mechanisms that exists since PHP 5.1 like Countable interface. It doesn't disallow you using POT with PHP 5.0. Compatibility library will create unexisting interfaces, classes, functions, constants etc. However keep in mind that you won't be able to use PHP 5.1 and newer language mechanisms as it is not possible to redefine PHP behaviour. Here is an example:

```
1   <?php
2
3   /**
4    * @ignore
5    * @package examples
6    * @author Wrzasq <wrzasq@gmail.com>
7    * @copyright 2007 (C) by Wrzasq
8    * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9    */
10
11  // do that before any POT operations!
12  include( '../compat.php');
13
14  // to not repeat all that stuff
15  include( 'quickstart.php');
16
17  // STEP 1: no error here - even thought we loaded class that implements Countable interface which does not
exists in PHP 5.0 SPL library, because 'compat' library defines it.
18  $list= POT::getInstance()->    createObject('Players_List');
19
20  // STEP 2: we can do that in every version - count() is in fact just a public method
21  echo $list->    count();
22
23  // STEP 3: it won't work correctly in PHP 5.0 - PHP won't call internaly count() method of object, will print trivial
count() evaluation result on object
24  echo count( $list);
```

```
25
26  ?>
```

Nothin new

Compatibility library makes you sure, that POT scripts won't cause FATAL errors if you run them on older versions of PHP. However it doesn't introduce any new mechanisms so you won't find anything new in this package. It is safe to include compat.php file even if you work with PHP version 5.1 or newer, but there is no point in doing that.

__autoload()

POT registers own __autoload() handler with spl_autoload_register(). This function exists since PHP 5.1.2. Compatibility library defines this function as definer of another function - ordinary __autoload(). If you have own __autoload() function, compat's spl_autoload_register() won't redefine __autoload() to avoid E_ERROR. You then need to bind POT::loadClass() method to your __autoload() function manualy.

Type hinting

In PHP5 new feature was introduced - type hinting. In PHP 5.0 only class names are supported, array type is supported since PHP 5.1. So if you want to use POT with PHP 5.0 you must remove all array hinted types in classes methods.

What about older PHP versions?

No way. POT was written using new PHP5 object engine - you cant use it with PHP4 and older versions of PHP, PHP/FI.

POT class preview

Here main POT class will be described in more guided way.

What it is

POT class is main class of this toolkit. You will access any other classes using this one. It creates for you instances of other classes when you call it's methods and handles class files loading.

Creating instance of POT class

To get POT object you have to use POT::getInstance() static method. You should never ever create POT class instances directly! POT::getInstance() will save static instance and return it globaly so you won't need to re-create instances of this class. It is important, as object of this class contains another resources like database connection, or classes directory path so after creating new instance it would not contain them from previous one.

__autoload() and POT classes

PHP5 provides nice autoloading mechanism. POT makes use of spl_autoload_register() function to bind own mechanism with it automaticly. If you have your own __autoload function defined, after including POT class you have to register your function with spl_autoload_register() aswell.

DAO classes

Key part of this toolbox are Data Access Objects which provides abstraction layer in PHP for plain database data. You create them via main POT class using createObject() method.

Quick start

Quick start guide.

Putting this all together

To set POT up for using you have to create it's instance and connect to database (it will automaticly bind POT classes loading mechanism to __autoload() function. Here is a startup code example:

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
 */

// binds your __autoload code
if( function_exists('__autoload') )
{
    spl_autoload_register('__autoload');
}

// includes POT main file
include( '../classes/OTS.php');

// database configuration - can be simply moved to external file, eg. config.php
$config= array(
    'driver' =>     POT::DB_MYSQL,
    'host' =>     'localhost',
    'user' =>     'wrzasq',
    'database' =>     'otserv'
);

// creates POT instance (or get existing one)
$ots = POT::getInstance();
$ots->     connect(null, $config);

?>
```

Account creation

It is very simple to create account with POT. Here is example code that is self-explainable:

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
```

```
9    */
10
11   // to not repeat all that stuff
12   include( 'quickstart.php');
13
14   // creates new OTS_Account object
15   $account= $ots->      createObject('Account');
16
17   // generates new account number
18   $number= $account->      create();
19
20   /*
21   to generate number from 111111 to 999999 use:
22   $number = $account->create(111111, 999999);
23   */
24
25   // sets account info
26   $account->      setPassword('secret');// $account->setPassword( md5('secret') );
27   $account->      setEMail('foo@example.com');
28   $account->      unblock();// remember to unblock!
29   $account->      setPACCDays(0);
30   $account->      save();
31
32   // give user his number
33   echo 'Your account number is: ',     $number;
34
35   ?>
```

It is important to remember that create() method sets `blocked` field of record to true by default, so for smaller projects where you, for example, wouldn't need e-mail activation unblock it after creation.

Character reading

Here comes also simple example for character search:

```
1    <?php
2
3    /**
4     * @ignore
5     * @package examples
6     * @author Wrzasq <wrzasq@gmail.com>
7     * @copyright 2007 (C) by Wrzasq
8     * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9     */
10
11   // to not repeat all that stuff
12   include( 'quickstart.php');
13
14   // creates new OTS_Player object
15   $player= $ots->      createObject('Player');
16
17   // loads player
18   $player->      find('Wrzasq');
19
20   // checks if player exists
21   if( $player->      isLoaded() )
22   {
23       // prints character info
```

```
24      echo 'Player \'' . $player->getName() . '\' has ' . $player->getLevel() . ' level.', "\n";
25
26      // example of associated objects retriving
27      echo 'Player \'' . $player->getName() . '\' is member of ' . $player->getGroup()->getName() . '
group.', "\n";
28  }
29  else
30  {
31      echo 'Player does not exists.', "\n";
32  }
33
34  ?>
```

Objects listings

There are also classes for entire sets of records. For each of row classes there is list class. Throught list object you can read single objects and/or delete them from database. Also you can set limitation (for example for pagination). All list classes implements Countable and Iterator interfaces:

```
1   <?php
2
3   /**
4    * @ignore
5    * @package examples
6    * @author Wrzasq <wrzasq@gmail.com>
7    * @copyright 2007 (C) by Wrzasq
8    * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9    */
10
11  // to not repeat all that stuff
12  include('quickstart.php');
13
14  // creates new OTS_Player object
15  $players = $ots->createObject('Players_List');
16
17  // count of all players - Countable interface implemented
18  echo 'There are ' . count($players) . ' players in our database.', "\n";
19
20  // sets limitation
21  $players->setLimit(10);
22  $players->setOffset(2);
23
24  // iterates throught selected players
25  foreach($players as $index => $player)
26  {
27      // each returned item is instance of OTS_Player class
28      echo (2 + $index) . ': ' . $player->getName(), "\n";
29  }
30
31  ?>
```

DAO objects

Main part of POT are Data Access Objects objects

What are DAO objects?

DAO stands for Data Access Objects. Those are objects which you use mostly - players, accounts, groups, objects lists. They use database resource to fetch/store data and provides you programming interface to access that data without using additional langauges like SQL, or XML.

Why this way?

PHP is a PHP. When you write a code in PHP each element has a meaning. While using SQL you have to use database queries. In code they are simply a strings which doesn't represent any particular data for programming enviroment. DAO objects wraps database operations in objective aspect, so "dead" string queries becomes a fully functional objects which you can control more strictly, allows you to assign relations and automate some parts.

Basic operations

Most basic operations are loading, editing and saving data. To see examples of this, see Quick start quide.

Lists objects

For each table there exist single object class and objects list class. List classes implements Iterator interface so to list their's content you must use foreach() loop. Each element returned for this loop will be instance of single DAO object. You also use lists to delete items.

Custom fields

POT was created for basic SVN database structure. However you can access custom fields with POT. You do that with getCustomField() and setCustomField() methods of DAO objects (single, not lists).

While accessing custom fields you have to remember about using proper PHP types of passed values. POT doesn't know anything about those fields so it uses value type to check the way it should serve it for a query. Don't worry about safety - it doesn't create any hole for SQL injections. But you must remember, that 1 (integer) is not same as '1' (string), or 1.0 (float). POT will quote strings to fit SQL query and to prevent from SQL injections so make sure you cast your values to type that represents field type to prevent (mainly) from quoting numeric fields.

You should use those methods only to access custom fields that are not accessible throught standard POT API. Those methods executes SQL query each time you call them so it would be a huge effectivity loss to access standard fields with getCustomField()/setCustomField().

Also it is important that in difference to fields accessible with standard setters you can set custom field value

on not loaded/saved object. You must either load object from database, or save standard record before using custom fields as they need record primary key assigned to object for queries. Here is an example:

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
 */

// to not repeat all that stuff
include('quickstart.php');

// creates new OTS_Player object
$player = $ots->createObject('Player');

// sets basic fields
$player->setName('Wrzasq');
$player->setSex(POT::SEX_MALE);
$player->setVocation(POT::VOCATION_KNIGHT);
/* etc... */

/*
this is bad! we can't call this now as we dont have object ID assinged yet

$player->setCustomField('my_field', 2);

must save before that to get automatic ID:
*/
$player->save();

// now we can call that:
// 2 won't be quoted - it's integer
$player->setCustomField('my_field', 2);
// 3 will be quoted - '3' is a string!
$player->setCustomField('another_field', '3');

?>
```

Player items

POT provides also objective way of browsing/editing player items (body slots and depot items with all containers). You have OTS_Item and OTS_Container classes for that. OTS_Item represents single item, OTS_Container can contain sub-items (either OTS_Item objects, or next level OTS_Container objects).

There is important thing to mention - POT doesn't know anything about item types! Items tree only contains item IDs from database, it doesn't load any information from items.otb, nor items.xml files.

Detailed API you will find in documentation of those classes. Here are examples of how you use slot and depot items fetching and saving:

```php
<?php

/**
 * @ignore
```

```
5   * @package examples
6   * @author Wrzasq <wrzasq@gmail.com>
7   * @copyright 2007 (C) by Wrzasq
8   * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9   */
10
11  // to not repeat all that stuff
12  include( 'quickstart.php');
13
14  // creates new OTS_Player object
15  $player = $ots->createObject('Player');
16  $player->find('Wrzasq');
17
18  /*
19      Items loading example.
20  */
21
22  // loading item from ammunition slot
23  $item = $player->getSlot(POT::SLOT_AMMO);
24
25  echo $player->getName(), ' has item with id ', $item->getId(), ' in his/her ammo slot.', "\n";
26
27  // checks if item is a container
28  if($item instanceof OTS_Container)
29  {
30      // list backpack content
31      foreach($item as $inside)
32      {
33          echo 'Container contains item with id ', $inside->getId(), '.', "\n";
34      }
35  }
36
37  /*
38      Items tree composing example.
39  */
40
41  // creates container - here it would be a depot locker (we pass ID of item to create)
42  $container = new OTS_Container(2590);
43
44  // now let's create depot chest
45  $chest = new OTS_Container(2594);
46
47  // let's put chest inside locker
48  $container->addItem($chest);
49
50  // now let's put something deeper - into the chest
51  $item1 = new OTS_Item(3015);
52  $chest->addItem($item1);
53
54  // and more...
55  $item2 = new OTS_Item(3013);
56  $chest->addItem($item2);
57
58  // let's set count for an item
59  $item2->setCount(2);
60
61  /*
62  Here is a tree of items which we created:
63
```

```
64  $container [depot locker]
65  `-- $chest [depot chest]
66      |-- $item1 [first item inserted into chest]
67      `-- $item2 [second item inserted into chest] count=2
68  */
69
70  /*
71      Items saving example.
72  */
73
74  // now we simply put those items into players depot (2 is depot ID)
75  $player->      setDepot(2,$container);
76
77  ?>
```

Important thing - OTS_Container class is subclass of OTS_Item. Each container is also an item.

Guilds

Guilds system basics.

Baiscs

Like for most other data types, for guilds and ranks there are two kinds of classes - single object class and list class. For guilds those are OTS_Guild and OTS_Guilds_List, for ranks - OTS_GuildRank and OTS_GuildRanks_List.

Guild management

Listing guilds is simple so there is no need to explain it more. More complex is listing guild members. Guild membership is not assigned directly - it is done throught guild ranks. To list guild members you first need to list it's ranks. Here is an example solution to list members in oryginal Tibia-like way:

```
1   <?php
2
3   /**
4    * @ignore
5    * @package examples
6    * @author Wrzasq <wrzasq@gmail.com>
7    * @copyright 2007 (C) by Wrzasq
8    * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9    */
10
11  // to not repeat all that stuff
12  include( 'quickstart.php');
13
14  // loads guild
15  $guild= $ots->      createObject('Guild');
16  $guild->      load(1);
17
18  $color=  '#FFFFCC';
19
20  echo '<h1>Members of '       , htmlspecialchars(       $guild->getName() ), '</h1>'       ;
21
22  ?>
23  <table>
24      <thead>
25          <tr>
26              <th>Rank</th>
27              <th>Members</th>
28          </tr>
29      </thead>
30      <tbody>
31  <?php
32
33  // lists members of all ranks
34  foreach( $guild->      getGuildRanks()as $guildRank)
35  {
36      // display rank in first row
37      $first=  true;
```

```
38      // switches rank rows color
39      $color= $color==   '#FFFFCC' ? '#FFCCFF' : '#FFFFCC';
40
41      // list members of this rank
42      foreach( $guildRank->       getPlayers() as $player)
43      {
44          echo '<tr style="background-color: '           ,       $color;
45      <td>'       ,   $first?htmlspecialchars(    $guildRank->   getName() ) : '', '</td>
46      <td>'       ,   $player->    getName(), '</td>
47  </tr>'      ;
48          $first=  false;
49      }
50  }
51
52  ?>
53      </tbody>
54  </table>
```

Guild action drivers

Handling invites/requests system for guilds.

How does it work?

OTServ database contains all guilds contents. But it is very common in AAC world to create invites system (or also requests system, but invitations are more common). It is not provided by standard OTServ database, thought nearly all AAC scripts contains such mechanisms. POT classes allows you to set own drivers for invitations and requests to extend basic OTS functionality.

You have to write a driver class and assign it's object to guild object - then guild object will call requested actions on driver which will execute action code dependent on your script.

Driver structure

Both invites and requests drivers are similar - they must implement IOTS_GuildAction interface. When the driver is assigned to guild object, each time a method of OTS_Guild object is called, it will forward this to action driver.

Sample driver

Driver implements your logic for invites (or membership requests). Here is sample code that you can base on:

```
1   <?php
2
3   /**
4    * @ignore
5    * @package examples
6    * @author Wrzasq <wrzasq@gmail.com>
7    * @copyright 2007 (C) by Wrzasq
8    * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9    */
10
11  // to not repeat all that stuff
12  include( 'quickstart.php');
13
14  /*
15      POT guilds invites driver.
16  */
17
18  /**
19   * @ignore
20   */
21  class InvitesDriver implements IOTS_GuildAction
22  {
23      // assigned guild
24      private  $guild;
25
26      // initializes driver
27      public  function __construct(OTS_Guild  $guild)
28      {
```

```
29          $this->guild = $guild;
30          // this line automates the process - you can call it manualy from outside, but why?
31          $this->guild->setInvitesDriver($this);
32      }
33
34      // returns all invited players to current guild
35      public function listRequests()
36      {
37          $invites = array();
38
39          /* here you must create OTS_Player object for each invited player */
40
41          return $invites;
42      }
43
44      // invites player to current guild
45      public function addRequest(OTS_Player $player)
46      {
47          /* here you must save invitation for given player */
48      }
49
50      // un-invites player
51      public function deleteRequest(OTS_Player $player)
52      {
53          /* here you must delete invitation for given player */
54      }
55
56      // commits invitation
57      public function submitRequest(OTS_Player $player)
58      {
59          $rank = null;
60
61          // finds normal member rank
62          foreach($this->guild->getGuildRanks() as $guildRank)
63          {
64              if($guildRank->getLevel() == 1)
65              {
66                  $rank = $guildRank;
67                  break;
68              }
69          }
70
71          $player->setRank($rank);
72          $player->save();
73
74          // clears invitation
75          $this->deleteRequest($player);
76      }
77  }
78
79  /*
80      Parts of this class driver has been taken from OTSCMS (http://otscms.sourceforge.net/) project source
code.
81  */
82
83  // loads player wiht ID 1
84  $player = $ots->createObject('Player');
85  $player->load(1);
86
```

```
87  // loads guild with ID 1
88  $guild = $ots->      createObject('Guild');
89  $guild->      load(1);
90
91  // creates invitation logic driver for your implementation for current guild
92  new InvitesDriver( $guild);
93
94  // note that you call guild method!
95  $guild->      invite($player);
96
97  ?>
```

Account number hack

Example code of how to use prepared account number instead of random.

Walkaround

POT always generates random account number - it is the way your script should work. It is done that way with premeditation. However you can walk aroud it with simple code:

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
 */

// to not repeat all that stuff
include( 'quickstart.php');

// your non-random number
$number = 123456;

// creates new OTS_Account object
$account = $ots->createObject('Account');
$account->load($number);

// number is busy
if( $account->isLoaded() )
{
    echo 'Account number ', $number, ' is used.', "\n";
}
// it is not
else
{
    // generate number from exacly $number - $number range
    $number = $account->create($number, $number);
    echo 'Your account number is: ', $number, "\n";
}

?>
```

Server online status

This tutorial will describe how to test server status with POT.

Such a simple way

POT class contains serverStatus() method which sends 'info' packet to OTS and handles results. It returns object of class OTS_InfoRespond which provides access methods for all OTServ respond info. It will return false if server is offline. Here is a simple example of this method usage:

```
1   <?php
2
3   /**
4    * @ignore
5    * @package examples
6    * @author Wrzasq <wrzasq@gmail.com>
7    * @copyright 2007 (C) by Wrzasq
8    * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9    */
10
11  // to not repeat all that stuff
12  include( 'quickstart.php');
13
14  // server and port
15  $server= '127.0.0.1';
16  $port= 7171;
17
18  // queries server of status info
19  $status= $ots->serverStatus($server, $port);
20
21  // offline
22  if(!$status)
23  {
24      echo 'Server ', $server, ' is offline.', "\n";
25  }
26  // displays various info
27  else
28  {
29      echo 'Server name: ', $status->getName(), "\n";
30      echo 'Server owner: ', $status->getOwner(), "\n";
31      echo 'Players online: ', $status->getOnlinePlayers(), "\n";
32      echo 'Maximum allowed number of players: ', $status->getMaxPlayers(), "\n";
33      echo 'Required client version: ', $status->getClientVersion(), "\n";
34      echo 'All monsters: ', $status->getMonstersCount(), "\n";
35      echo 'Server message: ', $status->getMOTD(), "\n";
36  }
37
38  ?>
```

DOM way

In case you would want to use this method for some non-SVN server which contains custom fields in respond packet you can still use it. OTS_InfoRespond class is child of DOMDocument class and doesn't overwrite it's

interface neither behaviour in any way. Returned object is standard DOM document so you can work with it in standard DOM-way.

Package POT Procedural Elements

E_OTS_NoDriver.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

E_OTS_NotLoaded.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.3
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.3
- **License** GNU Lesser General Public License, Version 3

IOTS_DAO.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

IOTS_DB.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

IOTS_GuildAction.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS.php

This file contains main toolkit class.

This file contains main toolkit class. Please read README file for quick startup guide and/or tutorials for more info.

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **TODO** 0.1.0: Get rid of POT::getInstance()->create*() calls - use POT::getInstance()->getDBHandle() in constructors.
- **TODO** 0.1.0: Items list (items.xml + items.otb -> cache).
- **TODO** 0.1.0: Implement __get()/__set()/__call()/__toString(); ArrayAccess interface.
- **TODO** 1.0.0: More detailed documentation.
- **TODO** 1.0.0: Complete phpUnit test.
- **TODO** 0.0.6: Spawns support (OTBM support -> cache).
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Account.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Accounts_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Base_DAO.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4+SVN
- **License** GNU Lesser General Public License, Version 3

OTS_Base_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4+SVN
- **License** GNU Lesser General Public License, Version 3

OTS_Container.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.3
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.3
- **License** GNU Lesser General Public License, Version 3

OTS_DB_MySQL.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_DB_ODBC.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS_DB_PostgreSQL.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS_DB_SQLite.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Group.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Groups_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Guild.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS_GuildRank.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS_GuildRanks_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS_Guilds_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4
- **License** GNU Lesser General Public License, Version 3

OTS_InfoRespond.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.2
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.2
- **License** GNU Lesser General Public License, Version 3

OTS_Item.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.3
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.3
- **License** GNU Lesser General Public License, Version 3

OTS_Player.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_Players_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

OTS_SQLField.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4+SVN
- **License** GNU Lesser General Public License, Version 3

OTS_SQLFilter.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.4+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.4+SVN
- **License** GNU Lesser General Public License, Version 3

OTS_SQLite_Results.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.1
- **License** GNU Lesser General Public License, Version 3

Package POT Classes

Class E_OTS_NoDriver

[line 20]

Occurs when code attempts to execute driven action that has no assigned driver to handle it.

Occurs when code attempts to execute driven action that has no assigned driver to handle it.

- **Package** POT
- **Version** 0.0.4
- **Since** 0.0.4

Class E_OTS_NotLoaded

[line 20]

Occurs when code attempts to access property of not loaded object.

Occurs when code attempts to access property of not loaded object.

- **Package** POT

- **Version** 0.0.3
- **Since** 0.0.3

Class IOTS_DAO

[line 22]

OTserv database object.

OTserv database object.
This insterface indicates that class is a OTServ DAO class.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.1

Constructor *void* function IOTS_DAO::__construct($db) *[line 30]*

Function Parameters:

- *PDO* **$db** Database connection object.

DAO objects must be initialized with a database.

DAO objects must be initialized with a database.

- **Version** 0.0.4+SVN
- **Deprecated** 0.0.4+SVN This constructor convention won't be part of interface in future.
- **Since** 0.0.1

- **Access** public

Class IOTS_DB

[line 25]

OTServ database handler interface.

OTServ database handler interface.
This interface specifies routines requires by DAO classes.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Don't rely on this interface - it is for backward compatibility only. Check POT instance instead.
- **Since** 0.0.1

Constructor *void* function IOTS_DB::__construct($params) *[line 32]*

Function Parameters:

- *array* **$params** Connection configuration.

Connection parameters.

Connection parameters.

- **Version** 0.0.1

- **Since** 0.0.1
- **Access** public

string function IOTS_DB::fieldName($name) *[line 40]*
Function Parameters:

- *string* **$name** Field name.

Query-quoted field name.
Query-quoted field name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

int function IOTS_DB::lastInsertId() *[line 67]*
ID of last created record.
ID of last created record.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

string function IOTS_DB::limit([$limit = false], [$offset = false]) *[line 75]*
Function Parameters:

- *int|bool* **$limit** Limit of rows to be affected by query (false if no limit).

- *int|bool* **$offset** Number of rows to be skipped before applying query effects (false if no offset).

LIMIT/OFFSET clause for queries.

LIMIT/OFFSET clause for queries.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

mixed function IOTS_DB::SQLquery($query) *[line 61]*

Function Parameters:

- *string* **$query** Database query.

Evaluates query.

Evaluates query.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

string function IOTS_DB::SQLquote($value) *[line 54]*

Function Parameters:

- *string* **$value** Value to be quoted to be suitable for database query.

Query-quoted string value.

Query-quoted string value.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

string function IOTS_DB::tableName($name) *[line 47]*

Function Parameters:

- *string* **$name** Table name.

Query-quoted table name.

Query-quoted table name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

Class IOTS_GuildAction

[line 32]

Guild action interface.

Guild action interface.

This insterface indicates that class can handle OTServ guild action.

You can use it for example to handle invites or membership requests.

If you want to serialise (for example save in session) your guild obejcts with assigned drivers you need to implement also __sleep() and __wakeup() methods in your drivers, as assigned drivers are also serialised.

- **Package** POT
- **Version** 0.0.4
- **Since** 0.0.4

Constructor *void* function IOTS_GuildAction::__construct($guild) *[line 41]*

Function Parameters:

- *OTS_Guild* **$guild** Guild that this driver is assigned to.

Objects are initialized with a guild that they are assigned to.

Objects are initialized with a guild that they are assigned to.

It is recommeded that your implementations calls assignment functions of $guild to automaticly assign itself as action handler.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function IOTS_GuildAction::addRequest($player) *[line 54]*

Function Parameters:

- *OTS_Player* **$player** Player which is object of request.

Adds new request.

Adds new request.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function IOTS_GuildAction::deleteRequest($player) *[line 60]*

Function Parameters:

- *OTS_Player* **$player** Player which is object of request.

Deletes request.

Deletes request.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

array function IOTS_GuildAction::listRequests() *[line 48]*

List of saved pending actions.

List of saved pending actions.

- **Version** 0.0.4
- **Since** 0.0.4

- **Access** public

void function IOTS_GuildAction::submitRequest($player) *[line 66]*
Function Parameters:

- *OTS_Player* **$player** Player which is object of request.

Finalizes request.
Finalizes request.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

Class OTS_Account
[line 22]

OTServ account abstraction.
OTServ account abstraction.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

void function OTS_Account::ban([$time = 0]) *[line 466]*

Function Parameters:

- *int* **$time** Time for time until expires (0 - forever).

Bans current account.

Bans current account.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Account::block() *[line 312]*

Blocks account.

Blocks account.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

int function OTS_Account::count() *[line 557]*

Returns number of player within.

Returns number of player within.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

int function OTS_Account::create([$min = 1], [$max = 9999999]) *[line 51]*

account.php

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
 */

// to not repeat all that stuff
include('quickstart.php');

// creates new OTS_Account object
$account = $ots->   createObject('Account');

// generates new account number
$number = $account->   create();

/*
to generate number from 111111 to 999999 use:
$number = $account->create(111111, 999999);
*/

// sets account info
$account->   setPassword('secret'); // $account->setPassword( md5('secret') );
$account->   setEMail('foo@example.com');
$account->   unblock(); // remember to unblock!
$account->   setPACCDays(0);
$account->   save();

// give user his number
echo 'Your account number is: ', $number;

?>
```

Function Parameters:

- *int* **$min** Minimum number.
- *int* **$max** Maximum number.

Creates new account.

Creates new account.

Create new account in given range (1 - 9999999 by default).
Remember! This method sets blocked flag to true after account creation!

IMPORTANT: Since 0.0.4 there is group_id field which this method does not support. Account's group_id is set to first one found in database. You should use createEx() method if you want to set group_id field during creation.

- **Version** 0.0.4
- **Version** 0.0.1
- **Throws** Exception When there are no free account numbers.
- **Since** 0.0.1
- **Access** public
- **Example**

int function OTS_Account::createEx($group, [$min = 1], [$max = 9999999]) *[line 77]*

account.php

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
 */

// to not repeat all that stuff
include('quickstart.php');

// creates new OTS_Account object
$account = $ots->   createObject('Account');

// group for account
$group = $ots->   createObject('Group');

// loads group with id 1
$group->   load(1);

// generates new account number
$number = $account->   createEx($group);

// give user his number
echo 'Your account number is: ', $number;

?>
```

Function Parameters:

- *OTS_Group* **$group** Group to be assigned to account.
- *int* **$min** Minimum number.
- *int* **$max** Maximum number.

Creates new account.

Creates new account.

Create new account in given range (1 - 9999999 by default) in given group.
Remember! This method sets blocked flag to true after account creation!

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** Exception When there are no free account numbers.
- **Since** 0.0.1
- **Since** 0.0.4
- **Access** public
- **Example**

void function OTS_Account::delete() *[line 520]*

Deletes account.

Deletes account.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.4+SVN

- **Since** 0.0.1
- **Access** public

void function OTS_Account::find($email) *[line 144]*

Function Parameters:

- *string* **$email** Account's e-mail address.

Loads account by it's e-mail address.

Loads account by it's e-mail address.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2
- **Access** public

string function OTS_Account::getCustomField($field) *[line 359]*

Function Parameters:

- *string* **$field** Field name.

Reads custom field.

Reads custom field.

Reads field by it's name. Can read any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

string function OTS_Account::getEMail() *[line 264]*

E-mail address.

E-mail address.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.1
- **Access** public

OTS_Group function OTS_Account::getGroup() *[line 208]*

Returns group of this account.

Returns group of this account.

- **Version** 0.0.4
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.4
- **Since** 0.0.1

- **Access** public

int function OTS_Account::getId() *[line 190]*

Account number.

Account number.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.1
- **Access** public

Iterator function OTS_Account::getIterator() *[line 544]*

Returns players iterator.

Returns players iterator.

There is no need to implement entire Iterator interface since we have players list class for it.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

int function OTS_Account::getPACCDays() *[line 325]*

PACC days.

PACC days.

- **Version** 0.0.4
- **Version** 0.0.1
- **Deprecated** 0.0.3 There is no more premdays field in accounts table.
- **Since** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Access** public

string function OTS_Account::getPassword() *[line 237]*

Account's password.

Account's password.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.1
- **Access** public

array function OTS_Account::getPlayers() *[line 409]*

List of characters on account.

List of characters on account.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Use getPlayersList().

- **Since** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Access** public

OTS_Players_List function OTS_Account::getPlayersList() *[line 439]*

List of characters on account.

List of characters on account.

In difference to getPlayers() method this method returns filtered OTS_Players_List object instead of array of OTS_Player objects. It is more effective since OTS_Player_List doesn't perform all rows loading at once.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

bool function OTS_Account::isBanned() *[line 501]*

Checks if account is banned.

Checks if account is banned.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

bool function OTS_Account::isBlocked() *[line 291]*

Checks if account is blocked.

Checks if account is blocked.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.1
- **Access** public

bool function OTS_Account::isLoaded() *[line 161]*

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Account::load($id) *[line 131]*

Function Parameters:

- *int* **$id** Account number.

Loads account with given number.

Loads account with given number.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Account::save() *[line 172]*

Updates account in database.

Updates account in database.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded False if account doesn't have ID assigned.
- **Since** 0.0.1
- **Access** public

void function OTS_Account::setCustomField($field, $value) *[line 385]*

Function Parameters:

- *string* **$field** Field name.
- *mixed* **$value** Field value.

Writes custom field.

Writes custom field.

Write field by it's name. Can write any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

Note: Make sure that you pass $value argument of correct type. This method determinates whether to quote field name. It is safe - it makes you sure that no unproper queries that could

lead to SQL injection will be executed, but it can make your code working wrong way. For example: $object->setCustomField('foo', '1'); will quote 1 as as string ('1') instead of passing it as a integer.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If account is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

void function OTS_Account::setEMail($email) *[line 279]*

Function Parameters:

- *string* **$email** E-mail address.

Sets account's email.

Sets account's email.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Account::setGroup($group) *[line 225]*

Function Parameters:

- *OTS_Group* **$group** Group to be a member.

Assigns account to group.

Assigns account to group.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Account::setPACCDays($premdays, $pacc) *[line 342]*

Function Parameters:

- *int* **$pacc** PACC days.
- **$premdays**

Sets PACC days count.

Sets PACC days count.

- **Version** 0.0.4
- **Version** 0.0.1
- **Deprecated** 0.0.3 There is no more premdays field in accounts table.
- **Since** 0.0.1
- **Access** public

void function OTS_Account::setPassword($password) *[line 252]*

Function Parameters:

- *string* **$password** Password.

Sets account's password.

Sets account's password.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Account::unban() *[line 483]*

Deletes ban from current account.

Deletes ban from current account.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Account::unblock() *[line 304]*

Unblocks account.

Unblocks account.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

Class OTS_Accounts_List

[line 21]

List of accounts.

List of accounts.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.1

void function OTS_Accounts_List::deleteAccount($account) *[line 30]*

Function Parameters:

- *OTS_Account* **$account** Account to be deleted.

Deletes account.

Deletes account.

- **Version** 0.0.4+SVN
- **Deprecated** 0.0.4+SVN Use OTS_Account->delete().
- **Since** 0.0.1
- **Access** public

void function OTS_Accounts_List::init() *[line 43]*

Sets list parameters.

Sets list parameters.

This method is called at object creation.

- **Version** 0.0.4+SVN
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_Base_DAO

[line 20]

Basic data access object routines.

Basic data access object routines.

- **Package** POT
- **Version** 0.0.4+SVN
- **Abstract Element**
- **Since** 0.0.4+SVN

OTS_Base_DAO::$db

PDO = [line 27]

Database connection.

Database connection.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** protected

Constructor *void* function OTS_Base_DAO::__construct($db) *[line 34]*

Function Parameters:

- *PDO* **$db** Database connection object.

Sets database connection handler.

Sets database connection handler.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_DAO::__clone() *[line 71]*

Creates clone of object.

Creates clone of object.
Copy of object needs to have different ID.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_DAO::__set_state($properties) *[line 84]*

Function Parameters:

- *array* **$properties** List of object properties.

Magic PHP5 method.

Magic PHP5 method.

Allows object importing from var_export().

- **Version** 0.0.4+SVN
- **Static**
- **Since** 0.0.4+SVN
- **Access** public

array function OTS_Base_DAO::__sleep() *[line 47]*

Magic PHP5 method.

Magic PHP5 method.

Allows object serialisation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_DAO::__wakeup() *[line 59]*

Magic PHP5 method.

Magic PHP5 method.

Allows object unserialisation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_Base_List

[line 20]

Basic list class routines.

Basic list class routines.

- **Package** POT
- **Version** 0.0.4+SVN
- **Abstract Element**
- **Since** 0.0.4+SVN

Constructor *void* function OTS_Base_List::__construct($db) *[line 83]*

Function Parameters:

- *PDO* **$db** Database connection object.

Sets database connection handler.

Sets database connection handler.

- **Version** 0.0.4+SVN

- **Since** 0.0.4+SVN
- **Access** public

int function OTS_Base_List::count() *[line 237]*

Returns number of accounts on list in current criterium.

Returns number of accounts on list in current criterium.

- **Version** 0.0.4+SVN
- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

IOTS_DAO function OTS_Base_List::current() *[line 186]*

Returns current row.

Returns current row.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::init() *[line 92]*

Sets list parameters.

Sets list parameters.

- **Version** 0.0.4+SVN

- **Abstract Element**
- **Since** 0.0.4+SVN
- **Access** public

mixed function OTS_Base_List::key() *[line 216]*

Current cursor position.

Current cursor position.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::next() *[line 206]*

Moves to next row.

Moves to next row.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::orderBy($filed, [$order = POT::ORDER_ASC], $field) *[line 269]*

Function Parameters:

- *string* **$field** Field name.
- *int* **$order** Sorting order (ascending by default).
- **$filed**

Appends sorting rule.

Appends sorting rule.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::resetOrder() *[line 258]*

Clears ORDER BY clause.

Clears ORDER BY clause.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::rewind() *[line 198]*

Select rows from database.

Select rows from database.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::setFilter([$filter = null]) *[line 250]*

Function Parameters:

- *OTS_SQLFilter*|*null* **$filter** Filter for list.

Sets filter on list.

Sets filter on list.
Call without argument to reset filter.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::setLimit([$limit = false]) *[line 152]*

Function Parameters:

- *int*|*bool* **$limit** Limit for SELECT (false to reset).

Sets LIMIT.

Sets LIMIT.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::setOffset([$offset = false]) *[line 169]*

Function Parameters:

- *int*|*bool* **$offset** Offset for SELECT (false to reset).

Sets OFFSET.

Sets OFFSET.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

bool function OTS_Base_List::valid() *[line 226]*

Checks if there are any rows left.

Checks if there are any rows left.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::__set_state($properties) *[line 127]*

Function Parameters:

- *array* **$properties** List of object properties.

Magic PHP5 method.

Magic PHP5 method.

Allows object importing from var_export().

- **Version** 0.0.4+SVN
- **Static**
- **Since** 0.0.4+SVN
- **Access** public

array function OTS_Base_List::__sleep() *[line 102]*

Magic PHP5 method.

Magic PHP5 method.

Allows object serialisation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Base_List::__wakeup() *[line 114]*

Magic PHP5 method.

Magic PHP5 method.

Allows object unserialisation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_Container

[line 20]

Container item representation.

Container item representation.

- **Package** POT
- **Version** 0.0.3
- **Since** 0.0.3

void function OTS_Container::addItem($item) *[line 34]*

Function Parameters:

- *OTS_Item* **$item** Item.

Adds item to container.

Adds item to container.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

int function OTS_Container::count() *[line 65]*

Number of items inside container.

Number of items inside container.

OTS_Container implementation of Countable interface differs from OTS_Item implemention. OTS_Item::count() returns count of given item, OTS_Container::count() returns number of items inside container. If somehow it would be possible to make container items with more then 1 in one place, you can use OTS_Item::getCount() and OTS_Item::setCount() in code where you are not sure if working with regular item, or container.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

OTS_Item function OTS_Container::current() *[line 75]*

Returns current item.

Returns current item.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

mixed function OTS_Container::key() *[line 93]*

Current cursor position.

Current cursor position.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

void function OTS_Container::next() *[line 83]*

Moves to next item.

Moves to next item.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

void function OTS_Container::removeItem($item) *[line 46]*

Function Parameters:

- *OTS_Item* **$item** Item.

Removes given item from current container.

Removes given item from current container.
Passed item must be exacly instance of item which is stored in container, not it's copy.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

void function OTS_Container::rewind() *[line 111]*

Resets internal items array pointer.

Resets internal items array pointer.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

bool function OTS_Container::valid() *[line 103]*

Checks if there are any items left.

Checks if there are any items left.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

Class OTS_DB_MySQL

[line 22]

MySQL connection interface.

MySQL connection interface.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

Constructor *void* function OTS_DB_MySQL::__construct($params) *[line 49]*

Function Parameters:

- *array* **$params** Connection parameters.

Creates database connection.

Creates database connection.

Connects to MySQL database on given arguments.

List of parameters for this drivers:

- *host* - database server.
- *port* - port (optional, also it is possible to use host:port in *host* parameter).
- *database* - database name.
- *user* - user login.
- *password* - user password.

- **Version** 0.0.1
- **See** POT::connect()
- **Since** 0.0.1
- **Access** public

string function OTS_DB_MySQL::fieldName($name) *[line 104]*

Function Parameters:

- *string* **$name** Field name.

Query-quoted field name.

Query-quoted field name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

string function OTS_DB_MySQL::limit([$limit = false], [$offset = false]) *[line 157]*

Function Parameters:

- *int|bool* **$limit** Limit of rows to be affected by query (false if no limit).

- *int|bool* **$offset** Number of rows to be skipped before applying query effects (false if no offset).

LIMIT/OFFSET clause for queries.

LIMIT/OFFSET clause for queries.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

PDOStatement|bool function OTS_DB_MySQL::SQLquery($query) *[line 145]*

Function Parameters:

- *string* **$query** SQL query.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Use PDO::query().
- **Since** 0.0.1
- **Access** public

string function OTS_DB_MySQL::SQLquote($string) *[line 130]*

Function Parameters:

- *stirng* **$string** String to be quoted.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method - we won't use quoting agains other values.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Use PDO::quote().
- **Since** 0.0.1
- **Access** public

string function OTS_DB_MySQL::tableName($name) *[line 115]*

Function Parameters:

- *string* **$name** Table name.

Query-quoted table name.

Query-quoted table name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

Class OTS_DB_ODBC

[line 22]

ODBC connection interface.

ODBC connection interface.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4

Constructor *void* function OTS_DB_ODBC::__construct($params) *[line 49]*

Function Parameters:

- *array* **$params** Connection parameters.

Creates database connection.

Creates database connection.

Connects to ODBC data source on given arguments.

List of parameters for this drivers:

- *host* - database host.
- *port* - ODBC driver.
- *database* - database name.
- *user* - user login.
- *password* - user password.

- **Version** 0.0.4
- **See** POT::connect()
- **Since** 0.0.4
- **Access** public

string function OTS_DB_ODBC::fieldName($name) *[line 97]*
Function Parameters:

- *string* **$name** Field name.

Query-quoted field name.
Query-quoted field name.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

string function OTS_DB_ODBC::limit([$limit = false], [$offset = false]) *[line 150]*
Function Parameters:

- *int|bool* **$limit** Limit of rows to be affected by query (false if no limit).
- *int|bool* **$offset** Number of rows to be skipped before applying query effects (false if no offset).

LIMIT/OFFSET clause for queries.
LIMIT/OFFSET clause for queries.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

PDOStatement|bool function OTS_DB_ODBC::SQLquery($query) *[line 138]*
Function Parameters:

- *string* **$query** SQL query.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

- **Version** 0.0.4
- **Deprecated** 0.0.4+SVN Use PDO::query().
- **Since** 0.0.4
- **Access** public

string function OTS_DB_ODBC::SQLquote($string) *[line 123]*

Function Parameters:

- *stirng* **$string** String to be quoted.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method - we won't use quoting agains other values.

- **Version** 0.0.4
- **Deprecated** 0.0.4+SVN Use PDO::quote().
- **Since** 0.0.4
- **Access** public

string function OTS_DB_ODBC::tableName($name) *[line 108]*

Function Parameters:

- *string* **$name** Table name.

Query-quoted table name.
Query-quoted table name.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

Class OTS_DB_PostgreSQL
[line 22]

PostgreSQL connection interface.
PostgreSQL connection interface.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4

Constructor *void* function OTS_DB_PostgreSQL::__construct($params) *[line 49]*
Function Parameters:

- *array* **$params** Connection parameters.

Creates database connection.

Creates database connection.

Connects to PgSQL database on given arguments.

List of parameters for this drivers:

- *host* - database server.
- *port* - port (optional, also it is possible to use host:port in *host* parameter).
- *database* - database name.
- *user* - user login.
- *password* - user password.

- **Version** 0.0.4
- **See** POT::connect()
- **Since** 0.0.4
- **Access** public

string function OTS_DB_PostgreSQL::fieldName($name) *[line 104]*

Function Parameters:

- *string* **$name** Field name.

Query-quoted field name.

Query-quoted field name.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

string function OTS_DB_PostgreSQL::limit([$limit = false], [$offset = false]) *[line 157]*

Function Parameters:

- *int|bool* **$limit** Limit of rows to be affected by query (false if no limit).
- *int|bool* **$offset** Number of rows to be skipped before applying query effects (false if no offset).

LIMIT/OFFSET clause for queries.

LIMIT/OFFSET clause for queries.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

PDOStatement|bool function OTS_DB_PostgreSQL::SQLquery($query) *[line 145]*

Function Parameters:

- *string* **$query** SQL query.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

- **Version** 0.0.4
- **Deprecated** 0.0.4+SVN Use PDO::query().
- **Since** 0.0.4
- **Access** public

string function OTS_DB_PostgreSQL::SQLquote($string) *[line 130]*

Function Parameters:

- *stirng* **$string** String to be quoted.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method - we won't use quoting agains other values.

- **Version** 0.0.4
- **Deprecated** 0.0.4+SVN Use PDO::quote().
- **Since** 0.0.4
- **Access** public

string function OTS_DB_PostgreSQL::tableName($name) *[line 115]*

Function Parameters:

- *string* **$name** Table name.

Query-quoted table name.

Query-quoted table name.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

Class OTS_DB_SQLite

[line 22]

SQLite connection interface.

SQLite connection interface.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

Constructor *void* function OTS_DB_SQLite::__construct($params) *[line 45]*

Function Parameters:

- *array* **$params** Connection parameters.

Creates database connection.

Creates database connection.

Connects to SQLite database on given arguments.

List of parameters for this drivers:

- *database* - database name.

- **Version** 0.0.1
- **See** POT::connect()
- **Since** 0.0.1
- **Access** public

string function OTS_DB_SQLite::fieldName($name) *[line 65]*

Function Parameters:

- *string* **$name** Field name.

Query-quoted field name.

Query-quoted field name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

string function OTS_DB_SQLite::limit([$limit = false], [$offset = false]) *[line 118]*

Function Parameters:

- *int|bool* **$limit** Limit of rows to be affected by query (false if no limit).
- *int|bool* **$offset** Number of rows to be skipped before applying query effects (false if no offset).

LIMIT/OFFSET clause for queries.

LIMIT/OFFSET clause for queries.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

PDOStatement|bool function OTS_DB_SQLite::SQLquery($query) *[line 106]*

Function Parameters:

- *string* **$query** SQL query.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Use PDO::query().
- **Since** 0.0.1
- **Access** public

string function OTS_DB_SQLite::SQLquote($string) *[line 91]*

Function Parameters:

- *stirng* **$string** String to be quoted.

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method - we won't use quoting agains other values.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Use PDO::quote().
- **Since** 0.0.1
- **Access** public

string function OTS_DB_SQLite::tableName($name) *[line 76]*

Function Parameters:

- *string* **$name** Table name.

Query-quoted table name.

Query-quoted table name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

Class OTS_Group

[line 22]

OTServ user group abstraction.

OTServ user group abstraction.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

int function OTS_Group::count() *[line 385]*

Returns number of player within.

Returns number of player within.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

void function OTS_Group::delete() *[line 348]*

Deletes group.

Deletes group.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

int function OTS_Group::getAccess() *[line 154]*

Access level.

Access level.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.1
- **Access** public

string function OTS_Group::getCustomField($field) *[line 241]*

Function Parameters:

- *string* **$field** Field name.

Reads custom field.

Reads custom field.

Reads field by it's name. Can read any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

int function OTS_Group::getFlags() *[line 127]*

Rights flags.

Rights flags.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Group::getId() *[line 83]*

Group ID.

Group ID.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.1
- **Access** public

Iterator function OTS_Group::getIterator() *[line 372]*

Returns players iterator.

Returns players iterator.

There is no need to implement entire Iterator interface since we have players list class for it.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1

- **Access** public

int function OTS_Group::getMaxDepotItems() *[line 181]*

Maximum count of items in depot.

Maximum count of items in depot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Group::getMaxVIPList() *[line 208]*

Maximum count of players in VIP list.

Maximum count of players in VIP list.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.1
- **Access** public

string function OTS_Group::getName() *[line 100]*

Group name.

Group name.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.1
- **Access** public

array function OTS_Group::getPlayers() *[line 291]*

List of characters in given group.

List of characters in given group.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Use getPlayersList().
- **Since** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Access** public

OTS_Players_List function OTS_Group::getPlayersList() *[line 321]*

List of characters in group.

List of characters in group.

In difference to getPlayers() method this method returns filtered OTS_Players_List object instead of array of OTS_Player objects. It is more effective since OTS_Player_List doesn't perform all rows loading at once.

- **Version** 0.0.4+SVN
- **Version** 0.0.1

- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

bool function OTS_Group::isLoaded() *[line 48]*

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::load($id) *[line 37]*

Function Parameters:

- *int* **$id** Group number.

Loads group with given id.

Loads group with given id.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::save() *[line 58]*

Saves account in database.

Saves account in database.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::setAccess($access) *[line 169]*

Function Parameters:

- *int* **$access** Access level.

Sets access level.

Sets access level.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::setCustomField($field, $value) *[line 267]*

Function Parameters:

- *string* **$field** Field name.
- *mixed* **$value** Field value.

Writes custom field.

Writes custom field.

Write field by it's name. Can write any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

Note: Make sure that you pass $value argument of correct type. This method determinates whether to quote field name. It is safe - it makes you sure that no unproper queries that could lead to SQL injection will be executed, but it can make your code working wrong way. For example: $object->setCustomField('foo', '1'); will quote 1 as as string ('1') instead of passing it as a integer.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If group is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

void function OTS_Group::setFlags($flags) *[line 142]*

Function Parameters:

- *int* **$flags** Flags.

Sets rights flags.

Sets rights flags.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::setMaxDepotItems($maxdepotitems) *[line 196]*
Function Parameters:

- *int* **$maxdepotitems** Maximum value.

Sets maximum count of items in depot.
Sets maximum count of items in depot.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::setMaxVIPList($maxviplist, $maxdepotitems) *[line 223]*
Function Parameters:

- *int* **$maxdepotitems** Maximum value.
- **$maxviplist**

Sets maximum count of players in VIP list.
Sets maximum count of players in VIP list.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Group::setName($name) *[line 115]*

Function Parameters:

- *string* **$name** Name.

Sets group's name.

Sets group's name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

Class OTS_Groups_List

[line 21]

List of groups.

List of groups.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.1

void function OTS_Groups_List::deleteGroup($group) *[line 30]*

Function Parameters:

- *OTS_Group* **$group** Group to be deleted.

Deletes group.

Deletes group.

- **Version** 0.0.4+SVN
- **Deprecated** 0.0.4+SVN Use OTS_Group->delete().
- **Since** 0.0.1
- **Access** public

void function OTS_Groups_List::init() *[line 43]*

Sets list parameters.

Sets list parameters.

This method is called at object creation.

- **Version** 0.0.4+SVN
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_Guild

[line 22]

OTServ guild abstraction.

OTServ guild abstraction.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4

void function OTS_Guild::acceptInvite($player) *[line 442]*
Function Parameters:

- *OTS_Player* **$player** Player to be joined.

Finalise invitation.
Finalise invitation.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::acceptRequest($player) *[line 534]*
Function Parameters:

- *OTS_Player* **$player** Player to be accepted.

Accepts player.
Accepts player.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.4
- **Access** public

int function OTS_Guild::count() *[line 594]*

Returns number of ranks within.

Returns number of ranks within.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::delete() *[line 557]*

Deletes guild.

Deletes guild.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4+SVN

- **Since** 0.0.4
- **Access** public

void function OTS_Guild::deleteInvite($player) *[line 419]*
Function Parameters:

- *OTS_Player* **$player** Player to be un-invited.

Deletes invitation for player to guild.
Deletes invitation for player to guild.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::deleteRequest($player) *[line 511]*
Function Parameters:

- *OTS_Player* **$player** Player to be rejected.

Deletes request from player.
Deletes request from player.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.

- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::find($name) *[line 114]*

Function Parameters:

- *string* **$name** Guild's name.

Loads guild by it's name.

Loads guild by it's name.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

int function OTS_Guild::getCreationData() *[line 235]*

Guild creation data.

Guild creation data.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4
- **Access** public

string function OTS_Guild::getCustomField($field) *[line 267]*

Function Parameters:

- *string* **$field** Field name.

Reads custom field.

Reads custom field.

Reads field by it's name. Can read any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4
- **Access** public

array function OTS_Guild::getGuildRanks() *[line 316]*

Reads all ranks that are in this guild.

Reads all ranks that are in this guild.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Deprecated** 0.0.4+SVN Use getGuildRanksList().
- **Since** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Access** public

OTS_GuildRanks_List function OTS_Guild::getGuildRanksList() *[line 346]*

List of ranks in guild.

List of ranks in guild.

In difference to getGuildRanks() method this method returns filtered OTS_GuildRanks_List object instead of array of OTS_GuildRank objects. It is more effective since OTS_GuildRanks_List doesn't perform all rows loading at once.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

int function OTS_Guild::getId() *[line 165]*

Guild ID.

Guild ID.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4
- **Access** public

Iterator function OTS_Guild::getIterator() *[line 581]*

Returns ranks iterator.

Returns ranks iterator.

There is no need to implement entire Iterator interface since we have ranks list class for it.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

string function OTS_Guild::getName() *[line 181]*

Guild name.

Guild name.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4
- **Access** public

OTS_Player function OTS_Guild::getOwner() *[line 207]*

Returns owning player of this player.

Returns owning player of this player.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::invite($player) *[line 396]*

Function Parameters:

- *OTS_Player* **$player** Player to be invited.

Invites player to guild.

Invites player to guild.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.4
- **Access** public

bool function OTS_Guild::isLoaded() *[line 131]*

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

array function OTS_Guild::listInvites() *[line 373]*

Returns list of invited players.

Returns list of invited players.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.4
- **Access** public

array function OTS_Guild::listRequests() *[line 465]*

Returns list of players that requested membership.

Returns list of players that requested membership.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::load($id) *[line 102]*

Function Parameters:

- *int* **$id** Guild's ID.

Loads guild with given id.

Loads guild with given id.

- **Version** 0.0.4+SVN

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::request($player) *[line 488]*
Function Parameters:

- *OTS_Player* **$player** Player that requested membership.

Requests membership in guild for player player.
Requests membership in guild for player player.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::save() *[line 141]*
Saves guild in database.
Saves guild in database.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::setCreationData($creationdata) *[line 250]*

Function Parameters:

- *int* **$creationdata** Guild creation data.

Sets guild creation data.

Sets guild creation data.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::setCustomField($field, $value) *[line 292]*

Function Parameters:

- *string* **$field** Field name.
- *mixed* **$value** Field value.

Writes custom field.

Writes custom field.

Write field by it's name. Can write any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

Note: Make sure that you pass $value argument of correct type. This method determinates whether to quote field value. It is safe - it makes you sure that no unproper queries that could lead to SQL injection will be executed, but it can make your code working wrong way. For example: $object->setCustomField('foo', '1'); will quote 1 as as string ('1') instead of passing it as a integer.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::setInvitesDriver([$invites = null]) *[line 81]*
Function Parameters:

- *IOTS_GuildAction* **$invites** Invites driver (don't pass it to clear driver).

Assigns invites handler.
Assigns invites handler.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::setName($name) *[line 196]*
Function Parameters:

- *string* **$name** Name.

Sets players's name.
Sets players's name.

- **Version** 0.0.4

- **Since** 0.0.4
- **Access** public

void function OTS_Guild::setOwner($owner) *[line 224]*
Function Parameters:

- *OTS_Player* **$owner** Owning player.

Assigns guild to owner.
Assigns guild to owner.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::setRequestsDriver([$requests = null]) *[line 91]*
Function Parameters:

- *IOTS_GuildAction* **$requests** Membership requests driver (don't pass it to clear driver).

Assigns requests handler.
Assigns requests handler.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_Guild::__clone() *[line 65]*

Creates clone of object.

Creates clone of object.
Copy of object needs to have different ID.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

array function OTS_Guild::__sleep() *[line 53]*

Magic PHP5 method.

Magic PHP5 method.
Allows object serialisation.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

Class OTS_GuildRank

[line 22]

OTServ guild rank abstraction.

OTServ guild rank abstraction.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4

int function OTS_GuildRank::count() *[line 356]*

Returns number of player within.

Returns number of player within.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::delete() *[line 319]*

Deletes guild rank.

Deletes guild rank.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If guild rank is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::find($name, [$guild = null]) *[line 52]*

Function Parameters:

- *string* **$name** Rank's name.
- *OTS_Guild* **$guild** Guild in which rank should be found.

Loads rank by it's name.

Loads rank by it's name.

As there can be several ranks with same name in different guilds you can pass optional second parameter to specify in which guild script should look for rank.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

string function OTS_GuildRank::getCustomField($field) *[line 213]*

Function Parameters:

- *string* **$field** Field name.

Reads custom field.

Reads custom field.

Reads field by it's name. Can read any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

- **Version** 0.0.4+SVN

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4
- **Access** public

OTS_Guild function OTS_GuildRank::getGuild() *[line 153]*

Returns guild of this rank.

Returns guild of this rank.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4
- **Access** public

int function OTS_GuildRank::getId() *[line 111]*

Rank ID.

Rank ID.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4
- **Access** public

Iterator function OTS_GuildRank::getIterator() *[line 343]*

Returns players iterator.

Returns players iterator.

There is no need to implement entire Iterator interface since we have players list class for it.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

int function OTS_GuildRank::getLevel() *[line 181]*

Rank's access level.

Rank's access level.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4
- **Access** public

string function OTS_GuildRank::getName() *[line 127]*

Rank name.

Rank name.

- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4

- **Access** public

array function OTS_GuildRank::getPlayers() *[line 262]*

Reads all players who has this rank set.

Reads all players who has this rank set.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Deprecated** 0.0.4+SVN Use getPlayersList().
- **Since** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Access** public

OTS_Players_List function OTS_GuildRank::getPlayersList() *[line 292]*

List of characters with current rank.

List of characters with current rank.

In difference to getPlayers() method this method returns filtered OTS_Players_List object instead of array of OTS_Player objects. It is more effective since OTS_Player_List doesn't perform all rows loading at once.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.4
- **Access** public

bool function OTS_GuildRank::isLoaded() *[line 77]*

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::load($id) *[line 37]*

Function Parameters:

- *int* **$id** Rank's ID.

Loads rank with given id.

Loads rank with given id.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::save() *[line 87]*

Saves rank in database.

Saves rank in database.

- **Version** 0.0.4+SVN

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::setCustomField($field, $value) *[line 238]*

Function Parameters:

- *string* **$field** Field name.
- *mixed* **$value** Field value.

Writes custom field.

Writes custom field.

Write field by it's name. Can write any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

Note: Make sure that you pass $value argument of correct type. This method determinates whether to quote field value. It is safe - it makes you sure that no unproper queries that could lead to SQL injection will be executed, but it can make your code working wrong way. For example: $object->setCustomField('foo', '1'); will quote 1 as as string ('1') instead of passing it as a integer.

- **Version** 0.0.4+SVN
- **Version** 0.0.4
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::setGuild($guild) *[line 170]*

Function Parameters:

- *OTS_Guild* **$guild** Owning guild.

Assigns rank to guild.

Assigns rank to guild.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::setLevel($level) *[line 196]*

Function Parameters:

- *int* **$level** access level within guild.

Sets rank's access level within guild.

Sets rank's access level within guild.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRank::setName($name) *[line 142]*

Function Parameters:

- *string* **$name** Name.

Sets rank's name.

Sets rank's name.

- **Version** 0.0.4
- **Since** 0.0.4
- **Access** public

Class OTS_GuildRanks_List

[line 21]

List of guild ranks.

List of guild ranks.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.4

void function OTS_GuildRanks_List::deleteGuildRank($guildRank) *[line 30]*

Function Parameters:

- *OTS_GuildRank* **$guildRank** Rank to be deleted.

Deletes guild rank.

Deletes guild rank.

- **Version** 0.0.4+SVN
- **Deprecated** 0.0.4+SVN Use OTS_GuildRank->delete().
- **Since** 0.0.4
- **Access** public

void function OTS_GuildRanks_List::init() *[line 43]*

Sets list parameters.

Sets list parameters.

This method is called at object creation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_Guilds_List

[line 21]

List of guilds.

List of guilds.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.4

void function OTS_Guilds_List::deleteGuild($guild) *[line 30]*

Function Parameters:

- *OTS_Guild* **$guild** Guild to be deleted.

Deletes guild.

Deletes guild.

- **Version** 0.0.4+SVN
- **Deprecated** 0.0.4+SVN Use OTS_Guild->delete().
- **Since** 0.0.4
- **Access** public

void function OTS_Guilds_List::init() *[line 43]*

Sets list parameters.

Sets list parameters.

This method is called at object creation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_InfoRespond

[line 22]

Wrapper for 'info' respond's DOMDocument.

Wrapper for 'info' respond's DOMDocument.

Note: as this class extends DOMDocument class and contains exacly respond XML tree you can work on it as on normal DOM tree.

- **Package** POT
- **Version** 0.0.2
- **Since** 0.0.2

string function OTS_InfoRespond::getClientVersion() *[line 121]*

Returns dedicated version of client.

Returns dedicated version of client.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getEMail() *[line 141]*

Returns owner e-mail.

Returns owner e-mail.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getIP() *[line 49]*

Returns server IP.

Returns server IP.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getLocation() *[line 79]*

Returns server location.

Returns server location.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getMapAuthor() *[line 202]*

Returns map author.

Returns map author.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getMapHeight() *[line 222]*

Returns map height.

Returns map height.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getMapName() *[line 191]*

Returns map name.

Returns map name.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getMapWidth() *[line 212]*

Returns map width.

Returns map width.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getMaxPlayers() *[line 161]*

Returns maximum amount of players online.

Returns maximum amount of players online.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getMonstersCount() *[line 181]*

Returns number of all monsters on map.

Returns number of all monsters on map.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getMOTD() *[line 232]*

Returns server's Message Of The Day

Returns server's Message Of The Day

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getName() *[line 59]*

Returns server name.

Returns server name.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getOnlinePlayers() *[line 151]*

Returns current amount of players online.

Returns current amount of players online.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getOwner() *[line 131]*

Returns owner name.

Returns owner name.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getPlayersPeak() *[line 171]*

Returns record of online players.

Returns record of online players.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getPort() *[line 69]*

Returns server port.

Returns server port.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getServer() *[line 101]*

Returns server attribute.

Returns server attribute.

I have no idea what the hell is it representing :P.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getServerVersion() *[line 111]*

Returns server version.

Returns server version.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getTSPQVersion() *[line 29]*

Returns version of root element.

Returns version of root element.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

int function OTS_InfoRespond::getUptime() *[line 39]*

Returns server uptime.

Returns server uptime.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

string function OTS_InfoRespond::getURL() *[line 89]*

Returns server website.

Returns server website.

- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public

Class OTS_Item

[line 20]

Single item representation.

Single item representation.

- **Package** POT
- **Version** 0.0.3
- **Since** 0.0.3

Constructor *void* function OTS_Item::__construct($id) *[line 48]*

Function Parameters:

- *int* **$id** Item ID.

Creates item of given ID.

Creates item of given ID.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

int function OTS_Item::count() *[line 108]*

Count value for current item.

Count value for current item.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

string function OTS_Item::getAttributes() *[line 88]*

Returns item custom attributes.

Returns item custom attributes.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

int function OTS_Item::getCount() *[line 68]*

Returns count of item.

Returns count of item.

- **Version** 0.0.3

- **Since** 0.0.3
- **Access** public

int function OTS_Item::getId() *[line 58]*

Returns item type.

Returns item type.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

void function OTS_Item::setAttributes($attributes) *[line 98]*

Function Parameters:

- *string* **$attributes** Item Attributes.

Sets item attributes.

Sets item attributes.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

void function OTS_Item::setCount($count) *[line 78]*

Function Parameters:

- *int* **$count** Count.

Sets count of item.

Sets count of item.

- **Version** 0.0.3
- **Since** 0.0.3
- **Access** public

Class OTS_Player

[line 22]

OTServ character abstraction.

OTServ character abstraction.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

void function OTS_Player::ban([$time = 0]) *[line 1648]*

Function Parameters:

- *int* **$time** Time for time until expires (0 - forever).

Bans current player.

Bans current player.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Player::delete() *[line 1702]*

Deletes player.

Deletes player.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.4+SVN
- **Since** 0.0.1
- **Access** public

void function OTS_Player::find($name) *[line 84]*

Function Parameters:

- *string* **$name** Player's name.

Loads player by it's name.

Loads player by it's name.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2
- **Access** public

OTS_Account function OTS_Player::getAccount() *[line 186]*

Returns account of this player.

Returns account of this player.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getCap() *[line 841]*

Capacity.

Capacity.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

mixed function OTS_Player::getConditions() *[line 955]*

Conditions.

Conditions.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

string function OTS_Player::getCustomField($field) *[line 1254]*

Function Parameters:

- *string* **$field** Field name.

Reads custom field.

Reads custom field.

Reads field by it's name. Can read any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3

- **Since** 0.0.1
- **Access** public

OTS_Item|*null* function OTS_Player::getDepot($depot) *[line 1529]*

Function Parameters:

- *int* **$depot** Depot ID to get items.

Returns items tree from given depot.

Returns items tree from given depot.

Note: OTS_Player class has no information about item types. It returns all items as OTS_Item, unless they have any contained items in database, so empty container will be instanced as OTS_Item object, not OTS_Container.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getDirection() *[line 571]*

Looking direction.

Looking direction.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.

- **Since** 0.0.1
- **Access** public

int function OTS_Player::getExperience() *[line 328]*

Experience points.

Experience points.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

OTS_Group function OTS_Player::getGroup() *[line 215]*

Returns group of this player.

Returns group of this player.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

string function OTS_Player::getGuildNick() *[line 1042]*

Guild nick.

Guild nick.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getHealth() *[line 409]*

Current HP.

Current HP.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getHealthMax() *[line 436]*

Maximum HP.

Maximum HP.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1

- **Access** public

int function OTS_Player::getId() *[line 142]*

Player ID.

Player ID.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLastIP() *[line 895]*

Last login IP.

Last login IP.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLastLogin() *[line 868]*

Last login timestamp.

Last login timestamp.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLevel() *[line 355]*

Experience level.

Experience level.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLookAddons() *[line 733]*

Addons.

Addons.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLookBody() *[line 598]*

Body color.

Body color.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLookFeet() *[line 625]*

Boots color.

Boots color.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLookHead() *[line 652]*

Hair color.

Hair color.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLookLegs() *[line 679]*

Legs color.

Legs color.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLookType() *[line 706]*

Outfit.

Outfit.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLossExperience() *[line 1165]*

Percentage of experience lost after dead.

Percentage of experience lost after dead.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLossMana() *[line 1192]*

Percentage of used mana lost after dead.

Percentage of used mana lost after dead.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getLossSkills() *[line 1219]*

Percentage of skills lost after dead.

Percentage of skills lost after dead.

- **Version** 0.0.3
- **Version** 0.0.1

- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getMagLevel() *[line 382]*

Magic level.

Magic level.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getMana() *[line 463]*

Current mana.

Current mana.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getManaMax() *[line 490]*

Maximum mana.

Maximum mana.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getManaSpent() *[line 517]*

Mana spent.

Mana spent.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

string function OTS_Player::getName() *[line 159]*

Player name.

Player name.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.

- **Since** 0.0.1
- **Access** public

int function OTS_Player::getPosX() *[line 760]*

X map coordinate.

X map coordinate.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getPosY() *[line 787]*

Y map coordinate.

Y map coordinate.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getPosZ() *[line 814]*

Z map coordinate.

Z map coordinate.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getPremiumEnd() *[line 245]*

Player's Premium Account expiration timestamp.

Player's Premium Account expiration timestamp.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

OTS_GuildRank|*null* function OTS_Player::getRank() *[line 1086]*

Assigned guild rank.

Assigned guild rank.

- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1

- **Access** public

int function OTS_Player::getRankId() *[line 1070]*

Guild rank ID.

Guild rank ID.

- **Version** 0.0.3
- **Version** 0.0.1
- **Deprecated** 0.0.4 Use getRank().
- **Since** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Access** public

int function OTS_Player::getRedSkullTime() *[line 982]*

Red skulled time remained.

Red skulled time remained.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getSex() *[line 274]*

Player gender.

Player gender.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getSkill($skill) *[line 1309]*
Function Parameters:

- *int* **$skill** Skill ID.

Returns player's skill.
Returns player's skill.

- **Version** 0.0.2
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.2
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getSkillTries($skill) *[line 1341]*
Function Parameters:

- *int* **$skill** Skill ID.

Returns player's skill's tries for next level.

Returns player's skill's tries for next level.

- **Version** 0.0.2
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.2
- **Since** 0.0.1
- **Access** public

OTS_Item|*null* function OTS_Player::getSlot($slot) *[line 1394]*

Function Parameters:

- *int* **$slot** Slot to get items.

Returns items tree from given slot.

Returns items tree from given slot.

Note: OTS_Player class has no information about item types. It returns all items as OTS_Item, unless they have any contained items in database, so empty container will be instanced as OTS_Item object, not OTS_Container.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getSoul() *[line 544]*

Soul points.

Soul points.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getTownId() *[line 1138]*

Residence town's ID.

Residence town's ID.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

int function OTS_Player::getVocation() *[line 301]*

Player proffesion.

Player proffesion.

- **Version** 0.0.3

- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

bool function OTS_Player::hasRedSkull() *[line 1009]*

Checks if player has red skull.

Checks if player has red skull.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

bool function OTS_Player::isBanned() *[line 1683]*

Checks if player is banned.

Checks if player is banned.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

bool function OTS_Player::isLoaded() *[line 101]*

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

bool function OTS_Player::isSaveSet() *[line 922]*

Checks if save flag is set.

Checks if save flag is set.

- **Version** 0.0.3
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.1
- **Access** public

void function OTS_Player::load($id) *[line 62]*

Function Parameters:

- *int* **$id** Player's ID.

Loads player with given id.

Loads player with given id.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::save() *[line 111]*

Saves player in database.

Saves player in database.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setAccount($account) *[line 203]*

Function Parameters:

- *OTS_Account* **$account** Owning account.

Assigns character to account.

Assigns character to account.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setCap($cap) *[line 856]*
Function Parameters:

- *int* **$cap** Capacity.

Sets capacity.
Sets capacity.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setConditions($conditions) *[line 970]*
Function Parameters:

- *mixed* **$conditions** Condition binary field.

Sets conditions.
Sets conditions.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setCustomField($field, $value) *[line 1284]*
Function Parameters:

- *string* **$field** Field name.

- *mixed* **$value** Field value.

Writes custom field.

Writes custom field.

Write field by it's name. Can write any field of given record that exists in database.

Note: You should use this method only for fields that are not provided in standard setters/getters (SVN fields). This method runs SQL query each time you call it so it highly overloads used resources.

Note: Make sure that you pass $value argument of correct type. This method determinates whether to quote field value. It is safe - it makes you sure that no unproper queries that could lead to SQL injection will be executed, but it can make your code working wrong way. For example: $object->setCustomField('foo', '1'); will quote 1 as as string ('1') instead of passing it as a integer.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setDepot($depot, [$item = null], [$pid = 0], [$depot_id = 0]) *[line 1584]*

Function Parameters:

- *int* **$depot** Depot ID to save items.
- *OTS_Item* **$item** Item (can be a container with content) for given depot. Leave this parameter blank to clear depot.
- *int* **$pid** Deprecated, not used anymore.
- *int* **$depot_id** Internal, for further use.

Sets depot content.

Sets depot content.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setDirection($direction) *[line 586]*

Function Parameters:

- *int* **$direction** Looking direction.

Sets looking direction.

Sets looking direction.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setExperience($experience) *[line 343]*

Function Parameters:

- *int* **$experience** Experience points.

Sets experience points.

Sets experience points.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setGroup($group) *[line 232]*

Function Parameters:

- *OTS_Group* **$group** Group to be a member.

Assigns character to group.

Assigns character to group.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setGuildNick($guildnick) *[line 1057]*

Function Parameters:

- *string* **$guildnick** Name.

Sets guild nick.

Sets guild nick.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setHealth($health) *[line 424]*
Function Parameters:

- *int* **$health** Current HP.

Sets current HP.
Sets current HP.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setHealthMax($healthmax) *[line 451]*
Function Parameters:

- *int* **$healthmax** Maximum HP.

Sets maximum HP.
Sets maximum HP.

- **Version** 0.0.1

- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLastIP($lastip) *[line 910]*
Function Parameters:

- *int* **$lastip** Last login IP.

Sets last login IP.
Sets last login IP.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLastLogin($lastlogin) *[line 883]*
Function Parameters:

- *int* **$lastlogin** Last login timestamp.

Sets last login timestamp.
Sets last login timestamp.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLevel($level) *[line 370]*
Function Parameters:

- *int* **$level** Experience level.

Sets experience level.
Sets experience level.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLookAddons($lookaddons) *[line 748]*
Function Parameters:

- *int* **$lookaddons** Addons.

Sets addons.
Sets addons.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLookBody($lookbody) *[line 613]*
Function Parameters:

- *int* **$lookbody** Body color.

Sets body color.

Sets body color.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLookFeet($lookfeet) *[line 640]*

Function Parameters:

- *int* **$lookfeet** Boots color.

Sets boots color.

Sets boots color.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLookHead($lookhead) *[line 667]*

Function Parameters:

- *int* **$lookhead** Hair color.

Sets hair color.

Sets hair color.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLookLegs($looklegs) *[line 694]*

Function Parameters:

- *int* **$looklegs** Legs color.

Sets legs color.

Sets legs color.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLookType($looktype) *[line 721]*

Function Parameters:

- *int* **$looktype** Outfit.

Sets outfit.

Sets outfit.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLossExperience($loss_experience) *[line 1180]*

Function Parameters:

- *int* **$loss_experience** Percentage of experience lost after dead.

Sets percentage of experience lost after dead.

Sets percentage of experience lost after dead.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setLossMana($loss_mana) *[line 1207]*

Function Parameters:

- *int* **$loss_mana** Percentage of used mana lost after dead.

Sets percentage of used mana lost after dead.

Sets percentage of used mana lost after dead.

- **Version** 0.0.1
- **Since** 0.0.1

- **Access** public

void function OTS_Player::setLossSkills($loss_skills) *[line 1234]*
Function Parameters:

- *int* **$loss_skills** Percentage of skills lost after dead.

Sets percentage of skills lost after dead.
Sets percentage of skills lost after dead.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setMagLevel($maglevel) *[line 397]*
Function Parameters:

- *int* **$maglevel** Magic level.

Sets magic level.
Sets magic level.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setMana($mana) *[line 478]*

Function Parameters:

- *int* **$mana** Current mana.

Sets current mana.

Sets current mana.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setManaMax($manamax) *[line 505]*

Function Parameters:

- *int* **$manamax** Maximum mana.

Sets maximum mana.

Sets maximum mana.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setManaSpent($manaspent) *[line 532]*

Function Parameters:

- *int* **$manaspent** Mana spent.

Sets mana spent.

Sets mana spent.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setName($name) *[line 174]*

Function Parameters:

- *string* **$name** Name.

Sets players's name.

Sets players's name.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setPosX($posx) *[line 775]*

Function Parameters:

- *int* **$posx** X map coordinate.

Sets X map coordinate.

Sets X map coordinate.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setPosY($posy) *[line 802]*

Function Parameters:

- *int* **$posy** Y map coordinate.

Sets Y map coordinate.

Sets Y map coordinate.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setPosZ($posz) *[line 829]*

Function Parameters:

- *int* **$posz** Z map coordinate.

Sets Z map coordinate.

Sets Z map coordinate.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setPremiumEnd($premend) *[line 262]*
Function Parameters:

- *int* **$premend** PACC expiration timestamp.

Sets player's Premium Account expiration timestamp.
Sets player's Premium Account expiration timestamp.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3
- **Access** public

void function OTS_Player::setRank([$guildRank = null]) *[line 1119]*
Function Parameters:

- *OTS_GuildRank*|*null* **$guildRank** Guild rank (null to clear assign).

Assigns guild rank.
Assigns guild rank.

- **Version** 0.0.1

- **Since** 0.0.1
- **Access** public

void function OTS_Player::setRankId($rank_id) *[line 1109]*

Function Parameters:

- *int* **$rank_id** Guild rank ID.

Sets guild rank ID.

Sets guild rank ID.

- **Version** 0.0.1
- **Deprecated** 0.0.4 Use setRank().
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setRedSkull() *[line 1030]*

Sets red skull flag.

Sets red skull flag.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setRedSkullTime($redskulltime) *[line 997]*

Function Parameters:

- *int* **$redskulltime** Red skulled time remained.

Sets red skulled time remained.

Sets red skulled time remained.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setSave() *[line 943]*

Sets save flag.

Sets save flag.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setSex($sex) *[line 289]*

Function Parameters:

- *int* **$sex** Player gender.

Sets player gender.

Sets player gender.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setSkill($skill, $value) *[line 1327]*

Function Parameters:

- *int* **$skill** Skill ID.
- *int* **$value** Skill value.

Sets skill value.

Sets skill value.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2
- **Access** public

void function OTS_Player::setSkillTries($skill, $tries) *[line 1359]*

Function Parameters:

- *int* **$skill** Skill ID.
- *int* **$tries** Skill tries.

Sets skill's tries for next level.

Sets skill's tries for next level.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2
- **Access** public

void function OTS_Player::setSlot($slot, [$item = null], [$pid = 0]) *[line 1448]*
Function Parameters:

- *int* **$slot** Slot to save items.
- *OTS_Item* **$item** Item (can be a container with content) for given slot. Leave this parameter blank to clear slot.
- *int* **$pid** Deprecated, not used anymore.

Sets slot content.
Sets slot content.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Throws** E_OTS_NotLoaded If player is not loaded.
- **Since** 0.0.3
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setSoul($soul) *[line 559]*
Function Parameters:

- *int* **$soul** Soul points.

Sets soul points.
Sets soul points.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setTownId($town_id) *[line 1153]*
Function Parameters:

- *int* **$town_id** Residence town's ID.

Sets residence town's ID.
Sets residence town's ID.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::setVocation($vocation) *[line 316]*
Function Parameters:

- *int* **$vocation** Player proffesion.

Sets player proffesion.

Sets player proffesion.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::unban() *[line 1665]*

Deletes ban from current player.

Deletes ban from current player.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_Player::unsetRedSkull() *[line 1022]*

Unsets red skull flag.

Unsets red skull flag.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function OTS_Player::unsetSave() *[line 935]*

Unsets save flag.

Unsets save flag.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

array function OTS_Player::__sleep() *[line 51]*

Magic PHP5 method.

Magic PHP5 method.
Allows object serialisation.

- **Version** 0.0.4
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4
- **Access** public

Class OTS_Players_List

[line 21]

List of players.

List of players.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.1

void function OTS_Players_List::deletePlayer($player) *[line 30]*

Function Parameters:

- *OTS_Player* **$player** Player to be deleted.

Deletes player.

Deletes player.

- **Version** 0.0.4+SVN
- **Deprecated** 0.0.4+SVN Use OTS_Player->delete().
- **Since** 0.0.1
- **Access** public

void function OTS_Players_List::init() *[line 43]*

Sets list parameters.

Sets list parameters.

This method is called at object creation.

- **Version** 0.0.4+SVN
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_SQLField

[line 20]

SQL identifier representation.

SQL identifier representation.

- **Package** POT
- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

Constructor *void* function OTS_SQLField::__construct($name, [$table = '']) *[line 41]*

Function Parameters:

- *string* **$name** Field name.
- *string* **$table** Table name.

Creates new field representation.

Creates new field representation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

string function OTS_SQLField::getName() *[line 52]*

Returns field name.

Returns field name.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

string function OTS_SQLField::getTable() *[line 62]*

Returns table name.

Returns table name.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

Class OTS_SQLFilter

[line 20]

SQL WHERE clause object.

SQL WHERE clause object.

- **Package** POT
- **Version** 0.0.4+SVN

- **Since** 0.0.4+SVN

OTS_SQLFilter::CRITERIUM_AND

= 1 *[line 58]*

AND sibling.

AND sibling.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::CRITERIUM_OR

= 2 *[line 62]*

OR sibling.

OR sibling.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_EQUAL

= 1 *[line 25]*

Equal operator.

Equal operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_GREATER

= 3 *[line 33]*

Greater-then operator.

Greater-then operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_LIKE

= 7 *[line 49]*

LIKE operator.

LIKE operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_LOWER

= 2 *[line 29]*

Lower-then operator.

Lower-then operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_NEQUAL

= 4 *[line 37]*

Not-equal operator.

Not-equal operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_NGREATER

= 6 *[line 45]*

Not-greater-then operator.

Not-greater-then operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_NLIKE

= 8 *[line 53]*

Not-LIKE operator.

Not-LIKE operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

OTS_SQLFilter::OPERATOR_NLOWER

= 5 *[line 41]*

Not-lower-then operator.

Not-lower-then operator.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

void function OTS_SQLFilter::addFilter($left, [$right = null], [$operator = self::OPERATOR_EQUAL], [$criterium = self::CRITERIUM_AND]) *[line 238]*

Function Parameters:

- *mixed* **$left** Left side (OTS_SQLField class object, or literal value).
- *mixed* **$right** Right side (OTS_SQLField class object, or literal value).
- *int* **$operator** Operator used for comparsion (equal check by default).
- *int* **$criterium** Criterium merging method (AND by default).

General-purpose filter.

General-purpose filter.
Appends new filter in universal way.

To append subset of another filters us addFilter($OTS_SQLFilterObject).

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

void function OTS_SQLFilter::compareField($field, $value, [$operator = self::OPERATOR_EQUAL], [$criterium = self::CRITERIUM_AND]) *[line 251]*

Function Parameters:

- *string* **$field** Field name.
- *mixed* **$value** Literal value.
- *int* **$operator** Operator used for comparsion (equal by default).
- *int* **$criterium** Criterium merging method (AND by default).

Compares field with a literal value.

Compares field with a literal value.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

array function OTS_SQLFilter::getTables() *[line 263]*

Returns list of all tables used by filter.

Returns list of all tables used by filter.
This is required for FROM clause.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN

- **Access** public

array function OTS_SQLFilter::__sleep() *[line 79]*

Magic PHP5 method.

Magic PHP5 method.

Allows object serialisation.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

string function OTS_SQLFilter::__toString() *[line 92]*

Returns string representation of WHERE clause.

Returns string representation of WHERE clause.

Returned string can be easily inserted into SQL query.

- **Version** 0.0.4+SVN
- **Since** 0.0.4+SVN
- **Access** public

Class POT

[line 30]

Main POT class.

Main POT class.

- **Package** POT
- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

POT::BAN_ACCOUNT

= 3 *[line 261]*

Account ban.

Account ban.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN

POT::BAN_IP

= 1 *[line 247]*

IP ban.

IP ban.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1

- **Since** 0.0.4+SVN

POT::BAN_PLAYER

= 2 *[line 254]*

Player ban.

Player ban.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN

POT::DB_MYSQL

= 1 *[line 35]*

MySQL driver.

MySQL driver.

- **Version** 0.0.1
- **Since** 0.0.1

POT::DB_ODBC

= 4 *[line 53]*

ODBC driver.

ODBC driver.

- **Version** 0.0.4
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4

POT::DB_PGSQL

= 3 *[line 46]*

PostgreSQL driver.

PostgreSQL driver.

- **Version** 0.0.4
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4

POT::DB_SQLITE

= 2 *[line 39]*

SQLite driver.

SQLite driver.

- **Version** 0.0.1
- **Since** 0.0.1

POT::DEPOT_SID_FIRST

= 100 *[line 239]*

First depot item sid.

First depot item sid.

- **Version** 0.0.4
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4

POT::DIRECTION_EAST

= 1 *[line 102]*

East.

East.

- **Version** 0.0.1
- **Since** 0.0.1

POT::DIRECTION_NORTH

= 0 *[line 98]*

North.

North.

- **Version** 0.0.1

- **Since** 0.0.1

POT::DIRECTION_SOUTH

= 2 *[line 106]*

South.
South.

- **Version** 0.0.1
- **Since** 0.0.1

POT::DIRECTION_WEST

= 3 *[line 110]*

West.
West.

- **Version** 0.0.1
- **Since** 0.0.1

POT::ORDER_ASC

= 1 *[line 269]*

Ascencind sorting order.
Ascencind sorting order.

- **Version** 0.0.4+SVN

- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN

POT::ORDER_DESC

= 2 *[line 276]*

Descending sorting order.

Descending sorting order.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN

POT::SEX_FEMALE

= 0 *[line 58]*

Female gender.

Female gender.

- **Version** 0.0.1
- **Since** 0.0.1

POT::SEX_MALE

= 1 *[line 62]*

Male gender.
Male gender.

- **Version** 0.0.1
- **Since** 0.0.1

POT::SKILL_AXE

= 3 *[line 139]*

Axe fighting.
Axe fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2

POT::SKILL_CLUB

= 1 *[line 125]*

Club fighting.
Club fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1

- **Since** 0.0.2

POT::SKILL_DISTANCE

= 4 *[line 146]*

Distance fighting.

Distance fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2

POT::SKILL_FISHING

= 6 *[line 160]*

Fishing.

Fishing.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2

POT::SKILL_FIST

= 0 *[line 118]*

Fist fighting.

Fist fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2

POT::SKILL_SHIELDING

= 5 *[line 153]*

Shielding.

Shielding.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2

POT::SKILL_SWORD

= 2 *[line 132]*

Sword fighting.

Sword fighting.

- **Version** 0.0.2
- **Version** 0.0.1

- **Since** 0.0.1
- **Since** 0.0.2

POT::SLOT_AMMO

= 10 *[line 231]*

Ammunition slot.

Ammunition slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_ARMOR

= 4 *[line 189]*

Armor slot.

Armor slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_BACKPACK

= 3 *[line 182]*

Backpack slot.

Backpack slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_FEET

= 8 *[line 217]*

Boots slot.

Boots slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_HEAD

= 1 *[line 168]*

Head slot.

Head slot.

- **Version** 0.0.3

- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_LEFT

= 6 *[line 203]*

Left hand slot.

Left hand slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_LEGS

= 7 *[line 210]*

Legs slot.

Legs slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_NECKLACE

= 2 *[line 175]*

Necklace slot.
Necklace slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_RIGHT

= 5 *[line 196]*

Right hand slot.
Right hand slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::SLOT_RING

= 9 *[line 224]*

Ring slot.
Ring slot.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.3

POT::VOCATION_DRUID

= 2 *[line 81]*

Druid.
Druid.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Vocations are now loaded dynamicly from vocations.xml file.
- **Since** 0.0.1

POT::VOCATION_KNIGHT

= 4 *[line 93]*

Knight.
Knight.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Vocations are now loaded dynamicly from vocations.xml file.
- **Since** 0.0.1

POT::VOCATION_NONE

= 0 *[line 69]*

None vocation.

None vocation.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Vocations are now loaded dynamicly from vocations.xml file.
- **Since** 0.0.1

POT::VOCATION_PALADIN

= 3 *[line 87]*

Paladin.

Paladin.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Vocations are now loaded dynamicly from vocations.xml file.
- **Since** 0.0.1

POT::VOCATION_SORCERER

= 1 *[line 75]*

Sorcerer.

Sorcerer.

- **Version** 0.0.1
- **Deprecated** 0.0.4+SVN Vocations are now loaded dynamicly from vocations.xml file.
- **Since** 0.0.1

void function POT::banIP($ip, [$mask = '255.255.255.255'], [$time = 0]) *[line 617]*

Function Parameters:

- *string* **$ip** IP to ban.
- *string* **$mask** Mask for ban (by default bans only given IP).
- *int* **$time** Time for time until expires (0 - forever).

Bans given IP number.

Bans given IP number.

Adds IP/mask ban. You can call this function with only one parameter to ban only given IP address without expiration.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

void function POT::connect($driver, $params) *[line 399]*

connect.php

```
1     <?php
2
3     /**
4      * @ignore
5      * @package examples
6      * @author Wrzasq <wrzasq@gmail.com>
7      * @copyright 2007 (C) by Wrzasq
8      * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9      */
10
11    // includes POT main file
12    include('../classes/OTS.php');
13
14    // you can easily store such structure in config.php
15    $config = array(
16        'driver' =>    POT::DB_MYSQL,
17        'prefix' =>    '',
18        'host' =>    'localhost',
```

```
19        'user' =>    'wrzasq',
20        'password' =>    '',
21        'database' =>    'otserv'
22    );
23
24    // connects to database
25    $ots = POT::getInstance();
26    $ots->   connect(null, $config);
27    // could be: $ots->connect(POT::DB_MYSQL, $config);
28
29    ?>
```

Function Parameters:

- *int|null* **$driver** Database driver type.
- *array* **$params** Connection info.

Connects to database.

Connects to database.

Creates OTServ database connection object.

First parameter is one of database driver constants values. Currently MySQL, SQLite, PostgreSQL and ODBC drivers are supported.

This parameter can be null, then you have to specify *'driver'* parameter.

Such way is comfortable to store entire database configuration in one array and possibly runtime evaluation and/or configuration file saving.

For parameters list see driver documentation. Common parameters for all drivers are:

- *driver* - optional, specifies driver, aplies when *$driver* method parameter is *null*
- *prefix* - optional, prefix for database tables, use if you have more then one OTServ installed on one database.

- **Version** 0.0.4
- **Version** 0.0.1
- **Throws** Exception When driver is not supported.
- **Since** 0.0.1
- **Access** public
- **Example**

OTS_SQLFilter function POT::createFilter() *[line 709]*

Creates lists filter.

Creates lists filter.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

IOTS_DAO function POT::createObject($class) *[line 450]*

Function Parameters:

- *string* **$class** Class name.

Creates OTServ DAO class instance.

Creates OTServ DAO class instance.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

PDO function POT::getDBHandle() *[line 525]*

Returns database connection handle.

Returns database connection handle.

At all you shouldn't use this method and work with database using POT classes, but it may be sometime necessary to use direct database access (mainly until POT won't provide many important features).

It is also important as serialised objects after unserialisation needs to be re-initialised with database connection.

- **Version** 0.0.4
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4
- **Access** public

POT function POT::getInstance() *[line 283]*

Singleton.

Singleton.

- **Version** 0.0.1
- **Static**
- **Since** 0.0.1
- **Access** public

int|*bool* function POT::getVocationID($name) *[line 569]*

Function Parameters:

- *string* **$name** Vocation.

Returns vocation's ID.

Returns vocation's ID.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

string|bool function POT::getVocationName($id) *[line 582]*

Function Parameters:

- *int* **$id** Vocation ID.

Returns name of given vocation's ID.

Returns name of given vocation's ID.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

array function POT::getVocationsList() *[line 601]*

Returns list (id => name) of loaded vocations.

Returns list (id => name) of loaded vocations.

- **Version** 0.0.4+SVN
- **Version** 0.0.1

- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

bool function POT::isIPBanned($ip) *[line 685]*

Function Parameters:

- *string* **$ip** IP to ban.

Checks if given IP is banned.

Checks if given IP is banned.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

void function POT::loadClass($class) *[line 358]*

Function Parameters:

- *string* **$class** Class name.

Loads POT class file.

Loads POT class file.

Runtime class loading on demand - usefull for __autoload() function.

Note: Since 0.0.2 version this function is suitable for spl_autoload_register().

Note: Since 0.0.3 version this function handles also exceptions.

- **Version** 0.0.3
- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public

void function POT::loadVocations($file) *[line 548]*

Function Parameters:

- *string* **$file** vocations.xml file location.

Loads vocations list.

Loads vocations list.

Loads vocations list from given file.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

OTS_InfoRespond|bool function POT::serverStatus($server, $port) *[line 468]*

example

```
1      <?php
2
3      /**
4       * @ignore
5       * @package examples
6       * @author Wrzasq <wrzasq@gmail.com>
7       * @copyright 2007 (C) by Wrzasq
8       * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
```

```
9       */
10
11      // to not repeat all that stuff
12      include('quickstart.php');
13
14      // server and port
15      $server = '127.0.0.1';
16      $port = 7171;
17
18      // queries server of status info
19      $status = $ots->   serverStatus($server, $port);
20
21      // offline
22      if(!$status)
23      {
24          echo 'Server ', $server, ' is offline.', "\n"          ;
25      }
26      // displays various info
27      else
28      {
29          echo 'Server name: ', $status->   getName(), "\n"          ;
30          echo 'Server owner: ', $status->   getOwner(), "\n"          ;
31          echo 'Players online: ', $status->   getOnlinePlayers(), "\n"          ;
32          echo 'Maximum allowed number of players: ', $status->   getMaxPlayers(), "\n"          ;
33          echo 'Required client version: ', $status->   getClientVersion(), "\n"          ;
34          echo 'All monsters: ', $status->   getMonstersCount(), "\n"          ;
35          echo 'Server message: ', $status->   getMOTD(), "\n"          ;
36      }
37
38      ?>
```

Function Parameters:

- *string* **$server** Server IP/domain.
- *int* **$port** OTServ port.

Queries server status.

Queries server status.
Sends 'info' packet to OTS server and return output.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.2
- **Access** public
- **Example**

void function POT::setPOTPath($path) *[line 314]*

```php
<?php

/**
 * @ignore
 * @package examples
 * @author Wrzasq <wrzasq@gmail.com>
 * @copyright 2007 (C) by Wrzasq
 * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
 */

// this is the way you should work with POT if you moved main OTS.php file outside POT's directory
include('path/to/OTS.php');

// dont use 'new POT()'!!!
$ots = POT::getInstance();
$ots->   setPOTPath('../classes/');

/*
    here comes your stuff...
*/

?>
```

Function Parameters:

- *string* **$path** POT files path.

Set POT directory.

Set POT directory.

Use this method if you keep your POT package in different directory then this file.

- **Version** 0.0.1
- **Since** 0.0.1
- **Access** public
- **Example**

void function POT::unbanIP($ip, [$mask = '255.255.255.255']) *[line 652]*

Function Parameters:

- *string* **$ip** IP to ban.
- *string* **$mask** Mask for ban (by default 255.255.255.255).

Deletes ban from given IP number.

Deletes ban from given IP number.
Removes given IP/mask ban.

- **Version** 0.0.4+SVN
- **Version** 0.0.1
- **Since** 0.0.1
- **Since** 0.0.4+SVN
- **Access** public

compat.php

POT compatibility assurance package.

POT compatibility assurance package.

This package makes you sure that POT scripts won't cause FATAL errors on PHP older PHP 5.x versions. However remember that some PHP features won't be enabled with it. For example if you have PHP 5.0.x, this package will define Countable interface for you so PHP will know it, but it won't allow you to use count($countableObject) structure.

- **Package** POT
- **Sub-Package** compat
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.2
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.2
- **License** GNU Lesser General Public License, Version 3

Appendices

Appendix A - Class Trees

Package POT

E_OTS_NoDriver

- Exception
  - E_OTS_NoDriver

E_OTS_NotLoaded

- Exception
  - E_OTS_NotLoaded

IOTS_DAO

- IOTS_DAO

IOTS_DB

- IOTS_DB

IOTS_GuildAction

- IOTS_GuildAction

OTS_Base_DAO

- OTS_Base_DAO

- OTS_Account
- OTS_Group
- OTS_Guild
- OTS_GuildRank
- OTS_Player
- OTS_SQLFilter

OTS_Base_List

- OTS_Base_List
  - OTS_Accounts_List
  - OTS_Groups_List
  - OTS_GuildRanks_List
  - OTS_Guilds_List
  - OTS_Players_List

OTS_DB_MySQL

- PDO
  - OTS_DB_MySQL

OTS_DB_ODBC

- PDO
  - OTS_DB_ODBC

OTS_DB_PostgreSQL

- PDO
  - OTS_DB_PostgreSQL

OTS_DB_SQLite

- PDO
  - OTS_DB_SQLite

OTS_InfoRespond

- DOMDocument
  - OTS_InfoRespond

OTS_Item

- OTS_Item
  - OTS_Container

OTS_SQLField

- OTS_SQLField

POT

- POT

Appendix B - README/CHANGELOG/INSTALL

CHANGELOG

[0.0.4+SVN]
* Added support for vocations.xml file. <wrzasq>
* Added support for bans. <wrzasq>
* Added sorting and filtering for lists. <wrzasq>
* Code grouped into base classes. <wrzasq>
* Some code optimalisation. <wrzasq>
* Fixed typos. <wrzasq>

[0.0.4]
* Added guild system support (guilds, ranks, invitations and requests drivers mechanisms). <wrzasq>
* Added account group support. <wrzasq>
* Added support for depot_id field (it is reserverd in OTServ for futher use). <wrzasq>
* Added PostgreSQL and ODBC drivers. <wrzasq>
* Added __sleep() and __wakeup() methods to allow POT objects to be stored in sessions. <wrzasq>
* Added __clone() methods to allow save ID-losing cloning of POT objects. <wrzasq>
* Added __set_state() methods. <wrzasq>
* Updated players table structure. <wrzasq>
* Dropped REGEXP operator bindings - not used anywhere. <wrzasq>
* Fixed items loading and saving. <wrzasq>
* Fixed typos. <wrzasq>

[0.0.3]
* Added custom fields support. <wrzasq>
* Added items and depots support. <wrzasq>
* Added support for players PACC timestamps. <wrzasq>
* Fixed loading skills. <wrzasq>
* Replaced E_USER_* with exceptions. <wrzasq>
* Uses fetchAll() in loops to prevent MySQL buffering problems. <wrzasq>
* Restricted access to POT class constructor to make sure it won't be instanced directly. <wrzasq>

[0.0.2]
* Added "compat" library for POT. <wrzasq>
* Added skills support in OTS_Player class. <wrzasq>
* Added 'info' serverStatus() method and respond handler for server status protocol. <wrzasq>
* Fixed `redskulltime` field name in OTS_Player. <wrzasq>
* Fixed 'password' parameter for DB_MYSQL driver. <wrzasq>
* Added find() to OTS_Account class to load accounts by their's e-mail addresses. <wrzasq>
* POT class now automaticly binds own __autoload() handler with spl_autoload_register(). <wrzasq>

[0.0.1]
* Initial release. <wrzasq>

README

POT (PHP OTServ Toolkit) is a PHP toolkit for scripts that work with OTServ database.

===== About =====

This toolkit provides a way for PHP programmers that don't know SQL langauge to work with OTServ database.

For installation help check INSTALL file.

For usage tutorial/API documentation check http://www.otserv-aac.info/pot/ or documentation.pdf file.

===== Contact =====

In case of any contact needed, please use following e-mail address: wrzasq@gmail.com.

===== Files =====

classes/ - POT class files.
examples/ - example files for learning.
tutorials/ - phpDocumentor directory.
CHANGELOG - changes history.
INSTALL - installation tutorial.
LICENSE - POT license (GNU LGPL v3), if you don't accept it - don't use any of those scripts.
NEWS - changes in current release.
README - this readme file.
RULES - rules to be followed during developing contributed code.
Makefile - make input, for documentation generation.
documentation.pdf - phpDocumentor-generater documentation in PDF format.
compat.php - Compatibility assurance library.
test.php - phpUnit test suite.

===== Makefile =====

Makefile contains some targets for make that can help in development. Makefile requires following command-line commands:

php: PHP CLI interface.
phpdoc: phpDocumentor.
phpunit: PHPUnit testing framework.

Possible targets:

all: default one, runs all other targets (in order: clean, check, documentation, pdf, online, test, package).
clean: deletes documentation.
check: checks syntax of all PHP files.
documentation: generates HTML documentation.
pdf: generates PDF documentation.
online: OTServ-AAC website documentation template used.
test: runs test suite.
package: creates pot.tar.gz file for distribution purposes.

For more readable output of phpUnit test run:
php test.php

===== Credits =====

* Wrzasq <wrzasq@gmail.com> - project initiator, main developer.

For more info see AUTHORS file in OTServ tree.

INSTALL

POT is a toolkit which means you don't literaly install it. You copy it's files and write code for it. All source files are located in classes/ subdirectory. Copy them to your script directory.

You can put main file - OTS.php in different directory then other files.

For information about how to include POT in your code see the documentation.

NEWS

What's new in 0.0.4 version?

* Added guild system support (guilds, ranks, invitations and requests drivers mechanisms).

  Main feature of new update - includes full support for guilds system.

* Added account group support.

  Support for new accounts table structure.

* Added support for depot_id field (it is reserverd in OTServ for futher use).

  Even thought depot_id field in OTServ database is not used, it was moved back as it is reserved for further use. POT also supports it.

* Added PostgreSQL and ODBC drivers.

  POT supports new revdbsys database drivers.

* Added __sleep() and __wakeup() methods to allow POT objects to be stored in sessions.

  Allows you to store POT objects in sessions.

* Added __clone() methods to allow save ID-losing cloning of POT objects.

  Allows you cloning of POT objects while deleting their's IDs for re-inserting.

* Added __set_state() methods.

  Magic PHP5 method - you can read more in PHP Manual.

* Updated players table structure.

  As always POT keeps your scripts up-to-date with support for latest structure of database.

* Dropped REGEXP operator bindings - not used anywhere.

  This internal feature wasn't used anywhere so it would be a waste to keep it out there.

* Fixed items loading and saving.

  Items saving (both body slots and depot items) in previous versions was coded wrong way. Fixed now.

* Fixed typos.

  Like always there were some bugs which of course were fixed.

Appendix D - Todo List

In Package POT

In **OTS.php**:

- 0.0.6: Spawns support (OTBM support -> cache).
- 0.1.0: Get rid of POT::getInstance()->create*() calls - use POT::getInstance()->getDBHandle() in constructors.
- 0.1.0: Implement __get()/__set()/__call()/__toString(); ArrayAccess interface.
- 0.1.0: Items list (items.xml + items.otb -> cache).
- 1.0.0: Complete phpUnit test.
- 1.0.0: More detailed documentation.

Index

A

C

D

E

G

I

N

O

P

Q

R

S