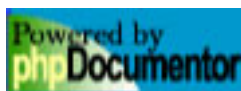


POT



Contents

POT	1
POT class preview	3
Quick start	5
Package POT Procedural Elements	10
IOTS DAO.php	10
IOTS DB.php	11
OTS.php	12
OTS Account.php	13
OTS Accounts List.php	14
OTS DB MySQL.php	15
OTS DB SQLite.php	16
OTS Group.php	17
OTS Groups List.php	18
OTS Player.php	19
OTS Players List.php	20
OTS SQLite Results.php	21
Package POT Classes	22
Class IOTS DAO	22
Constructor construct	22
Class IOTS DB	23
Constructor construct	23
Method fieldName	23
Method lastInsertId	24
Method limit	24
Method SQLquery	25
Method SQLquote	25
Method tableName	25
Class OTS Account	26
Constructor construct	26
Method block	27
Method create	27
example: account.php	27
Method getEmail	28
Method getId	28
Method getPACCDays	29
Method getPassword	29
Method getPlayers	29
Method isBlocked	30
Method isLoaded	30
Method load	30
Method save	31
Method setEmail	31

Method setPACCDays	31
Method setPassword	32
Method unblock	32
Class OTS Accounts List	33
Constructor construct	33
Method count	33
Method current	34
Method deleteAccount	34
Method key	34
Method next	35
Method rewind	35
Method setLimit	35
Method setOffset	36
Method valid	36
Class OTS DB MySQL	36
Constructor construct	37
Method fieldName	37
Method limit	38
Method SQLquery	38
Method SQLquote	39
Method tableName	39
Class OTS DB SQLite	40
Constructor construct	40
Method fieldName	40
Method limit	41
Method regexp	41
Method SQLquery	42
Method SQLquote	42
Method tableName	43
Class OTS Group	43
Constructor construct	43
Method getAccess	44
Method getFlags	44
Method getIdx	44
Method getMaxDepotItems	45
Method getMaxVIPList	45
Method getName	45
Method getPlayers	46
Method isLoaded	46
Method load	46
Method save	47
Method setAccess	47
Method setFlags	47
Method setMaxDepotItems	48
Method setMaxVIPList	48
Method setName	49
Class OTS Groups List	49
Constructor construct	49
Method count	50

Method current	50
Method deleteGroup	50
Method key	51
Method next	51
Method rewind	51
Method setLimit	52
Method setOffset	52
Method valid	53
Class OTS_Player	53
Constructor construct	53
Method find	54
Method getAccount	54
Method getCap	54
Method getConditions	55
Method getDirection	55
Method getExperience	55
Method getGroup	56
Method getGuildNick	56
Method getHealth	56
Method getHealthMax	56
Method getId	57
Method getLastIP	57
Method getLastLogin	57
Method getLevel	58
Method getLookAddons	58
Method getLookBody	58
Method getLookFeet	58
Method getLookHead	59
Method getLookLegs	59
Method getLookType	59
Method getLossExperience	60
Method getLossMana	60
Method getLossSkills	60
Method getMagLevel	60
Method getMana	61
Method getManaMax	61
Method getManaSpent	61
Method getName	62
Method getPosX	62
Method getPosY	62
Method getPosZ	62
Method getRankId	63
Method getRedSkullTime	63
Method getSex	63
Method getSoul	64
Method getTownId	64
Method getVocation	64
Method hasRedSkull	65
Method isLoaded	65

Method isSaveSet	65
Method load	65
Method save	66
Method setAccount	66
Method setCap	67
Method setConditions	67
Method setDirection	67
Method setExperience	68
Method setGroup	68
Method setGuildNick	69
Method setHealth	69
Method setHealthMax	69
Method setLastIP	70
Method setLastLogin	70
Method setLevel	71
Method setLookAddons	71
Method setLookBody	72
Method setLookFeet	72
Method setLookHead	72
Method setLookLegs	73
Method setLookType	73
Method setLossExperience	74
Method setLossMana	74
Method setLossSkills	74
Method setMagLevel	75
Method setMana	75
Method setManaMax	76
Method setManaSpent	76
Method setName	77
Method setPosX	77
Method setPosY	77
Method setPosZ	78
Method setRankId	78
Method setRedSkull	79
Method setRedSkullTime	79
Method setSave	79
Method setSex	80
Method setSoul	80
Method setTownId	80
Method setVocation	81
Method unsetRedSkull	81
Method unsetSave	82
Class OTS_Players_List	82
Constructor construct	82
Method count	83
Method current	83
Method deletePlayer	83
Method key	84
Method next	84

Method rewind	84
Method setLimit	85
Method setOffset	85
Method valid	85
Class POT	86
Class Constant DB_MYSQL	86
Class Constant DB_SQLITE	86
Class Constant DIRECTION_EAST	87
Class Constant DIRECTION_NORTH	87
Class Constant DIRECTION_SOUTH	87
Class Constant DIRECTION_WEST	88
Class Constant SEX_FEMALE	88
Class Constant SEX_MALE	88
Class Constant VOCATION_DRUID	89
Class Constant VOCATION_KNIGHT	89
Class Constant VOCATION_NONE	89
Class Constant VOCATION_PALADIN	89
Class Constant VOCATION_SORCERER	90
Constructor __construct	90
Method connect	90
example: connect.php	90
Method createObject	92
Method getInstance	92
Method loadClass	92
example: autoload.php	92
Method setPOTPath	93
example: fakeroot.php	93
Appendices	95
Appendix A - Class Trees	96
POT	96
Appendix B - README/CHANGELOG/INSTALL	98
CHANGELOG	99
INSTALL	99
NEWS	99
README	99

POT

This is documentation of POT - official toolkit for [OTServ AAC scripts](#).

PHP OTServ Toolkit

There are several purposes 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 won't work with old database structure as well as with broken database - it relies on database foreign key constraints, 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 information 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 doesn't 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 really 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 enough. If you have your own `__autoload()` mechanism you won't be able to just include example codes - you would need to redefine `__autoload()` function, which PHP doesn't 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).

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 globally 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](#). You can combine [POT class loading mechanism](#) with it. For 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 // includes POT main file
12 include( './classes/OTS.php' );
13
14 function __autoload( $class )
15 {
16     // checks if it's POT class
17     if( preg_match( '/^OTS_/', $class ) != 0 )
18     {
19         POT::getInstance\(\)-> loadClass( $class );
20     }
21 }
22 // possibly call your own __autoload() handler
23 else
24 {
25     here comes your stuff...
26 }
27 */
28 }
29
30 ?>
```

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 (we also encourage you to bind [POT classes loading mechanism](#) to `__autoload()` function. Here is a startup code 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 // includes POT main file
12 include( './classes/OTS.php' );
13
14 // for further POT classes
15 function __autoload( $class )
16 {
17     // checks if it's POT class
18     if( preg_match( '/^?OTS_/', $class ) != 0 )
19     {
20         POT::getInstance()-> loadClass( $class );
21     }
22 }
23 // possibly call your own __autoload() handler
24 else
25 {
26     here comes your stuff...
27 }
28 */
29 }
30
31 // database configuration - can be simply moved to external file, eg. config.php
32 $config= array(
33     'driver' => POT::DB_MYSQL,
34     'host' => 'localhost',
35     'user' => 'wrzasq',
36     'database' => 'otserv'
37 );
38
39 // creates POT instance (or get existing one)
40 $ots= POT::getInstance();
41 $ots-> connect( null, $config );
42
43 ?>
```

Account creation

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

```
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_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 retrieving
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. Through 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 through 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 ?>
```


Package POT Procedural Elements

IOTS_DAO.php

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

IOTS_DB.php

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

OTS_Account.php

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

OTS_Accounts_List.php

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

OTS_DB_MySQL.php

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

OTS_DB_SQLite.php

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

OTS_Group.php

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

OTS_Groups_List.php

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

OTS_Player.php

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

OTS_Players_List.php

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

Package POT Classes

Class IOTS_DAO

[line 21]

OTserv database object.

OTserv database object.

This interface indicates that class is a OTServ DAO class.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function IOTS_DAO::__construct(\$db) *[line 28]*

Function Parameters:

- [*IOTS_DB*](#) **\$db** Database connection object.

DAO objects must be initialized with a database.

DAO objects must be initialized with a database.

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

Class IOTS_DB

[line 21]

OTServ database handler interface.

OTServ database handler interface.

This interface specifies routines requires by DAO classes.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function IOTS_DB::__construct(\$params) [line 28]

Function Parameters:

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

Connection parameters.

Connection parameters.

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

string function IOTS_DB::fieldName(\$name) [line 36]

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

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

int function IOTS_DB::lastInsertId() [*line 63*]

ID of last created record.

ID of last created record.

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

string function IOTS_DB::limit([\$limit = false], [\$offset = false]) [*line 71*]

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
- **Access** public

mixed function IOTS_DB::SQLquery(\$query) [*line 57*]

Function Parameters:

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

Evaluates query.

Evaluates query.

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

string function IOTS_DB::SQLquote(\$value) [*line 50*]

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
- **Access** public

string function IOTS_DB::tableName(\$name) [*line 43*]

Function Parameters:

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

Query-quoted table name.

Query-quoted table name.

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

Class OTS_Account

[line 19]

OTServ account abstraction.

OTServ account abstraction.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_Account::__construct(\$db) [line 40]

Function Parameters:

- [*IOTS_DB*](#) \$db Database connection object.

Sets database connection handler.

Sets database connection handler.

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

void function OTS_Account::block() [*line 242*]

Blocks account.

Blocks account.

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

int function OTS_Account::create([\$min = 1], [\$max = 9999999]) [*line 60*]

account.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     // 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     ?>

```

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!

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

string|bool function OTS_Account::getEmail() [*line 194*]

E-mail address.

E-mail address.

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

int|bool function OTS_Account::getId() [*line 152*]

Account number.

Account number.

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

int/bool function OTS_Account::getPACCDays() [*line 252*]

PACC days.

PACC days.

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

string/bool function OTS_Account::getPassword() [*line 168*]

Account's password.

Account's password.

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

array/bool function OTS_Account::getPlayers() [*line 278*]

List of characters on account.

List of characters on account.

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

bool|null function OTS_Account::isBlocked() [*line 220*]

Checks if account is blocked.

Checks if account is blocked.

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

bool function OTS_Account::isLoading() [*line 123*]

Checks if object is loaded.

Checks if object is loaded.

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

void function OTS_Account::load(\$id) [*line 112*]

Function Parameters:

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

Loads account with given number.

Loads account with given number.

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

bool function OTS_Account::save() [*line 133*]

Updates account in database.

Updates account in database.

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

void function OTS_Account::setEMail(\$email) [*line 210*]

Function Parameters:

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

Sets account's email.

Sets account's email.

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

void function OTS_Account::setPACCDays(\$premdays, \$pacc) [*line 268*]

Function Parameters:

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

Sets PACC days count.

Sets PACC days count.

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

void function OTS_Account::setPassword(\$password) [line 184]

Function Parameters:

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

Sets account's password.

Sets account's password.

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

void function OTS_Account::unblock() [line 234]

Unblocks account.

Unblocks account.

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

Class OTS_Accounts_List

[line 19]

List of accounts.

List of accounts.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_Accounts_List::__construct(\$db) [line 54]

Function Parameters:

- [*IOTS_DB*](#) **\$db** Database connection object.

Sets database connection handler.

Sets database connection handler.

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

int function OTS_Accounts_List::count() [line 166]

Returns number of accounts on list in current criterium.

Returns number of accounts on list in current criterium.

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

OTS_Account function OTS_Accounts_List::current() [*line 116*]

Returns current row.

Returns current row.

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

bool function OTS_Accounts_List::deleteAccount(\$account) [*line 99*]

Function Parameters:

- [*OTS_Account*](#) **\$account** Account to be deleted.

Deletes account.

Deletes account.

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

mixed function OTS_Accounts_List::key() [*line 138*]

Current cursor position.

Current cursor position.

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

void function OTS_Accounts_List::next() [line 128]

Moves to next row.

Moves to next row.

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

void function OTS_Accounts_List::rewind() [line 156]

Select accounts from database.

Select accounts from database.

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

void function OTS_Accounts_List::setLimit([\$limit = false]) [line 64]

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

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

void function OTS_Accounts_List::setOffset([\$offset = false]) [*line 81*]

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

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

bool function OTS_Accounts_List::valid() [*line 148*]

Checks if there are any rows left.

Checks if there are any rows left.

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

Class OTS_DB_MySQL

[*line 19*]

MySQL connection interface.

MySQL connection interface.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_DB_MySQL::__construct(\$params) [*line 46*]

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\(\)](#)
- **Access** public

string function OTS_DB_MySQL::fieldName(\$name) [*line 101*]

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

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

string function OTS_DB_MySQL::limit([\$limit = false], [\$offset = false]) [*line 152*]

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
- **Access** public

PDOStatement|bool function OTS_DB_MySQL::SQLquery(\$query) [*line 140*]

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

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

string function OTS_DB_MySQL::SQLquote(\$string) [*line 126*]

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

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

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

string function OTS_DB_MySQL::tableName(\$name) [*line 112*]

Function Parameters:

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

Query-quoted table name.

Query-quoted table name.

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

Class OTS_DB_SQLite

[line 19]

SQLite connection interface.

SQLite connection interface.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_DB_SQLite::__construct(\$params) [line 42]

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\(\)](#)
- **Access** public

string function OTS_DB_SQLite::fieldName(\$name) [line 64]

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

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

string function OTS_DB_SQLite::limit([\$limit = false], [\$offset = false]) [*line 128*]

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
- **Access** public

bool function OTS_DB_SQLite::regexp(\$name, \$content) [*line 88*]

Function Parameters:

- *string* **\$name** Regular expression to test.
- *string* **\$content** String to test.

REGEXP operator for SQLite

REGEXP operator for SQLite

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

PDOStatement|bool function OTS_DB_SQLite::SQLquery(\$query) [*line 116*]

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

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

string function OTS_DB_SQLite::SQLquote(\$string) [*line 102*]

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

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

- **Version** 0.0.1

- **Access** public

string function OTS_DB_SQLite::tableName(\$name) [*line 75*]

Function Parameters:

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

Query-quoted table name.

Query-quoted table name.

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

Class OTS_Group

[*line 19*]

OTServ user group abstraction.

OTServ user group abstraction.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_Group::__construct(\$db) [*line 40*]

Function Parameters:

- [IOTS_DB](#) \$db Database connection object.

Sets database connection handler.

Sets database connection handler.

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

int|bool function OTS_Group::getAccess() [*line 160*]

Access level.

Access level.

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

int|bool function OTS_Group::getFlags() [*line 134*]

Rights flags.

Rights flags.

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

int|bool function OTS_Group::getId() [*line 92*]

Group ID.

Group ID.

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

int/bool function OTS_Group::getMaxDepotItems() [*line 186*]

Maximum count of items in depot.

Maximum count of items in depot.

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

int/bool function OTS_Group::getMaxVIPList() [*line 212*]

Maximum count of players in VIP list.

Maximum count of players in VIP list.

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

string/bool function OTS_Group::getName() [*line 108*]

Group name.

Group name.

- **Version** 0.0.1

- **Access** public

array/bool function OTS_Group::getPlayers() [*line 238*]

List of characters in given group.

List of characters in given group.

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

bool function OTS_Group::isLoading() [*line 61*]

Checks if object is loaded.

Checks if object is loaded.

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

void function OTS_Group::load(\$id) [*line 50*]

Function Parameters:

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

Loads group with given id.

Loads group with given id.

- **Version** 0.0.1

- **Access** public

void function OTS_Group::save() [line 69]

Saves account in database.

Saves account in database.

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

void function OTS_Group::setAccess(\$access) [line 176]

Function Parameters:

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

Sets access level.

Sets access level.

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

void function OTS_Group::setFlags(\$flags) [line 150]

Function Parameters:

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

Sets rights flags.

Sets rights flags.

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

void function OTS_Group::setMaxDepotItems(\$maxdepotitems) [line 202]

Function Parameters:

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

Sets maximum count of items in depot.

Sets maximum count of items in depot.

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

void function OTS_Group::setMaxVIPList(\$maxviplist, \$maxdepotitems) [line 228]

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

- **Access** public

void function OTS_Group::setName(\$name) [*line 124*]

Function Parameters:

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

Sets group's name.

Sets group's name.

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

Class OTS_Groups_List

[*line 19*]

List of groups.

List of groups.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_Groups_List::__construct(\$db) [*line 54*]

Function Parameters:

- [IOTS_DB](#) \$db Database connection object.

Sets database connection handler.

Sets database connection handler.

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

int function OTS_Groups_List::count() [*line 166*]

Returns number of groups on list in current criterium.

Returns number of groups on list in current criterium.

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

OTS_Group function OTS_Groups_List::current() [*line 116*]

Returns current row.

Returns current row.

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

bool function OTS_Groups_List::deleteGroup(\$group) [*line 99*]

Function Parameters:

- [OTS_Group](#) \$group Group to be deleted.

Deletes group.

Deletes group.

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

mixed function OTS_Groups_List::key() [*line 138*]

Current cursor position.

Current cursor position.

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

void function OTS_Groups_List::next() [*line 128*]

Moves to next row.

Moves to next row.

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

void function OTS_Groups_List::rewind() [*line 156*]

Select groups from database.

Select groups from database.

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

void function OTS_Groups_List::setLimit([\$limit = false]) [line 64]

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

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

void function OTS_Groups_List::setOffset([\$offset = false]) [line 81]

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

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

bool function OTS_Groups_List::valid() [*line 148*]

Checks if there are any rows left.

Checks if there are any rows left.

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

Class OTS_Player

[*line 19*]

OTServ character abstraction.

OTServ character abstraction.

- **Package** POT
- **Version** 0.0.1

Constructor *void* function OTS_Player::__construct(\$db) [*line 40*]

Function Parameters:

- [*IOTS_DB*](#) **\$db** Database connection object.

Sets database connection handler.

Sets database connection handler.

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

void function OTS_Player::find(\$name) [*line 61*]

Function Parameters:

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

Loads player by it's name.

Loads player by it's name.

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

OTS_Account function OTS_Player::getAccount() [*line 151*]

Returns account of this player.

Returns account of this player.

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

int/bool function OTS_Player::getCap() [*line 753*]

Capacity.

Capacity.

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

mixed|bool function OTS_Player::getConditions() [*line 863*]

Conditions.

Conditions.

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

int|bool function OTS_Player::getDirection() [*line 493*]

Looking direction.

Looking direction.

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

int|bool function OTS_Player::getExperience() [*line 259*]

Experience points.

Experience points.

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

OTS_Group function OTS_Player::getGroup() [*line 179*]

Returns group of this player.

Returns group of this player.

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

string/bool function OTS_Player::getGuildNick() [*line 947*]

Guild nick.

Guild nick.

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

int/bool function OTS_Player::getHealth() [*line 337*]

Current HP.

Current HP.

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

int/bool function OTS_Player::getHealthMax() [*line 363*]

Maximum HP.

Maximum HP.

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

int/bool function OTS_Player::getId() [*line 109*]

Player ID.

Player ID.

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

int/bool function OTS_Player::getLastIP() [*line 805*]

Last login IP.

Last login IP.

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

int/bool function OTS_Player::getLastLogin() [*line 779*]

Last login timestamp.

Last login timestamp.

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

int/bool function OTS_Player::getLevel() [*line 285*]

Experience level.

Experience level.

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

int/bool function OTS_Player::getLookAddons() [*line 649*]

Addons.

Addons.

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

int/bool function OTS_Player::getLookBody() [*line 519*]

Body color.

Body color.

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

int/bool function OTS_Player::getLookFeet() [*line 545*]

Boots color.

Boots color.

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

int/bool function OTS_Player::getLookHead() [*line 571*]

Hair color.

Hair color.

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

int/bool function OTS_Player::getLookLegs() [*line 597*]

Legs color.

Legs color.

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

int/bool function OTS_Player::getLookType() [*line 623*]

Outfit.

Outfit.

- **Version** 0.0.1

- **Access** public

int/bool function OTS_Player::getLossExperience() [*line 1023*]

Percentage of experience lost after dead.

Percentage of experience lost after dead.

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

int/bool function OTS_Player::getLossMana() [*line 1048*]

Percentage of used mana lost after dead.

Percentage of used mana lost after dead.

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

int/bool function OTS_Player::getLossSkills() [*line 1073*]

Percentage of skills lost after dead.

Percentage of skills lost after dead.

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

int/bool function OTS_Player::getMagLevel() [*line 311*]

Magic level.

Magic level.

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

int/bool function OTS_Player::getMana() [*line 389*]

Current mana.

Current mana.

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

int/bool function OTS_Player::getManaMax() [*line 415*]

Maximum mana.

Maximum mana.

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

int/bool function OTS_Player::getManaSpent() [*line 441*]

Mana spent.

Mana spent.

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

string|bool function OTS_Player::getName() [*line 125*]

Player name.

Player name.

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

int|bool function OTS_Player::getPosX() [*line 675*]

X map coordinate.

X map coordinate.

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

int|bool function OTS_Player::getPosY() [*line 701*]

Y map coordinate.

Y map coordinate.

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

int|bool function OTS_Player::getPosZ() [*line 727*]

Z map coordinate.

Z map coordinate.

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

int/bool function OTS_Player::getRankId() [*line 973*]

Guild rank ID.

Guild rank ID.

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

int/bool function OTS_Player::getRedSkullTime() [*line 889*]

Red skulled time remained.

Red skulled time remained.

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

int/bool function OTS_Player::getSex() [*line 207*]

Player gender.

Player gender.

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

int|bool function OTS_Player::getSoul() [*line 467*]

Soul points.

Soul points.

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

int|bool function OTS_Player::getTownId() [*line 998*]

Residence town's ID.

Residence town's ID.

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

int|bool function OTS_Player::getVocation() [*line 233*]

Player proffesion.

Player proffesion.

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

bool|null function OTS_Player::hasRedSkull() [*line 915*]

Checks if player has red skull.

Checks if player has red skull.

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

bool function OTS_Player::isLoading() [*line 78*]

Checks if object is loaded.

Checks if object is loaded.

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

bool|null function OTS_Player::isSaveSet() [*line 831*]

Checks if save flag is set.

Checks if save flag is set.

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

void function OTS_Player::load(\$id) [*line 50*]

Function Parameters:

- *int* **\$id** Player's ID.

Loads player with given id.

Loads player with given id.

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

void function OTS_Player::save() [line 86]

Saves account in database.

Saves account in database.

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

void function OTS_Player::setAccount(\$account) [line 169]

Function Parameters:

- [OTS_Account](#) **\$account** Owning account.

Assigns character to account.

Assigns character to account.

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

void function OTS_Player::setCap(\$cap) [line 769]

Function Parameters:

- *int* **\$cap** Capacity.

Sets capacity.

Sets capacity.

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

void function OTS_Player::setConditions(\$conditions) [line 879]

Function Parameters:

- *mixed* **\$conditions** Condition binary field.

Sets conditions.

Sets conditions.

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

void function OTS_Player::setDirection(\$direction) [line 509]

Function Parameters:

- *int* **\$direction** Looking direction.

Sets looking direction.

Sets looking direction.

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

void function OTS_Player::setExperience(\$experience) [line 275]

Function Parameters:

- *int* **\$experience** Experience points.

Sets experience points.

Sets experience points.

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

void function OTS_Player::setGroup(\$group) [line 197]

Function Parameters:

- [*OTS_Group*](#) **\$group** Group to be a member.

Assigns character to group.

Assigns character to group.

- **Version** 0.0.1

- **Access** public

void function OTS_Player::setGuildNick(\$guildnick) [line 963]

Function Parameters:

- *string* **\$guildnick** Name.

Sets guild nick.

Sets guild nick.

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

void function OTS_Player::setHealth(\$health) [line 353]

Function Parameters:

- *int* **\$health** Current HP.

Sets current HP.

Sets current HP.

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

void function OTS_Player::setHealthMax(\$healthmax) [line 379]

Function Parameters:

- *int* **\$healthmax** Maximum HP.

Sets maximum HP.

Sets maximum HP.

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

void function OTS_Player::setLastIP(\$lastip) [*line 821*]

Function Parameters:

- *int* **\$lastip** Last login IP.

Sets last login IP.

Sets last login IP.

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

void function OTS_Player::setLastLogin(\$lastlogin) [*line 795*]

Function Parameters:

- *int* **\$lastlogin** Last login timestamp.

Sets last login timestamp.

Sets last login timestamp.

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

void function OTS_Player::setLevel(\$level) [line 301]

Function Parameters:

- *int* **\$level** Experience level.

Sets experience level.

Sets experience level.

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

void function OTS_Player::setLookAddons(\$lookaddons) [line 665]

Function Parameters:

- *int* **\$lookaddons** Addons.

Sets addons.

Sets addons.

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

void function OTS_Player::setLookBody(\$lookbody) [line 535]

Function Parameters:

- *int* **\$lookbody** Body color.

Sets body color.

Sets body color.

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

void function OTS_Player::setLookFeet(\$lookfeet) [line 561]

Function Parameters:

- *int* **\$lookfeet** Boots color.

Sets boots color.

Sets boots color.

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

void function OTS_Player::setLookHead(\$lookhead) [line 587]

Function Parameters:

- *int* **\$lookhead** Hair color.

Sets hair color.

Sets hair color.

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

void function OTS_Player::setLookLegs(\$looklegs) [line 613]

Function Parameters:

- *int* **\$looklegs** Legs color.

Sets legs color.

Sets legs color.

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

void function OTS_Player::setLookType(\$looktype) [line 639]

Function Parameters:

- *int* **\$looktype** Outfit.

Sets outfit.

Sets outfit.

- **Version** 0.0.1

- **Access** public

void function OTS_Player::setLossExperience(\$loss_experience) [line 1039]

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
- **Access** public

void function OTS_Player::setLossMana(\$loss_mana) [line 1064]

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
- **Access** public

void function OTS_Player::setLossSkills(\$loss_skills) [line 1089]

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
- **Access** public

void function OTS_Player::setMagLevel(\$maglevel) [*line 327*]

Function Parameters:

- *int* **\$maglevel** Magic level.

Sets magic level.

Sets magic level.

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

void function OTS_Player::setMana(\$mana) [*line 405*]

Function Parameters:

- *int* **\$mana** Current mana.

Sets current mana.

Sets current mana.

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

void function OTS_Player::setManaMax(\$manamax) [*line 431*]

Function Parameters:

- *int* **\$manamax** Maximum mana.

Sets maximum mana.

Sets maximum mana.

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

void function OTS_Player::setManaSpent(\$manaspent) [*line 457*]

Function Parameters:

- *int* **\$manaspent** Mana spent.

Sets mana spent.

Sets mana spent.

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

void function OTS_Player::setName(\$name) [line 141]

Function Parameters:

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

Sets players's name.

Sets players's name.

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

void function OTS_Player::setPosX(\$posx) [line 691]

Function Parameters:

- *int* **\$posx** X map coordinate.

Sets X map coordinate.

Sets X map coordinate.

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

void function OTS_Player::setPosY(\$posy) [line 717]

Function Parameters:

- *int* **\$posy** Y map coordinate.

Sets Y map coordinate.

Sets Y map coordinate.

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

void function OTS_Player::setPosZ(\$posz) [line 743]

Function Parameters:

- *int* **\$posz** Z map coordinate.

Sets Z map coordinate.

Sets Z map coordinate.

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

void function OTS_Player::setRankId(\$rank_id) [line 989]

Function Parameters:

- *int* **\$rank_id** Guild rank ID.

Sets guild rank ID.

Sets guild rank ID.

- **Version** 0.0.1

- **Access** public

void function OTS_Player::setRedSkull() [line 937]

Sets red skull flag.

Sets red skull flag.

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

void function OTS_Player::setRedSkullTime(\$redskulltime) [line 905]

Function Parameters:

- *int* **\$redskulltime** Red skulled time remained.

Sets red skulled time remained.

Sets red skulled time remained.

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

void function OTS_Player::setSave() [line 853]

Sets save flag.

Sets save flag.

- **Version** 0.0.1

- **Access** public

void function OTS_Player::setSex(\$sex) [line 223]

Function Parameters:

- *int* **\$sex** Player gender.

Sets player gender.

Sets player gender.

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

void function OTS_Player::setSoul(\$soul) [line 483]

Function Parameters:

- *int* **\$soul** Soul points.

Sets soul points.

Sets soul points.

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

void function OTS_Player::setTownId(\$town_id) [line 1014]

Function Parameters:

- *int* **\$town_id** Residence town's ID.

Sets residence town's ID.

Sets residence town's ID.

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

void function OTS_Player::setVocation(\$vocation) [*line 249*]

Function Parameters:

- *int* **\$vocation** Player proffesion.

Sets player proffesion.

Sets player proffesion.

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

void function OTS_Player::unsetRedSkull() [*line 929*]

Unsets red skull flag.

Unsets red skull flag.

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

void function OTS_Player::unsetSave() [line 845]

Unsets save flag.

Unsets save flag.

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

Class OTS_Players_List

[line 19]

List of players.

List of players.

- **Package** POT
- **Version** 0.0.1

Constructor *void function OTS_Players_List::__construct(\$db) [line 54]*

Function Parameters:

- [*IOTS_DB*](#) **\$db** Database connection object.

Sets database connection handler.

Sets database connection handler.

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

int function OTS_Players_List::count() [*line 166*]

Returns number of characters on list in current criterium.

Returns number of characters on list in current criterium.

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

OTS_Player function OTS_Players_List::current() [*line 116*]

Returns current row.

Returns current row.

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

bool function OTS_Players_List::deletePlayer(\$player) [*line 99*]

Function Parameters:

- [*OTS_Player*](#) **\$player** Player to be deleted.

Deletes player.

Deletes player.

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

mixed function OTS_Players_List::key() [*line 138*]

Current cursor position.

Current cursor position.

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

void function OTS_Players_List::next() [*line 128*]

Moves to next row.

Moves to next row.

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

void function OTS_Players_List::rewind() [*line 156*]

Select players from database.

Select players from database.

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

void function OTS_Players_List::setLimit([\$limit = false]) [line 64]

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

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

void function OTS_Players_List::setOffset([\$offset = false]) [line 81]

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

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

bool function OTS_Players_List::valid() [line 148]

Checks if there are any rows left.

Checks if there are any rows left.

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

Class POT

[line 21]

Main POT class.
Main POT class.

- **Package** POT
- **Version** 0.0.1

POT::DB_MYSQL

= 1 *[line 26]*

MySQL driver.
MySQL driver.

- **Version** 0.0.1

POT::DB_SQLITE

= 2 *[line 30]*

SQLite driver.
SQLite driver.

- **Version 0.0.1**

POT::DIRECTION_EAST

= 1 *[line 69]*

East.
East.

- **Version 0.0.1**

POT::DIRECTION_NORTH

= 0 *[line 65]*

North.
North.

- **Version 0.0.1**

POT::DIRECTION_SOUTH

= 2 *[line 73]*

South.
South.

- **Version 0.0.1**

POT::DIRECTION_WEST

= 3 *[line 77]*

West.

West.

- **Version 0.0.1**

POT::SEX_FEMALE

= 0 *[line 35]*

Female gender.

Female gender.

- **Version 0.0.1**

POT::SEX_MALE

= 1 *[line 39]*

Male gender.

Male gender.

- **Version 0.0.1**

POT::VOCATION_DRUID

= 2 *[line 52]*

Druid.

Druid.

- **Version 0.0.1**

POT::VOCATION_KNIGHT

= 4 *[line 60]*

Knight.

Knight.

- **Version 0.0.1**

POT::VOCATION_NONE

= 0 *[line 44]*

None vocation.

None vocation.

- **Version 0.0.1**

POT::VOCATION_PALADIN

= 3 *[line 56]*

Paladin.
Paladin.

- **Version** 0.0.1

POT::VOCATION_SORCERER

= 1 *[line 48]*

Sorcerer.
Sorcerer.

- **Version** 0.0.1

Constructor *void* function POT::__construct() *[line 134]*

Class initialization tools.

Class initialization tools.

Never create instance of this class by yourself! Use POT::getInstance()!

- **Version** 0.0.1
- **See** POT::getInstance();
- **Access** public

void function POT::connect(\$driver, \$params) *[line 191]*

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 and SQLite drivers are supported. XML is not planned.

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, applies 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** 0.0.1
- **Throws** Exception When driver is not supported.

- **Access** public
- **Example**

IOTS_DAO function POT::createObject(\$class) [*line 236*]

Function Parameters:

- *string* **\$class** Class name.

Creates OTServ DAO class instance.

Creates OTServ DAO class instance.

Currently it means Account, or Player object.

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

POT function POT::getInstance() [*line 84*]

Singleton.

Singleton.

- **Version** 0.0.1
- **Static**
- **Access** public

void function POT::loadClass(\$class) [*line 149*]

autoload.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     function __autoload($class)
15     {
16         // checks if it's POT class
17         if( preg_match('/^I?OTS_/', $class) != 0)
18         {
19             POT::getInstance()-> loadClass($class);
20         }
21     /*
22         // possibly call your own __autoload() handler
23         else
24         {
25             here comes your stuff...
26         }
27     */
28     }
29
30     ?>

```

Function Parameters:

- **string \$class** Class name.

Loads POT class file.

Loads POT class file.

Runtime class loading on demand - usefull for __autoload() function.

- **Version** 0.0.1
- **Throws** Exception When give class is not POT toolkit class.
- **Access** public
- **Example**

void function POT::setPOTPath(\$path) [line 115]

fakeroot.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 // this is the way you should work with POT if you moved main OTS.php file outside POT's directory
12 include('path/to/OTS.php');
13
14 // dont use 'new POT()'!!!
15 $ots = POT::getInstance();
16 $ots-> setPOTPath('../classes/');
17
18 /*
19  here comes your stuff...
20 */
21
22 ?>

```

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
- **Access** public
- **Example**

Appendices

Appendix A - Class Trees

Package POT

IOTS_DAO

- [IOTS_DAO](#)

IOTS_DB

- [IOTS_DB](#)

OTS_Account

- [OTS_Account](#)

OTS_Accounts_List

- [OTS_Accounts_List](#)

OTS_DB_MySQL

- PDO
 - [OTS_DB_MySQL](#)

OTS_DB_SQLite

- PDO
 - [OTS_DB_SQLite](#)

OTS_Group

- [OTS_Group](#)

OTS_Groups_List

- [OTS_Groups_List](#)

OTS_Player

- [OTS_Player](#)

OTS_Players_List

- [OTS_Players_List](#)

POT

- [POT](#)

Appendix B - README/CHANGELOG/INSTALL

CHANGELOG

[0.0.1]

* Initial release. <wrzasq>

INSTALL

POT is a toolkit which means you don't literally 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

This is the very first release of this toolkit. Read README file for more info.

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 language to work with OTServ database.

For installation help check INSTALL file.

For usage tutorial/API documentation check documentation/index.html or documentation.pdf files.

===== Contact =====

In case of any contact needed, please use following e-mail address: wrzasq@gmail.com.

===== Files =====

classes/ - POT class files.
documentation/ - phpDocumentor-generator documentation.
examples/ - example files for learning.
tutorials/ - phpDocumentor directory.
BUGS - known bugs.
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.
TODO - list of things to be done.
Makefile - make input, for documentation generation.
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, test).
clean: deletes documentation.
check: checks syntax of all PHP files.
documentation: generates HTML documentation.
pdf: generates PDF documentation.
test: runs test suite.

For more readable output of PHPUnit test run:
php test.php

===== Credits =====

* Wrzasq <wrzasq@gmail.com> - project initiator, main developer.

Index

C

constructor OTS_Player::__construct()	53
<i>Sets database connection handler.</i>	
constructor OTS_Groups_List::__construct()	49
<i>Sets database connection handler.</i>	
constructor OTS_Players_List::__construct()	82
<i>Sets database connection handler.</i>	
constructor POT::__construct()	90
<i>Class initialization tools.</i>	
CHANGELOG	99
constructor OTS_Group::__construct()	43
<i>Sets database connection handler.</i>	
constructor OTS_DB_SQLite::__construct()	40
<i>Creates database connection.</i>	
constructor IOTS_DB::__construct()	23
<i>Connection parameters.</i>	
constructor OTS_Account::__construct()	26
<i>Sets database connection handler.</i>	
constructor OTS_Accounts_List::__construct()	33
<i>Sets database connection handler.</i>	
constructor OTS_DB_MySQL::__construct()	37
<i>Creates database connection.</i>	
constructor IOTS_DAO::__construct()	22
<i>DAO objects must be initialized with a database.</i>	

I

IOTS_DB::SQLquery()	25
<i>Evaluates query.</i>	
IOTS_DB::SQLquote()	25
<i>Query-quoted string value.</i>	
IOTS_DB::tableName()	25
<i>Query-quoted table name.</i>	
INSTALL	99
IOTS_DB::limit()	24
<i>LIMIT/OFFSET clause for queries.</i>	
IOTS_DB::lastInsertId()	24
<i>ID of last created record.</i>	
IOTS_DB.php	11
IOTS_DAO	22
<i>OTServ database object.</i>	
IOTS_DB	23
<i>OTServ database handler interface.</i>	
IOTS_DB::fieldName()	23

Query-quoted field name.	
IOTS.DAO.php	10

N

NEWS	99
----------------------	----

O

OTS_Player::getTownId()	64
<i>Residence town's ID.</i>	
OTS_Player::getVocation()	64
<i>Player proffesion.</i>	
OTS_Player::hasRedSkull()	65
<i>Checks if player has red skull.</i>	
OTS_Player::isLoading()	65
<i>Checks if object is loaded.</i>	
OTS_Player::getSoul()	64
<i>Soul points.</i>	
OTS_Player::getSex()	63
<i>Player gender.</i>	
OTS_Player::getPosZ()	62
<i>Z map coordinate.</i>	
OTS_Player::getRankId()	63
<i>Guild rank ID.</i>	
OTS_Player::getRedSkullTime()	63
<i>Red skulled time remained.</i>	
OTS_Player::isSaveSet()	65
<i>Checks if save flag is set.</i>	
OTS_Player::load()	65
<i>Loads player with given id.</i>	
OTS_Player::setExperience()	68
<i>Sets experience points.</i>	
OTS_Player::setGroup()	68
<i>Assigns character to group.</i>	
OTS_Player::setGuildNick()	69
<i>Sets guild nick.</i>	
OTS_Player::setDirection()	67
<i>Sets looking direction.</i>	
OTS_Player::setConditions()	67
<i>Sets conditions.</i>	
OTS_Player::save()	66
<i>Saves account in database.</i>	
OTS_Player::setAccount()	66
<i>Assigns character to account.</i>	
OTS_Player::setCap()	67
<i>Sets capacity.</i>	
OTS_Player::getPosY()	62
<i>Y map coordinate.</i>	
OTS_Player::getPosX()	62
<i>X map coordinate.</i>	

<u>OTS_Player::getLookAddons()</u>	58
<i>Addons.</i>	
<u>OTS_Player::getLookBody()</u>	58
<i>Body color.</i>	
<u>OTS_Player::getLookFeet()</u>	58
<i>Boots color.</i>	
<u>OTS_Player::getLookHead()</u>	59
<i>Hair color.</i>	
<u>OTS_Player::getLevel()</u>	58
<i>Experience level.</i>	
<u>OTS_Player::getLastLogin()</u>	57
<i>Last login timestamp.</i>	
<u>OTS_Player::getHealthMax()</u>	56
<i>Maximum HP.</i>	
<u>OTS_Player::getId()</u>	57
<i>Player ID.</i>	
<u>OTS_Player::getLastIP()</u>	57
<i>Last login IP.</i>	
<u>OTS_Player::getLookLegs()</u>	59
<i>Legs color.</i>	
<u>OTS_Player::getLookType()</u>	59
<i>Outfit.</i>	
<u>OTS_Player::getManaMax()</u>	61
<i>Maximum mana.</i>	
<u>OTS_Player::getManaSpent()</u>	61
<i>Mana spent.</i>	
<u>OTS_Player::getName()</u>	62
<i>Player name.</i>	
<u>OTS_Player::getMana()</u>	61
<i>Current mana.</i>	
<u>OTS_Player::getMagLevel()</u>	60
<i>Magic level.</i>	
<u>OTS_Player::getLossExperience()</u>	60
<i>Percentage of experience lost after dead.</i>	
<u>OTS_Player::getLossMana()</u>	60
<i>Percentage of used mana lost after dead.</i>	
<u>OTS_Player::getLossSkills()</u>	60
<i>Percentage of skills lost after dead.</i>	
<u>OTS_Player::setHealth()</u>	69
<i>Sets current HP.</i>	
<u>OTS_Player::setHealthMax()</u>	69
<i>Sets maximum HP.</i>	
<u>OTS_Player::setTownId()</u>	80
<i>Sets residence town's ID.</i>	
<u>OTS_Player::setVocation()</u>	81
<i>Sets player proffesion.</i>	
<u>OTS_Player::unsetRedSkull()</u>	81
<i>Unsets red skull flag.</i>	
<u>OTS_Player::unsetSave()</u>	82
<i>Unsets save flag.</i>	
<u>OTS_Player::setSoul()</u>	80
<i>Sets soul points.</i>	
<u>OTS_Player::setSex()</u>	80

<i>Sets player gender.</i>	
OTS_Player::setRedSkull()	79
<i>Sets red skull flag.</i>	
OTS_Player::setRedSkullTime()	79
<i>Sets red skulled time remained.</i>	
OTS_Player::setSave()	79
<i>Sets save flag.</i>	
OTS_Players_List	82
<i>List of players.</i>	
OTS_Players_List::count()	83
<i>Returns number of characters on list in current criterium.</i>	
OTS_Players_List::setLimit()	85
<i>Sets LIMIT.</i>	
OTS_Players_List::setOffset()	85
<i>Sets OFFSET.</i>	
OTS_Players_List::valid()	85
<i>Checks if there are any rows left.</i>	
OTS_Players_List::rewind()	84
<i>Select players from database.</i>	
OTS_Players_List::next()	84
<i>Moves to next row.</i>	
OTS_Players_List::current()	83
<i>Returns current row.</i>	
OTS_Players_List::deletePlayer()	83
<i>Deletes player.</i>	
OTS_Players_List::key()	84
<i>Current cursor position.</i>	
OTS_Player::setRankId()	78
<i>Sets guild rank ID.</i>	
OTS_Player::setPosZ()	78
<i>Sets Z map coordinate.</i>	
OTS_Player::setLookFeet()	72
<i>Sets boots color.</i>	
OTS_Player::setLookHead()	72
<i>Sets hair color.</i>	
OTS_Player::setLookLegs()	73
<i>Sets legs color.</i>	
OTS_Player::setLookType()	73
<i>Sets outfit.</i>	
OTS_Player::setLookBody()	72
<i>Sets body color.</i>	
OTS_Player::setLookAddons()	71
<i>Sets addons.</i>	
OTS_Player::setLastIP()	70
<i>Sets last login IP.</i>	
OTS_Player::setLastLogin()	70
<i>Sets last login timestamp.</i>	
OTS_Player::setLevel()	71
<i>Sets experience level.</i>	
OTS_Player::setLossExperience()	74
<i>Sets percentage of experience lost after dead.</i>	
OTS_Player::setLossMana()	74
<i>Sets percentage of used mana lost after dead.</i>	

OTS_Player::setName()	77
<i>Sets players's name.</i>	
OTS_Player::setPosX()	77
<i>Sets X map coordinate.</i>	
OTS_Player::setPosY()	77
<i>Sets Y map coordinate.</i>	
OTS_Player::setManaSpent()	76
<i>Sets mana spent.</i>	
OTS_Player::setManaMax()	76
<i>Sets maximum mana.</i>	
OTS_Player::setLossSkills()	74
<i>Sets percentage of skills lost after dead.</i>	
OTS_Player::setMagLevel()	75
<i>Sets magic level.</i>	
OTS_Player::setMana()	75
<i>Sets current mana.</i>	
OTS_Player::getHealth()	56
<i>Current HP.</i>	
OTS_Player::getGuildNick()	56
<i>Guild nick.</i>	
OTS_Accounts_List::count()	33
<i>Returns number of accounts on list in current criterium.</i>	
OTS_Accounts_List::current()	34
<i>Returns current row.</i>	
OTS_Accounts_List::deleteAccount()	34
<i>Deletes account.</i>	
OTS_Accounts_List::key()	34
<i>Current cursor position.</i>	
OTS_Accounts_List	33
<i>List of accounts.</i>	
OTS_Account::unblock()	32
<i>Unblocks account.</i>	
OTS_Account::setEMail()	31
<i>Sets account's email.</i>	
OTS_Account::setPACCDays()	31
<i>Sets PACC days count.</i>	
OTS_Account::setPassword()	32
<i>Sets account's password.</i>	
OTS_Accounts_List::next()	35
<i>Moves to next row.</i>	
OTS_Accounts_List::rewind()	35
<i>Select accounts from database.</i>	
OTS_DB_MySQL::limit()	38
<i>LIMIT/OFFSET clause for queries.</i>	
OTS_DB_MySQL::SQLquery()	38
<i>IOTS_DB method.</i>	
OTS_DB_MySQL::SQLquote()	39
<i>IOTS_DB method.</i>	
OTS_DB_MySQL::fieldName()	37
<i>Query-quoted field name.</i>	
OTS_DB_MySQL	36
<i>MySQL connection interface.</i>	
OTS_Accounts_List::setLimit()	35

Sets <i>LIMIT</i> .	
OTS Accounts List::setOffset()	36
Sets <i>OFFSET</i> .	
OTS Accounts List::valid()	36
Checks if there are any rows left.	
OTS Account::save()	31
Updates account in database.	
OTS Account::load()	30
Loads account with given number.	
OTS Groups List.php	18
OTS Player.php	19
OTS Players List.php	20
OTS SQLite Results.php	21
OTS Group.php	17
OTS DB SQLite.php	16
OTS Account.php	13
OTS Accounts List.php	14
OTS DB MySQL.php	15
OTS Account	26
OTServ account abstraction.	
OTS Account::block()	27
Blocks account.	
OTS Account::getPlayers()	29
List of characters on account.	
OTS Account::isBlocked()	30
Checks if account is blocked.	
OTS Account::isLoaded()	30
Checks if object is loaded.	
OTS Account::getPassword()	29
Account's password.	
OTS Account::getPACCDays()	29
PACC days.	
OTS Account::create()	27
Creates new account.	
OTS Account::getEmail()	28
E-mail address.	
OTS Account::getId()	28
Account number.	
OTS DB MySQL::tableName()	39
Query-quoted table name.	
OTS DB SQLite	40
SQLite connection interface.	
OTS Groups List::key()	51
Current cursor position.	
OTS Groups List::next()	51
Moves to next row.	
OTS Groups List::rewind()	51
Select groups from database.	
OTS Groups List::setLimit()	52
Sets <i>LIMIT</i> .	
OTS Groups List::deleteGroup()	50
Deletes group.	
OTS Groups List::current()	50

<i>Returns current row.</i>	
OTS_Group::setName()	49
<i>Sets group's name.</i>	
OTS_Groups_List	49
<i>List of groups.</i>	
OTS_Groups_List::count()	50
<i>Returns number of groups on list in current criterium.</i>	
OTS_Groups_List::setOffset()	52
<i>Sets OFFSET.</i>	
OTS_Groups_List::valid()	53
<i>Checks if there are any rows left.</i>	
OTS_Player::getDirection()	55
<i>Looking direction.</i>	
OTS_Player::getExperience()	55
<i>Experience points.</i>	
OTS_Player::getGroup()	56
<i>Returns group of this player.</i>	
OTS_Player::getConditions()	55
<i>Conditions.</i>	
OTS_Player::getCap()	54
<i>Capacity.</i>	
OTS_Player	53
<i>OTServ character abstraction.</i>	
OTS_Player::find()	54
<i>Loads player by it's name.</i>	
OTS_Player::getAccount()	54
<i>Returns account of this player.</i>	
OTS_Group::setMaxVIPList()	48
<i>Sets maximum count of players in VIP list.</i>	
OTS_Group::setMaxDepotItems()	48
<i>Sets maximum count of items in depot.</i>	
OTS_DB_SQLite::tableName()	43
<i>Query-quoted table name.</i>	
OTS_Group	43
<i>OTServ user group abstraction.</i>	
OTS_Group::getAccess()	44
<i>Access level.</i>	
OTS_Group::getFlags()	44
<i>Rights flags.</i>	
OTS_DB_SQLite::SQLquote()	42
<i>IOTS_DB method.</i>	
OTS_DB_SQLite::SQLquery()	42
<i>IOTS_DB method.</i>	
OTS_DB_SQLite::fieldName()	40
<i>Query-quoted field name.</i>	
OTS_DB_SQLite::limit()	41
<i>LIMIT/OFFSET clause for queries.</i>	
OTS_DB_SQLite::regexp()	41
<i>REGEXP operator for SQLite</i>	
OTS_Group::getId()	44
<i>Group ID.</i>	
OTS_Group::getMaxDepotItems()	45
<i>Maximum count of items in depot.</i>	

OTS_Group::save()	47
<i>Saves account in database.</i>	
OTS_Group::setAccess()	47
<i>Sets access level.</i>	
OTS_Group::setFlags()	47
<i>Sets rights flags.</i>	
OTS_Group::load()	46
<i>Loads group with given id.</i>	
OTS_Group::isLoading()	46
<i>Checks if object is loaded.</i>	
OTS_Group::getMaxVIPList()	45
<i>Maximum count of players in VIP list.</i>	
OTS_Group::getName()	45
<i>Group name.</i>	
OTS_Group::getPlayers()	46
<i>List of characters in given group.</i>	
OTS.php	12
<i>This file contains main toolkit class.</i>	

P

POT::VOCATION_SORCERER	90
<i>Sorcerer.</i>	
POT::VOCATION_PALADIN	89
<i>Paladin.</i>	
POT::VOCATION_NONE	89
<i>None vocation.</i>	
POT::VOCATION_KNIGHT	89
<i>Knight.</i>	
POT::connect()	90
<i>Connects to database.</i>	
POT::createObject()	92
<i>Creates OTServ DAO class instance.</i>	
POT::setPOTPath()	93
<i>Set POT directory.</i>	
POT::loadClass()	92
<i>Loads POT class file.</i>	
POT::getInstance()	92
<i>Singleton.</i>	
POT::VOCATION_DRUID	89
<i>Druid.</i>	
POT::SEX_MALE	88
<i>Male gender.</i>	
POT::DB_SQLITE	86
<i>SQLite driver.</i>	
POT::DB_MYSQL	86
<i>MySQL driver.</i>	
POT	86
<i>Main POT class.</i>	
POT class preview	3
POT::DIRECTION_EAST	87
<i>East.</i>	

POT::DIRECTION NORTH	87
<i>North.</i>	
POT::SEX FEMALE	88
<i>Female gender.</i>	
POT::DIRECTION WEST	88
<i>West.</i>	
POT::DIRECTION SOUTH	87
<i>South.</i>	
POT	1

Q	
Quick start	5

R	
README	99