

PHP OTServ Toolkit



Contents

POT	1
PHP 5.0	3
POT class preview	5
Quick start	6
DAO objects	9
Account number hack	13
Server online status	14
Package POT Procedural Elements	17
E OTS NoDriver.php	17
E OTS NotLoaded.php	18
IOTS DAO.php	19
IOTS DB.php	20
IOTS GuildAction.php	21
OTS.php	22
OTS Account.php	23
OTS Accounts List.php	24
OTS Container.php	25
OTS DB MySQL.php	26
OTS DB ODBC.php	27
OTS DB PostgreSQL.php	28
OTS DB SQLite.php	29
OTS Group.php	30
OTS Groups List.php	31
OTS Guild.php	32
OTS GuildRank.php	33
OTS GuildRanks List.php	34
OTS Guilds List.php	35
OTS InfoRespond.php	36
OTS Item.php	37
OTS Player.php	38
OTS Players List.php	39
OTS SQLite Results.php	40
Package POT Classes	41
Class E OTS NoDriver	41
Class E OTS NotLoaded	41
Class IOTS DAO	42
Constructor construct	42
Class IOTS DB	43
Constructor construct	43
Method fieldName	43
Method lastInsertId	44
Method limit	44

Method SQLquery	45
Method SQLquote	45
Method tableName	45
Class IOTS GuildAction	46
Constructor construct	46
Method addRequest	47
Method deleteRequest	47
Method listRequests	48
Method submitRequest	48
Class OTS Account	49
Constructor construct	49
Method block	49
Method create	50
example: account.php	50
Method createEx	51
example: account.php	51
Method find	52
Method getCustomField	52
Method getEmail	53
Method getGroup	53
Method getId	54
Method getPACCDays	54
Method getPassword	55
Method getPlayers	55
Method isBlocked	55
Method isLoaded	56
Method load	56
Method save	56
Method setCustomField	57
Method setEmail	57
Method setGroup	58
Method setPACCDays	58
Method setPassword	59
Method unblock	59
Class OTS Accounts List	60
Constructor construct	60
Method count	60
Method current	61
Method deleteAccount	61
Method key	61
Method next	62
Method rewind	62
Method setLimit	62
Method setOffset	63
Method valid	63
Class OTS Container	64
Method addItem	64
Method count	64
Method current	65

Method key	65
Method next	65
Method removeItem	66
Method rewind	66
Method valid	66
Class OTS_DB_MySQL	67
Constructor construct	67
Method fieldName	68
Method limit	68
Method SQLquery	69
Method SQLquote	69
Method tableName	70
Class OTS_DB_ODBC	70
Constructor construct	70
Method fieldName	71
Method limit	72
Method SQLquery	72
Method SQLquote	73
Method tableName	73
Class OTS_DB_PostgreSQL	74
Constructor construct	74
Method fieldName	75
Method limit	75
Method SQLquery	76
Method SQLquote	76
Method tableName	76
Class OTS_DB_SQLite	77
Constructor construct	77
Method fieldName	78
Method limit	78
Method SQLquery	79
Method SQLquote	79
Method tableName	80
Class OTS_Group	80
Constructor construct	81
Method getAccess	81
Method getCustomField	81
Method getFlags	82
Method getID	82
Method getMaxDepotItems	83
Method getMaxVIPList	83
Method getName	83
Method getPlayers	84
Method isLoaded	84
Method load	84
Method save	85
Method setAccess	85
Method setCustomField	86
Method setFlags	86

Method setMaxDepotItems	87
Method setMaxVIPList	87
Method setName	88
Class OTS_Groups_List	88
Constructor construct	88
Method count	89
Method current	89
Method deleteGroup	89
Method key	90
Method next	90
Method rewind	90
Method setLimit	91
Method setOffset	91
Method valid	92
Class OTS_Guild	92
Constructor construct	92
Method acceptInvite	93
Method acceptRequest	93
Method deleteInvite	94
Method deleteRequest	94
Method find	95
Method getCreationData	95
Method getCustomField	96
Method getGuildRanks	96
Method getId	97
Method getName	97
Method getOwner	97
Method invite	98
Method isLoading	98
Method listInvites	99
Method listRequests	99
Method load	99
Method request	100
Method save	100
Method setCreationData	101
Method setCustomField	101
Method setInvitesDriver	102
Method setName	102
Method setOwner	103
Method setRequestsDriver	103
Class OTS_GuildRank	104
Constructor construct	104
Method find	104
Method getCustomField	105
Method getGuild	106
Method getId	106
Method getLevel	106
Method getName	107
Method getPlayers	107

Method isLoaded	107
Method load	108
Method save	108
Method setCustomField	108
Method setGuild	109
Method setLevel	110
Method setName	110
Class OTS_GuildRanks_List	111
Constructor construct	111
Method count	111
Method current	112
Method deleteGuildRank	112
Method key	112
Method next	113
Method rewind	113
Method setLimit	113
Method setOffset	114
Method valid	114
Class OTS_Guilds_List	115
Constructor construct	115
Method count	116
Method current	116
Method deleteGuild	116
Method key	117
Method next	117
Method rewind	117
Method setLimit	118
Method setOffset	118
Method valid	119
Class OTS_InfoRespond	119
Method getClientVersion	119
Method getEmail	120
Method getIP	120
Method getLocation	120
Method getMapAuthor	121
Method getMapHeight	121
Method getMapName	121
Method getMapWidth	122
Method getMaxPlayers	122
Method getMonstersCount	122
Method getMOTD	123
Method getName	123
Method getOnlinePlayers	123
Method getOwner	124
Method getPlayersPeak	124
Method getPort	124
Method getServer	125
Method getServerVersion	125
Method getTSPQVersion	125

Method <code>getUptime</code>	126
Method <code>getURL</code>	126
Class <code>OTS_Item</code>	126
Constructor <code>construct</code>	127
Method <code>count</code>	127
Method <code>getAttributes</code>	127
Method <code>getCount</code>	128
Method <code>getId</code>	128
Method <code>setAttributes</code>	128
Method <code>setCount</code>	129
Class <code>OTS_Player</code>	129
Constructor <code>construct</code>	130
Method <code>find</code>	130
Method <code>getAccount</code>	131
Method <code>getCap</code>	131
Method <code>getConditions</code>	131
Method <code>getCustomField</code>	132
Method <code>getDepot</code>	132
Method <code>getDirection</code>	133
Method <code>getExperience</code>	133
Method <code>getGroup</code>	134
Method <code>getGuildNick</code>	134
Method <code>getHealth</code>	134
Method <code>getHealthMax</code>	135
Method <code>getId</code>	135
Method <code>getLastIP</code>	135
Method <code>getLastLogin</code>	136
Method <code>getLevel</code>	136
Method <code>getLookAddons</code>	136
Method <code>getLookBody</code>	137
Method <code>getLookFeet</code>	137
Method <code>getLookHead</code>	137
Method <code>getLookLegs</code>	138
Method <code>getLookType</code>	138
Method <code>getLossExperience</code>	139
Method <code>getLossMana</code>	139
Method <code>getLossSkills</code>	139
Method <code>getMagLevel</code>	140
Method <code>getMana</code>	140
Method <code>getManaMax</code>	140
Method <code>getManaSpent</code>	141
Method <code>getName</code>	141
Method <code>getPosX</code>	141
Method <code>getPosY</code>	142
Method <code>getPosZ</code>	142
Method <code>getPremiumEnd</code>	142
Method <code>getRank</code>	143
Method <code>getRankId</code>	143
Method <code>getRedSkullTime</code>	144

Method <u>getSex</u>	144
Method <u>getSkill</u>	144
Method <u>getSkillTries</u>	145
Method <u>getSlot</u>	145
Method <u>getSoul</u>	146
Method <u>getTownId</u>	146
Method <u>getVocation</u>	147
Method <u>hasRedSkull</u>	147
Method <u>isLoading</u>	147
Method <u>isSaveSet</u>	148
Method <u>load</u>	148
Method <u>save</u>	148
Method <u>setAccount</u>	149
Method <u>setCap</u>	149
Method <u>setConditions</u>	150
Method <u>setCustomField</u>	150
Method <u>setDepot</u>	151
Method <u>setDirection</u>	151
Method <u>setExperience</u>	152
Method <u>setGroup</u>	152
Method <u>setGuildNick</u>	153
Method <u>setHealth</u>	153
Method <u>setHealthMax</u>	154
Method <u>setLastIP</u>	154
Method <u>setLastLogin</u>	154
Method <u>setLevel</u>	155
Method <u>setLookAddons</u>	155
Method <u>setLookBody</u>	156
Method <u>setLookFeet</u>	156
Method <u>setLookHead</u>	156
Method <u>setLookLegs</u>	157
Method <u>setLookType</u>	157
Method <u>setLossExperience</u>	158
Method <u>setLossMana</u>	158
Method <u>setLossSkills</u>	159
Method <u>setMagLevel</u>	159
Method <u>setMana</u>	159
Method <u>setManaMax</u>	160
Method <u>setManaSpent</u>	160
Method <u>setName</u>	161
Method <u>setPosX</u>	161
Method <u>setPosY</u>	161
Method <u>setPosZ</u>	162
Method <u>setPremiumEnd</u>	162
Method <u>setRank</u>	163
Method <u>setRankId</u>	163
Method <u>setRedSkull</u>	164
Method <u>setRedSkullTime</u>	164
Method <u>setSave</u>	164

Method setSex	165
Method setSkill	165
Method setSkillTries	166
Method setSlot	166
Method setSoul	167
Method setTownId	167
Method setVocation	168
Method unsetRedSkull	168
Method unsetSave	168
Class OTS Players List	169
Constructor construct	169
Method count	169
Method current	170
Method deletePlayer	170
Method key	170
Method next	171
Method rewind	171
Method setLimit	171
Method setOffset	172
Method valid	172
Class POT	173
Class Constant DB MYSQL	173
Class Constant DB ODBC	173
Class Constant DB PGSQL	174
Class Constant DB SQLITE	174
Class Constant DIRECTION EAST	174
Class Constant DIRECTION NORTH	175
Class Constant DIRECTION SOUTH	175
Class Constant DIRECTION WEST	175
Class Constant SEX FEMALE	175
Class Constant SEX MALE	176
Class Constant SKILL AXE	176
Class Constant SKILL CLUB	176
Class Constant SKILL DISTANCE	177
Class Constant SKILL FISHING	177
Class Constant SKILL FIST	178
Class Constant SKILL SHIELDING	178
Class Constant SKILL SWORD	178
Class Constant SLOT AMMO	179
Class Constant SLOT ARMOR	179
Class Constant SLOT BACKPACK	179
Class Constant SLOT FEET	180
Class Constant SLOT HEAD	180
Class Constant SLOT LEFT	181
Class Constant SLOT LEGS	181
Class Constant SLOT NECKLACE	181
Class Constant SLOT RIGHT	182
Class Constant SLOT RING	182
Class Constant VOCATION DRUID	182

Class Constant VOCATION_KNIGHT	183
Class Constant VOCATION_NONE	183
Class Constant VOCATION_PALADIN	183
Class Constant VOCATION_SORCERER	184
Method connect	184
example: connect.php	184
Method createObject	185
Method getInstance	186
Method loadClass	186
Method serverStatus	186
example: example	186
Method setPOTPath	188
example: fakeroot.php	188
compat.php	189
Appendices	190
Appendix A - Class Trees	191
POT	191
Appendix B - README/CHANGELOG/INSTALL	195
CHANGELOG	196
README	196
INSTALL	197
NEWS	198

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 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).

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 
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 history. It is completely other language than 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 though we loaded class that implements Countable interface which does not
18 // exists in PHP 5.0 SPL library, because 'compat' library defines it.
19 $list= POT::getInstance()-> createObject('Players_List');
20
21 // STEP 2: we can do that in every version - count() is in fact just a public method
22 echo $list-> count();
23
24 // STEP 3: it won't work correctly in PHP 5.0 - PHP won't call internal count() method of object, will print trivial
25 // count() evaluation result on object
26 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 manually.

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 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](#). POT makes use of [spl_autoload_register\(\) function](#) to bind own mechanism with it automatically. If you have your own `__autoload` function defined, after including POT class you have to register your function with `spl_autoload_register()` as well.

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 automatically 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 // binds your __autoload code
12 if( function_exists('__autoload') )
13 {
14     spl_autoload_register('__autoload');
15 }
16
17 // includes POT main file
18 include( './classes/OTS.php' );
19
20 // database configuration - can be simply moved to external file, eg. config.php
21 $config= array(
22     'driver' =>  POT::DB_MYSQL,
23     'host' =>    'localhost',
24     'user' =>    'wrzasq',
25     'database' => 'otserv'
26 );
27
28 // creates POT instance (or get existing one)
29 $ots= POT::getInstance();
30 $ots-> connect(null, $config;
31
32 ?>
```

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->    isLoading() )
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. 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 $players> setOffset(2);
23
24 // iterates through selected players
25 foreach($playersas $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 languages 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 environment. 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 guide](#).

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 through 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:

```
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 // sets basic fields
18 $player-> setName('Wrzasq');
19 $player-> setSex(POT::SEX_MALE);
20 $player-> setVocation(POT::VOCATION_KNIGHT);
21 /* etc... */
22
23 /*
24  this is bad! we can't call this now as we dont have object ID assinged yet
25
26  $player->setCustomField('my_field', 2);
27
28  must save before that to get automatic ID:
29  */
30 $player-> save();
31
32 // now we can call that:
33 // 2 won't be quoted - it's integer
34 $player-> setCustomField('my_field', 2);
35 // 3 will be quoted - '3' is a string!
36 $player-> setCustomField('another_field', '3');
37
38 ?>
```

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:

```
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 $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.

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 around it with simple code:

```
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 // your non-random number
15 $number= 123456;
16
17 // creates new OTS_Account object
18 $account= $ots->createObject('Account');
19 $account->load($number);
20
21 // number is busy
22 if( $account->isLoaded() )
23 {
24     echo 'Account number ', $number, 'is used.', "\n" ;
25 }
26 // it is not
27 else
28 {
29     // generate number from exactly $number - $number range
30     $number= $account->create($number, $number);
31     echo 'Your account number is: ', $number, "\n" ;
32 }
33
34 ?>
```

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** 0.0.3+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.3+SVN
- **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** 0.0.1
- **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](#)

IOTS_GuildAction.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com >
- **Version** 0.0.3+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.3+SVN
- **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
- **Version** 0.0.3+SVN
- **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
- **Version** 0.0.3+SVN
- **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
- **Version** 0.0.3
- **Copyright** 2007 (C) by Wrzasq
- **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** 0.0.1
- **Copyright** 2007 (C) by Wrzasq
- **License** [GNU Lesser General Public License, Version 3](#)

OTS_DB_ODBC.php

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

OTS_DB_PostgreSQL.php

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

OTS_DB_SQLite.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com >
- **Version** 0.0.1
- **Version** 0.0.3+SVN
- **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
- **Version** 0.0.3
- **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
- **Version** 0.0.3
- **Copyright** 2007 (C) by Wrzasq
- **License** [GNU Lesser General Public License, Version 3](#)

OTS_Guild.php

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

OTS_GuildRank.php

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

OTS_GuildRanks_List.php

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

OTS_Guilds_List.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.3+SVN
- **Copyright** 2007 (C) by Wrzasq
- **Since** 0.0.3+SVN
- **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** 0.0.1
- **Version** 0.0.3+SVN
- **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
- **Version** 0.0.3
- **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 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** 0.0.3+SVN
- **Since** 0.0.3+SVN

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 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 IOTS_GuildAction

[line 24]

Guild action interface.

Guild action interface.

This interface indicates that class can handle OTServ guild action.
You can use it for example to handle invites or membership requests.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function IOTS_GuildAction::__construct(\$guild) [line 33]

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 recommended that your implementations calls assignment functions of \$guild to automatically assign itself as action handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function IOTS_GuildAction::addRequest(\$player) [*line 46*]

Function Parameters:

- [*OTS_Player*](#) **\$player** Player which is object of request.

Adds new request.

Adds new request.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function IOTS_GuildAction::deleteRequest(\$player) [*line 52*]

Function Parameters:

- [*OTS_Player*](#) **\$player** Player which is object of request.

Deletes request.

Deletes request.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

array function IOTS_GuildAction::listRequests() [*line 40*]

List of saved pending actions.

List of saved pending actions.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function IOTS_GuildAction::submitRequest(\$player) [*line 58*]

Function Parameters:

- [*OTS Player*](#) **\$player** Player which is object of request.

Finalizes request.

Finalizes request.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

Class OTS_Account

[line 21]

OTServ account abstraction.

OTServ account abstraction.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3+SVN

Constructor *void* function OTS_Account::__construct(\$db) [line 42]

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 328]

Blocks account.

Blocks account.

- **Version** 0.0.1

- **Access public**

int function `OTS_Account::create([$min = 1], [$max = 9999999])` [line 67]

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!

IMPORTANT: Since 0.0.3+SVN 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** 0.0.3+SVN
- **Version** 0.0.1
- **Throws** Exception When there are no free account numbers.
- **Access** public
- **Example**

int function `OTS_Account::createEx($group, [$min = 1], [$max = 9999999])` [*line 93*]
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 // group for account
18 $group = $ots->createObject('Group');
19
20 // loads group with id 1
21 $group->load(1);
22
23 // generates new account number
24 $number = $account->createEx($group);
25
26 // give user his number
27 echo 'Your account number is: ', $number;
28
29 ?>

```

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** 0.0.3+SVN
- **Version** 0.0.1
- **Throws** Exception When there are no free account numbers.
- **Since** 0.0.3+SVN
- **Access** public
- **Example**

void function OTS_Account::find(\$email) [*line 160*]

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** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2
- **Access** public

string function OTS_Account::getCustomField(\$field) [*line 375*]

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

string function OTS_Account::getEmail() [*line 280*]

E-mail address.

E-mail address.

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

OTS_Group function OTS_Account::getGroup() [*line 224*]

Returns group of this account.

Returns group of this account.

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

int function OTS_Account::getId() [*line 206*]

Account number.

Account number.

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

int function OTS_Account::getPACCDays() [*line 341*]

PACC days.

PACC days.

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

string function OTS_Account::getPassword() [*line 253*]

Account's password.

Account's password.

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

array function OTS_Account::getPlayers() [*line 424*]

List of characters on account.

List of characters on account.

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

bool function OTS_Account::isBlocked() [*line 307*]

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.
- **Access** public

bool function OTS_Account::isLoaded() [*line 177*]

Checks if object is loaded.

Checks if object is loaded.

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

void function OTS_Account::load(\$id) [*line 147*]

Function Parameters:

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

Loads account with given number.

Loads account with given number.

- **Version** 0.0.3+SVN
- **Version** 0.0.1
- **Access** public

void function OTS_Account::save() [*line 188*]

Updates account in database.

Updates account in database.

- **Version** 0.0.3+SVN

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

void function OTS_Account::setCustomField(\$field, \$value) [line 401]

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

void function OTS_Account::setEMail(\$email) [line 295]

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::setGroup(\$group) [*line 241*]

Function Parameters:

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

Assigns account to group.

Assigns account to group.

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

void function OTS_Account::setPACCDays(\$premdays, \$pacc) [*line 358*]

Function Parameters:

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

Sets PACC days count.

Sets PACC days count.

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

void function OTS_Account::setPassword(\$password) [*line 268*]

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 320*]

Unlocks account.

Unlocks account.

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

Class OTS_Accounts_List

[line 21]

List of accounts.

List of accounts.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3

Constructor *void* function OTS_Accounts_List::__construct(\$db) [line 56]

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 161]

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 111*]

Returns current row.

Returns current row.

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

void function OTS_Accounts_List::deleteAccount(\$account) [*line 101*]

Function Parameters:

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

Deletes account.

Deletes account.

- **Version** 0.0.3
- **Version** 0.0.1
- **Access** public

mixed function OTS_Accounts_List::key() [*line 133*]

Current cursor position.

Current cursor position.

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

void function OTS_Accounts_List::next() [line 123]

Moves to next row.

Moves to next row.

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

void function OTS_Accounts_List::rewind() [line 151]

Select accounts from database.

Select accounts from database.

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

void function OTS_Accounts_List::setLimit([\$limit = false]) [line 66]

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 83*]

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 143*]

Checks if there are any rows left.

Checks if there are any rows left.

- **Version** 0.0.1
- **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 implementation. [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 than 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 exactly 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 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 againsts 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_ODBC

[*line 20*]

ODBC connection interface.

ODBC connection interface.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function OTS_DB_ODBC::__construct(\$params) [*line 47*]

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** 0.0.3+SVN
- **See** [POT::connect\(\)](#)
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_ODBC::fieldName(\$name) [*line 95*]

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_ODBC::limit([\$limit = false], [\$offset = false]) [*line 146*]

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** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

PDOStatement|bool function OTS_DB_ODBC::SQLquery(\$query) [*line 134*]

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_ODBC::SQLquote(\$string) [*line 120*]

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** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_ODBC::tableName(\$name) [*line 106*]

Function Parameters:

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

Query-quoted table name.

Query-quoted table name.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

Class OTS_DB_PostgreSQL

[line 20]

PostgreSQL connection interface.

PostgreSQL connection interface.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function OTS_DB_PostgreSQL::__construct(\$params) [line 47]

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** 0.0.3+SVN
- **See** [POT::connect\(\)](#)
- **Since** 0.0.3+SVN

- **Access** public

string function OTS_DB_PostgreSQL::fieldName(\$name) [*line 102*]

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_PostgreSQL::limit([\$limit = false], [\$offset = false]) [*line 153*]

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** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

PDOStatement|bool function OTS_DB_PostgreSQL::SQLquery(\$query) [*line 141*]

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_PostgreSQL::SQLquote(\$string) [*line 127*]

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** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_DB_PostgreSQL::tableName(\$name) [*line 113*]

Function Parameters:

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

Query-quoted table name.

Query-quoted table name.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

Class OTS_DB_SQLite

[line 21]

SQLite connection interface.

SQLite connection interface.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3+SVN

Constructor *void* function OTS_DB_SQLite::__construct(\$params) [line 44]

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 115*]

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_SQLite::SQLquery(\$query) [*line 103*]

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 89*]

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_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 21*]

OTServ user group abstraction.

OTServ user group abstraction.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3

Constructor *void* function OTS_Group::__construct(\$db) [line 42]

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_Group::getAccess() [line 167]

Access level.

Access level.

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

string function OTS_Group::getCustomField(\$field) [line 254]

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

int function OTS_Group::getFlags() [*line 140*]

Rights flags.

Rights flags.

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

int function OTS_Group::getId() [*line 96*]

Group ID.

Group ID.

- **Version** 0.0.3

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

int function OTS_Group::getMaxDepotItems() [*line 194*]

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.
- **Access** public

int function OTS_Group::getMaxVIPList() [*line 221*]

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.
- **Access** public

string function OTS_Group::getName() [*line 113*]

Group name.

Group name.

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

array function OTS_Group::getPlayers() [*line 303*]

List of characters in given group.

List of characters in given group.

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

bool function OTS_Group::isLoading() [*line 63*]

Checks if object is loaded.

Checks if object is loaded.

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

void function OTS_Group::load(\$id) [*line 52*]

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 71]

Saves account in database.

Saves account in database.

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

void function OTS_Group::setAccess(\$access) [line 182]

Function Parameters:

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

Sets access level.

Sets access level.

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

void function OTS_Group::setCustomField(\$field, \$value) [line 280]

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

void function OTS_Group::setFlags(\$flags) [line 155]

Function Parameters:

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

Sets rights flags.

Sets rights flags.

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

void function OTS_Group::setMaxDepotItems(\$maxdepotitems) [line 209]

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 236]

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 128*]

Function Parameters:

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

Sets group's name.

Sets group's name.

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

Class OTS_Groups_List

[*line 21*]

List of groups.

List of groups.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3

Constructor *void* function OTS_Groups_List::__construct(\$db) [*line 56*]

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 161*]

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 111*]

Returns current row.

Returns current row.

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

void function OTS_Groups_List::deleteGroup(\$group) [*line 101*]

Function Parameters:

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

Deletes group.

Deletes group.

- **Version** 0.0.3
- **Version** 0.0.1
- **Access** public

mixed function OTS_Groups_List::key() [*line 133*]

Current cursor position.

Current cursor position.

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

void function OTS_Groups_List::next() [*line 123*]

Moves to next row.

Moves to next row.

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

void function OTS_Groups_List::rewind() [*line 151*]

Select groups from database.

Select groups from database.

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

void function OTS_Groups_List::setLimit([\$limit = false]) [*line 66*]

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 83*]

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 143*]

Checks if there are any rows left.

Checks if there are any rows left.

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

Class OTS_Guild

[*line 20*]

OTServ guild abstraction.

OTServ guild abstraction.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function OTS_Guild::__construct(\$db) [*line 55*]

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::acceptInvite(\$player) [line 388]

Function Parameters:

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

Finalise invitation.

Finalise invitation.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::acceptRequest(\$player) [line 480]

Function Parameters:

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

Accepts player.

Accepts player.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::deleteInvite(\$player) [line 365]

Function Parameters:

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

Deletes invitation for player to guild.

Deletes invitation for player to guild.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::deleteRequest(\$player) [line 457]

Function Parameters:

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

Deletes request from player.

Deletes request from player.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::find(\$name) [*line 96*]

Function Parameters:

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

Loads guild by it's name.

Loads guild by it's name.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

int function OTS_Guild::getCreationData() [*line 215*]

Guild creation data.

Guild creation data.

- **Version** 0.0.3+SVN

- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_Guild::getCustomField(\$field) [*line 246*]

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** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.3+SVN
- **Access** public

array function OTS_Guild::getGuildRanks() [*line 292*]

Reads all ranks that are in this guild.

Reads all ranks that are in this guild.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.3+SVN

- **Access** public

int function OTS_Guild::getId() [*line 145*]

Guild ID.

Guild ID.

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

string function OTS_Guild::getName() [*line 161*]

Guild name.

Guild name.

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

OTS_Player function OTS_Guild::getOwner() [*line 187*]

Returns owning player of this player.

Returns owning player of this player.

- **Version** 0.0.3+SVN

- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::invite(\$player) [*line 342*]

Function Parameters:

- [*OTS Player*](#) **\$player** Player to be invited.

Invites player to guild.

Invites player to guild.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

bool function OTS_Guild::isLoaded() [*line 113*]

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

array function OTS_Guild::listInvites() [*line 319*]

Returns list of invited players.

Returns list of invited players.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no invites driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

array function OTS_Guild::listRequests() [*line 411*]

Returns list of players that requested membership.

Returns list of players that requested membership.

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::load(\$id) [*line 85*]

Function Parameters:

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

Loads guild with given id.

Loads guild with given id.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::request(\$player) [line 434]

Function Parameters:

- [*OTS_Player*](#) **\$player** Player that requested membership.

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

- **Version** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Throws** E_OTS_NoDriver If there is no requests driver assigned.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::save() [line 121]

Saves guild in database.
Saves guild in database.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

- **Access** public

void function OTS_Guild::setCreationData(\$creationdata) [line 230]

Function Parameters:

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

Sets guild creation data.

Sets guild creation data.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::setCustomField(\$field, \$value) [line 270]

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 improper 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** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If guild is not loaded.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::setInvitesDriver([\$invites = null]) [*line 65*]

Function Parameters:

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

Assigns invites handler.

Assigns invites handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::setName(\$name) [*line 176*]

Function Parameters:

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

Sets players's name.

Sets players's name.

- **Version** 0.0.3+SVN

- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::setOwner(\$owner) [line 204]

Function Parameters:

- [*OTS Player*](#) **\$owner** Owning player.

Assigns guild to owner.

Assigns guild to owner.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guild::setRequestsDriver([\$requests = null]) [line 75]

Function Parameters:

- [*IOTS GuildAction*](#) **\$requests** Membership requests driver (don't pass it to clear driver).

Assigns requests handler.

Assigns requests handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

Class OTS_GuildRank

[line 20]

OTServ guild rank abstraction.

OTServ guild rank abstraction.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function OTS_GuildRank::__construct(\$db) [line 41]

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRank::find(\$name, [\$guild = null]) [line 65]

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** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

string function OTS_GuildRank::getCustomField(\$field) [*line 223*]

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** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.3+SVN
- **Access** public

OTS_Guild function *OTS_GuildRank::getGuild()* [*line 164*]

Returns guild of this rank.

Returns guild of this rank.

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

int function *OTS_GuildRank::getId()* [*line 122*]

Rank ID.

Rank ID.

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

int function *OTS_GuildRank::getLevel()* [*line 192*]

Rank's access level.

Rank's access level.

- **Version** 0.0.3+SVN
- **Throws** *E_OTS_NotLoaded* If rank is not loaded.

- **Since** 0.0.3+SVN
- **Access** public

string function OTS_GuildRank::getName() [*line 138*]

Rank name.

Rank name.

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

array function OTS_GuildRank::getPlayers() [*line 269*]

Reads all players who has this rank set.

Reads all players who has this rank set.

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

bool function OTS_GuildRank::isLoaded() [*line 90*]

Checks if object is loaded.

Checks if object is loaded.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRank::load(\$id) [line 51]

Function Parameters:

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

Loads rank with given id.

Loads rank with given id.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRank::save() [line 98]

Saves rank in database.

Saves rank in database.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRank::setCustomField(\$field, \$value) [line 247]

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** 0.0.3+SVN
- **Throws** E_OTS_NotLoaded If rank is not loaded.
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRank::setGuild(\$guild) [*line 181*]

Function Parameters:

- [*OTS_Guild*](#) **\$guild** Owing guild.

Assigns rank to guild.

Assigns rank to guild.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

- **Access** public

void function OTS_GuildRank::setLevel(\$level) [line 207]

Function Parameters:

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

Sets rank's access level within guild.

Sets rank's access level within guild.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRank::setName(\$name) [line 153]

Function Parameters:

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

Sets rank's name.

Sets rank's name.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

Class OTS_GuildRanks_List

[line 20]

List of guild ranks.

List of guild ranks.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function OTS_GuildRanks_List::__construct(\$db) [line 55]

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

int function OTS_GuildRanks_List::count() [line 159]

Returns number of ranks on list in current criterium.

Returns number of ranks on list in current criterium.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

OTS_GuildRank function OTS_GuildRanks_List::current() [*line 109*]

Returns current row.

Returns current row.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRanks_List::deleteGuildRank(\$guildRank) [*line 99*]

Function Parameters:

- [*OTS_GuildRank*](#) **\$guildRank** Rank to be deleted.

Deletes guild rank.

Deletes guild rank.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

mixed function OTS_GuildRanks_List::key() [*line 131*]

Current cursor position.
Current cursor position.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRanks_List::next() [line 121]

Moves to next row.

Moves to next row.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRanks_List::rewind() [line 149]

Select ranks from database.

Select ranks from database.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRanks_List::setLimit([\$limit = false]) [line 65]

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_GuildRanks_List::setOffset([\$offset = false]) [*line 82*]

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

bool function OTS_GuildRanks_List::valid() [*line 141*]

Checks if there are any rows left.

Checks if there are any rows left.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

Class OTS_Guilds_List

[line 20]

List of guilds.
List of guilds.

- **Package** POT
- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN

Constructor *void* function OTS_Guilds_List::__construct(\$db) [line 55]
Function Parameters:

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

Sets database connection handler.
Sets database connection handler.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

int function OTS_Guilds_List::count() [*line 159*]

Returns number of guilds on list in current criterium.

Returns number of guilds on list in current criterium.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

OTS_Guild function OTS_Guilds_List::current() [*line 109*]

Returns current row.

Returns current row.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guilds_List::deleteGuild(\$guild) [*line 99*]

Function Parameters:

- [OTS_Guild](#) **\$guild** Guild to be deleted.

Deletes guild.

Deletes guild.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

mixed function OTS_Guilds_List::key() [*line 131*]

Current cursor position.

Current cursor position.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guilds_List::next() [*line 121*]

Moves to next row.

Moves to next row.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guilds_List::rewind() [*line 149*]

Select guilds from database.

Select guilds from database.

- **Version** 0.0.3+SVN

- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guilds_List::setLimit([\$limit = false]) [line 65]

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

void function OTS_Guilds_List::setOffset([\$offset = false]) [line 82]

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+SVN
- **Access** public

bool function OTS_Guilds_List::valid() [*line 141*]

Checks if there are any rows left.

Checks if there are any rows left.

- **Version** 0.0.3+SVN
- **Since** 0.0.3+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 exactly 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 21*]

OTServ character abstraction.

OTServ character abstraction.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3+SVN

Constructor *void* function OTS_Player::__construct(\$db) [*line 52*]

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 84*]

Function Parameters:

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

Loads player by it's name.

Loads player by it's name.

- **Version** 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.
- **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.
- **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.
- **Access** public

string function OTS_Player::getCustomField(\$field) [*line 1252*]

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

OTS_Item|null function OTS_Player::getDepot(\$depot) [*line 1517*]

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

- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.

- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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
- **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.
- **Access** public

int function OTS_Player::getRankId() [*line 1070*]

Guild rank ID.

Guild rank ID.

- **Version** 0.0.3
- **Version** 0.0.1
- **Deprecated** 0.0.3+SVN Use getRank().
- **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.
- **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.
- **Access** public

int function OTS_Player::getSkill(\$skill) [*line 1303*]

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
- **Access** public

int function OTS_Player::getSkillTries(\$skill) [*line 1335*]

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
- **Access** public

OTS_Item|null function OTS_Player::getSlot(\$slot) [*line 1388*]

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

bool function OTS_Player::isLoading() [*line 101*]

Checks if object is loaded.

Checks if object is loaded.

- **Version** 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.
- **Access** public

void function OTS_Player::load(\$id) [*line 63*]

Function Parameters:

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

Loads player with given id.

Loads player with given id.

- **Version** 0.0.2
- **Version** 0.0.1
- **Access** public

void function OTS_Player::save() [*line 111*]

Saves player in database.

Saves player in database.

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

void function OTS_Player::setCap(\$cap) [line 856]

Function Parameters:

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

Sets capacity.

Sets capacity.

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

void function OTS_Player::setCustomField(\$field, \$value) [line 1278]

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

void function OTS_Player::setDepot(\$depot, [\$item = null], [\$pid = 0], [\$depot_id = 0]) [*line 1572*]

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** For internal recursive insertion.
- *int* **\$depot_id** Internal, for further use.

Sets slot content.

Sets slot content.

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

void function OTS_Player::setLookAddons(\$lookaddons) [line 748]

Function Parameters:

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

Sets addons.

Sets addons.

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

void function OTS_Player::setLookType(\$looktype) [*line 721*]

Function Parameters:

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

Sets outfit.

Sets outfit.

- **Version** 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
- **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
- **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
- **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
- **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
- **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
- **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

- **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
- **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
- **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
- **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
- **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** 0.0.3
- **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
- **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** 0.0.3+SVN Use setRank().
- **Access** public

void function OTS_Player::setRedSkull() [line 1030]

Sets red skull flag.

Sets red skull flag.

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

void function OTS_Player::setSave() [line 943]

Sets save flag.

Sets save flag.

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

void function OTS_Player::setSkill(\$skill, \$value) [line 1321]

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** 0.0.2
- **Access** public

void function OTS_Player::setSkillTries(\$skill, \$tries) [*line 1353*]

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** 0.0.2
- **Access** public

void function OTS_Player::setSlot(\$slot, [\$item = null], [\$pid = 0]) [*line 1442*]

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** For internal use in case of containers.

Sets slot content.

Sets slot content.

- **Version** 0.0.3

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

void function OTS_Player::unsetRedSkull() [line 1022]

Unsets red skull flag.

Unsets red skull flag.

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

void function OTS_Player::unsetSave() [line 935]

Unsets save flag.

Unsets save flag.

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

Class OTS_Players_List

[line 21]

List of players.

List of players.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3

Constructor *void* function OTS_Players_List::__construct(\$db) [line 56]

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 161]

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 111*]

Returns current row.

Returns current row.

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

void function OTS_Players_List::deletePlayer(\$player) [*line 101*]

Function Parameters:

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

Deletes player.

Deletes player.

- **Version** 0.0.3
- **Version** 0.0.1
- **Access** public

mixed function OTS_Players_List::key() [*line 133*]

Current cursor position.

Current cursor position.

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

void function OTS_Players_List::next() [line 123]

Moves to next row.

Moves to next row.

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

void function OTS_Players_List::rewind() [line 151]

Select players from database.

Select players from database.

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

void function OTS_Players_List::setLimit([\$limit = false]) [line 66]

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 83*]

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 143*]

Checks if there are any rows left.

Checks if there are any rows left.

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

Class POT

[line 23]

Main POT class.

Main POT class.

- **Package** POT
- **Version** 0.0.1
- **Version** 0.0.3+SVN

POT::DB_MYSQL

= 1 [line 28]

MySQL driver.

MySQL driver.

- **Version** 0.0.1

POT::DB_ODBC

= 4 [line 46]

ODBC driver.

ODBC driver.

- **Version** 0.0.3+SVN
- **Version** 0.0.1
- **Since** 0.0.3+SVN

POT::DB_PGSQL

= 3 *[line 39]*

PostgreSQL driver.

PostgreSQL driver.

- **Version** 0.0.3+SVN
- **Version** 0.0.1
- **Since** 0.0.3+SVN

POT::DB_SQLITE

= 2 *[line 32]*

SQLite driver.

SQLite driver.

- **Version** 0.0.1

POT::DIRECTION_EAST

= 1 *[line 85]*

East.

East.

- **Version** 0.0.1

POT::DIRECTION_NORTH

= 0 *[line 81]*

North.

North.

- **Version 0.0.1**

POT::DIRECTION_SOUTH

= 2 *[line 89]*

South.

South.

- **Version 0.0.1**

POT::DIRECTION_WEST

= 3 *[line 93]*

West.

West.

- **Version 0.0.1**

POT::SEX_FEMALE

= 0 *[line 51]*

Female gender.
Female gender.

- **Version** 0.0.1

POT::SEX_MALE

= 1 [*line 55*]

Male gender.
Male gender.

- **Version** 0.0.1

POT::SKILL_AXE

= 3 [*line 122*]

Axe fighting.
Axe fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SKILL_CLUB

= 1 [*line 108*]

Club fighting.

Club fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SKILL_DISTANCE

= 4 [*line 129*]

Distance fighting.

Distance fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SKILL_FISHING

= 6 [*line 143*]

Fishing.

Fishing.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SKILL_FIST

= 0 *[line 101]*

Fist fighting.
Fist fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SKILL_SHIELDING

= 5 *[line 136]*

Shielding.
Shielding.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SKILL_SWORD

= 2 *[line 115]*

Sword fighting.
Sword fighting.

- **Version** 0.0.2
- **Version** 0.0.1
- **Since** 0.0.2

POT::SLOT_AMMO

= 10 [*line 214*]

Ammunition slot.

Ammunition slot.

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

POT::SLOT_ARMOR

= 4 [*line 172*]

Armor slot.

Armor slot.

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

POT::SLOT_BACKPACK

= 3 [*line 165*]

Backpack slot.

Backpack slot.

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

POT::SLOT_FEET

= 8 *[line 200]*

Boots slot.

Boots slot.

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

POT::SLOT_HEAD

= 1 *[line 151]*

Head slot.

Head slot.

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

POT::SLOT_LEFT

= 6 [*line 186*]

Left hand slot.

Left hand slot.

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

POT::SLOT_LEGS

= 7 [*line 193*]

Legs slot.

Legs slot.

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

POT::SLOT_NECKLACE

= 2 [*line 158*]

Necklace slot.

Necklace slot.

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

POT::SLOT_RIGHT

= 5 [*line 179*]

Right hand slot.

Right hand slot.

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

POT::SLOT_RING

= 9 [*line 207*]

Ring slot.

Ring slot.

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

POT::VOCATION_DRUID

= 2 [*line 68*]

Druid.
Druid.

- **Version 0.0.1**

POT::VOCATION_KNIGHT

= 4 [*line 76*]

Knight.
Knight.

- **Version 0.0.1**

POT::VOCATION_NONE

= 0 [*line 60*]

None vocation.
None vocation.

- **Version 0.0.1**

POT::VOCATION_PALADIN

= 3 [*line 72*]

Paladin.
Paladin.

- **Version 0.0.1**

POT::VOCATION_SORCERER

= 1 [line 64]

Sorcerer.

Sorcerer.

- **Version 0.0.1**

void function POT::connect(\$driver, \$params) [line 333]

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**

LOTS_DAO function POT::createObject(\$class) [*line 376*]

Function Parameters:

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

Creates OTServ DAO class instance.

Creates OTServ DAO class instance.

- **Version** 0.0.1

- **Access** public

POT function POT::getInstance() [*line 221*]

Singleton.

Singleton.

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

void function POT::loadClass(\$class) [*line 293*]

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
- **Access** public
- **Example** example not found

OTS_InfoRespond|bool function POT::serverStatus(\$server, \$port) [*line 394*]

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** 0.0.1
- **Version** 0.0.2
- **Since** 0.0.2
- **Access** public
- **Example**

void function POT::setPOTPath(\$path) [line 252]

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**

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
- **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_Account

- [OTS_Account](#)

OTS_Accounts_List

- [OTS_Accounts_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_Group

- [OTS_Group](#)

OTS_Groups_List

- [OTS_Groups_List](#)

OTS_Guild

- [OTS_Guild](#)

OTS_GuildRank

- [OTS_GuildRank](#)

OTS_GuildRanks_List

- [OTS_GuildRanks_List](#)

OTS_Guilds_List

- [OTS_Guilds_List](#)

OTS_InfoRespond

- DOMDocument
 - [OTS_InfoRespond](#)

OTS_Item

- [OTS_Item](#)
 - [OTS_Container](#)

OTS_Player

- [OTS_Player](#)

OTS_Players_List

- [OTS_Players_List](#)

POT

- [POT](#)

Appendix B - README/CHANGELOG/INSTALL

CHANGELOG

[0.0.3+SVN]

- * 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 reserved in OTServ for further use). <wrzasq>
- * Added PostgreSQL and ODBC drivers. <wrzasq>
- * Updated players table structure. <wrzasq>
- * Dropped REGEXP operator bindings - not used anywhere. <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 instantiated 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 automatically 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 language 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.
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.
documentation.pdf - phpDocumentor-generator 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.zip file for distribution purposes.

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

===== Credits =====

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

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

What's new in 0.0.3 version?

- * Added custom fields support.

You can now use POT with non-standard SVN database structure (however it is not as comfortable as with standard SVN fields). You have to save your standard record before saving custom fields.

- * Added items and depots support.

OTS_Item and OTS_Container classes. OTS_Player now has getSlot(), setSlot(), getDepot(), setDepot() methods. You can manage items tables as objects trees.

- * Added support for players PACC timestamps.

In current OTServ SVN premium time is not stored in accounts table, but in players table also not as days, but as ending moment timestamp. Account PACC methods are now obsolete.

- * Fixed loading skills.

Small typo.

- * Replaced E_USER_* with exceptions.

No more error messages between text on website, everything is now thrown as exceptions.

- * Uses fetchAll() in loops to prevent MySQL buffering problems.

PDO is really fucked up in some places and MySQL driver queries buffering is one of them. This change should prevent POT from producing some errors in very particular situations.

Index

A

[Account number hack](#) 13

C

[constructor OTS Guilds List:: construct\(\)](#) 115
 Sets database connection handler.
[constructor OTS GuildRanks List:: construct\(\)](#) 111
 Sets database connection handler.
[constructor OTS GuildRank:: construct\(\)](#) 104
 Sets database connection handler.
[constructor OTS Guild:: construct\(\)](#) 92
 Sets database connection handler.
[constructor OTS Item:: construct\(\)](#) 127
 Creates item of given ID.
[constructor OTS Player:: construct\(\)](#) 130
 Sets database connection handler.
[CHANGELOG](#) 196
[compat.php](#) 189
 POT compatibility assurance package.
[constructor OTS Players List:: construct\(\)](#) 169
 Sets database connection handler.
[constructor OTS Groups List:: construct\(\)](#) 88
 Sets database connection handler.
[constructor OTS Group:: construct\(\)](#) 81
 Sets database connection handler.
[constructor OTS Account:: construct\(\)](#) 49
 Sets database connection handler.
[constructor IOTS GuildAction:: construct\(\)](#) 46
 Objects are initialized with a guild that they are assigned to.
[constructor IOTS DB:: construct\(\)](#) 43
 Connection parameters.
[constructor OTS Accounts List:: construct\(\)](#) 60
 Sets database connection handler.
[constructor OTS DB MySQL:: construct\(\)](#) 67
 Creates database connection.
[constructor OTS DB SQLite:: construct\(\)](#) 77
 Creates database connection.
[constructor OTS DB PostgreSQL:: construct\(\)](#) 74
 Creates database connection.
[constructor OTS DB ODBC:: construct\(\)](#) 70
 Creates database connection.
[constructor IOTS DAO:: construct\(\)](#) 42
 DAO objects must be initialized with a database.

D

DAO objects	9
-----------------------------	---

E

E_OTs_NotLoaded	41
<i>Occurs when code attempts to access property of not loaded object.</i>	
E_OTs_NoDriver	41
<i>Occurs when code attempts to execute driven action that has no assigned driver to handle it.</i>	
E_OTs_NotLoaded.php	18
E_OTs_NoDriver.php	17

I

IOTS_GuildAction::addRequest()	47
<i>Adds new request.</i>	
IOTS_GuildAction	46
<i>Guild action interface.</i>	
IOTS_DB::tableName()	45
<i>Query-quoted table name.</i>	
IOTS_GuildAction::deleteRequest()	47
<i>Deletes request.</i>	
IOTS_GuildAction::listRequests()	48
<i>List of saved pending actions.</i>	
INSTALL	197
IOTS_GuildAction::submitRequest()	48
<i>Finalizes request.</i>	
IOTS_DB::SQLquote()	45
<i>Query-quoted string value.</i>	
IOTS_DB::SQLquery()	45
<i>Evaluates query.</i>	
IOTS_DAO	42
<i>OTServ database object.</i>	
IOTS_GuildAction.php	21
IOTS_DB.php	20
IOTS_DB	43
<i>OTServ database handler interface.</i>	
IOTS_DB::fieldName()	43
<i>Query-quoted field name.</i>	
IOTS_DB::limit()	44
<i>LIMIT/OFFSET clause for queries.</i>	
IOTS_DB::lastInsertId()	44
<i>ID of last created record.</i>	
IOTS_DAO.php	19

N

NEWS	198
----------------------	-----

O

OTS_Player::getConditions()	131
<i>Conditions.</i>	
OTS_Player::getCap()	131
<i>Capacity.</i>	
OTS_Player::getAccount()	131
<i>Returns account of this player.</i>	
OTS_Player::getCustomField()	132
<i>Reads custom field.</i>	
OTS_Player::getDepot()	132
<i>Returns items tree from given depot.</i>	
OTS_Player::getGroup()	134
<i>Returns group of this player.</i>	
OTS_Player::getExperience()	133
<i>Experience points.</i>	
OTS_Player::getDirection()	133
<i>Looking direction.</i>	
OTS_Player::find()	130
<i>Loads player by it's name.</i>	
OTS_Player	129
<i>OTServ character abstraction.</i>	
OTS_Item::count()	127
<i>Count value for current item.</i>	
OTS_Item	126
<i>Single item representation.</i>	
OTS_InfoRespond::getURL()	126
<i>Returns server website.</i>	
OTS_Item::getAttributes()	127
<i>Returns item custom attributes.</i>	
OTS_Item::getCount()	128
<i>Returns count of item.</i>	
OTS_Item::setCount()	129
<i>Sets count of item.</i>	
OTS_Item::setAttributes()	128
<i>Sets item attributes.</i>	
OTS_Item::getId()	128
<i>Returns item type.</i>	
OTS_Player::getGuildNick()	134
<i>Guild nick.</i>	
OTS_Player::getHealth()	134
<i>Current HP.</i>	
OTS_Player::getLossExperience()	139
<i>Percentage of experience lost after dead.</i>	
OTS_Player::getLookType()	138
<i>Outfit.</i>	
OTS_Player::getLookLegs()	138
<i>Legs color.</i>	
OTS_Player::getLossMana()	139
<i>Percentage of used mana lost after dead.</i>	
OTS_Player::getLossSkills()	139
<i>Percentage of skills lost after dead.</i>	
OTS_Player::getManaMax()	140

<i>Maximum mana.</i>	
OTS_Player::getMana()	140
<i>Current mana.</i>	
OTS_Player::getMagLevel()	140
<i>Magic level.</i>	
OTS_Player::getLookHead()	137
<i>Hair color.</i>	
OTS_Player::getLookFeet()	137
<i>Boots color.</i>	
OTS_Player::getLastIP()	135
<i>Last login IP.</i>	
OTS_Player::getId()	135
<i>Player ID.</i>	
OTS_Player::getHealthMax()	135
<i>Maximum HP.</i>	
OTS_Player::getLastLogin()	136
<i>Last login timestamp.</i>	
OTS_Player::getLevel()	136
<i>Experience level.</i>	
OTS_Player::getLookBody()	137
<i>Body color.</i>	
OTS_Player::getLookAddons()	136
<i>Addons.</i>	
OTS_InfoRespond::getUptime()	126
<i>Returns server uptime.</i>	
OTS_InfoRespond::getTSPQVersion()	125
<i>Returns version of root element.</i>	
OTS_Guilds_List::key()	117
<i>Current cursor position.</i>	
OTS_Guilds_List::deleteGuild()	116
<i>Deletes guild.</i>	
OTS_Guilds_List::current()	116
<i>Returns current row.</i>	
OTS_Guilds_List::next()	117
<i>Moves to next row.</i>	
OTS_Guilds_List::rewind()	117
<i>Select guilds from database.</i>	
OTS_Guilds_List::valid()	119
<i>Checks if there are any rows left.</i>	
OTS_Guilds_List::setOffset()	118
<i>Sets OFFSET.</i>	
OTS_Guilds_List::setLimit()	118
<i>Sets LIMIT.</i>	
OTS_Guilds_List::count()	116
<i>Returns number of guilds on list in current criterium.</i>	
OTS_Guilds_List	115
<i>List of guilds.</i>	
OTS_GuildRanks_List::next()	113
<i>Moves to next row.</i>	
OTS_GuildRanks_List::key()	112
<i>Current cursor position.</i>	
OTS_GuildRanks_List::deleteGuildRank()	112
<i>Deletes guild rank.</i>	

OTS_GuildRanks_List::rewind()	113
<i>Select ranks from database.</i>	
OTS_GuildRanks_List::setLimit()	113
<i>Sets LIMIT.</i>	
OTS_GuildRanks_List::valid()	114
<i>Checks if there are any rows left.</i>	
OTS_GuildRanks_List::setOffset()	114
<i>Sets OFFSET.</i>	
OTS_InfoRespond	119
<i>Wrapper for 'info' respond's DOMDocument.</i>	
OTS_InfoRespond::getClientVersion()	119
<i>Returns dedicated version of client.</i>	
OTS_InfoRespond::getOnlinePlayers()	123
<i>Returns current amount of players online.</i>	
OTS_InfoRespond::getName()	123
<i>Returns server name.</i>	
OTS_InfoRespond::getMOTD()	123
<i>Returns server's Message Of The Day</i>	
OTS_InfoRespond::getOwner()	124
<i>Returns owner name.</i>	
OTS_InfoRespond::getPlayersPeak()	124
<i>Returns record of online players.</i>	
OTS_InfoRespond::getServerVersion()	125
<i>Returns server version.</i>	
OTS_InfoRespond::getServer()	125
<i>Returns server attribute.</i>	
OTS_InfoRespond::getPort()	124
<i>Returns server port.</i>	
OTS_InfoRespond::getMonstersCount()	122
<i>Returns number of all monsters on map.</i>	
OTS_InfoRespond::getMaxPlayers()	122
<i>Returns maximum amount of players online.</i>	
OTS_InfoRespond::getLocation()	120
<i>Returns server location.</i>	
OTS_InfoRespond::getIP()	120
<i>Returns server IP.</i>	
OTS_InfoRespond::getEmail()	120
<i>Returns owner e-mail.</i>	
OTS_InfoRespond::getMapAuthor()	121
<i>Returns map author.</i>	
OTS_InfoRespond::getMapHeight()	121
<i>Returns map height.</i>	
OTS_InfoRespond::getMapWidth()	122
<i>Returns map width.</i>	
OTS_InfoRespond::getMapName()	121
<i>Returns map name.</i>	
OTS_Player::getManaSpent()	141
<i>Mana spent.</i>	
OTS_Player::getName()	141
<i>Player name.</i>	
OTS_Player::setPremiumEnd()	162
<i>Sets player's Premium Account expiration timestamp.</i>	
OTS_Player::setPosZ()	162

<i>Sets Z map coordinate.</i>	
OTS_Player::setPosY()	161
<i>Sets Y map coordinate.</i>	
OTS_Player::setRank()	163
<i>Assigns guild rank.</i>	
OTS_Player::setRankId()	163
<i>Sets guild rank ID.</i>	
OTS_Player::setSave()	164
<i>Sets save flag.</i>	
OTS_Player::setRedSkullTime()	164
<i>Sets red skulled time remained.</i>	
OTS_Player::setRedSkull()	164
<i>Sets red skull flag.</i>	
OTS_Player::setPosX()	161
<i>Sets X map coordinate.</i>	
OTS_Player::setName()	161
<i>Sets players's name.</i>	
OTS_Player::setLossMana()	158
<i>Sets percentage of used mana lost after dead.</i>	
OTS_Player::setLossExperience()	158
<i>Sets percentage of experience lost after dead.</i>	
OTS_Player::setLookType()	157
<i>Sets outfit.</i>	
OTS_Player::setLossSkills()	159
<i>Sets percentage of skills lost after dead.</i>	
OTS_Player::setMagLevel()	159
<i>Sets magic level.</i>	
OTS_Player::setManaSpent()	160
<i>Sets mana spent.</i>	
OTS_Player::setManaMax()	160
<i>Sets maximum mana.</i>	
OTS_Player::setMana()	159
<i>Sets current mana.</i>	
OTS_Player::setSex()	165
<i>Sets player gender.</i>	
OTS_Player::setSkill()	165
<i>Sets skill value.</i>	
OTS_Players_List::key()	170
<i>Current cursor position.</i>	
OTS_Players_List::deletePlayer()	170
<i>Deletes player.</i>	
OTS_Players_List::current()	170
<i>Returns current row.</i>	
OTS_Players_List::next()	171
<i>Moves to next row.</i>	
OTS_Players_List::rewind()	171
<i>Select players from database.</i>	
OTS_Players_List::valid()	172
<i>Checks if there are any rows left.</i>	
OTS_Players_List::setOffset()	172
<i>Sets OFFSET.</i>	
OTS_Players_List::setLimit()	171
<i>Sets LIMIT.</i>	

OTS Players List::count()	169
<i>Returns number of characters on list in current criterium.</i>	
OTS Players List	169
<i>List of players.</i>	
OTS Player::setSoul()	167
<i>Sets soul points.</i>	
OTS Player::setSlot()	166
<i>Sets slot content.</i>	
OTS Player::setSkillTries()	166
<i>Sets skill's tries for next level.</i>	
OTS Player::setTownId()	167
<i>Sets residence town's ID.</i>	
OTS Player::setVocation()	168
<i>Sets player proffesion.</i>	
OTS Player::unsetSave()	168
<i>Unsets save flag.</i>	
OTS Player::unsetRedSkull()	168
<i>Unsets red skull flag.</i>	
OTS Player::setLookLegs()	157
<i>Sets legs color.</i>	
OTS Player::setLookHead()	156
<i>Sets hair color.</i>	
OTS Player::getSoul()	146
<i>Soul points.</i>	
OTS Player::getSlot()	145
<i>Returns items tree from given slot.</i>	
OTS Player::getSkillTries()	145
<i>Returns player's skill's tries for next level.</i>	
OTS Player::getTownId()	146
<i>Residence town's ID.</i>	
OTS Player::getVocation()	147
<i>Player proffesion.</i>	
OTS Player::isSaveSet()	148
<i>Checks if save flag is set.</i>	
OTS Player::isLoading()	147
<i>Checks if object is loaded.</i>	
OTS Player::hasRedSkull()	147
<i>Checks if player has red skull.</i>	
OTS Player::getSkill()	144
<i>Returns player's skill.</i>	
OTS Player::getSex()	144
<i>Player gender.</i>	
OTS Player::getPosZ()	142
<i>Z map coordinate.</i>	
OTS Player::getPosY()	142
<i>Y map coordinate.</i>	
OTS Player::getPosX()	141
<i>X map coordinate.</i>	
OTS Player::getPremiumEnd()	142
<i>Player's Premium Account expiration timestamp.</i>	
OTS Player::getRank()	143
<i>Assigned guild rank.</i>	
OTS Player::getRedSkullTime()	144

<i>Red skulled time remained.</i>	
OTS_Player::getRankId()	143
<i>Guild rank ID.</i>	
OTS_Player::load()	148
<i>Loads player with given id.</i>	
OTS_Player::save()	148
<i>Saves player in database.</i>	
OTS_Player::setLastIP()	154
<i>Sets last login IP.</i>	
OTS_Player::setHealthMax()	154
<i>Sets maximum HP.</i>	
OTS_Player::setHealth()	153
<i>Sets current HP.</i>	
OTS_Player::setLastLogin()	154
<i>Sets last login timestamp.</i>	
OTS_Player::setLevel()	155
<i>Sets experience level.</i>	
OTS_Player::setLookFeet()	156
<i>Sets boots color.</i>	
OTS_Player::setLookBody()	156
<i>Sets body color.</i>	
OTS_Player::setLookAddons()	155
<i>Sets addons.</i>	
OTS_Player::setGuildNick()	153
<i>Sets guild nick.</i>	
OTS_Player::setGroup()	152
<i>Assigns character to group.</i>	
OTS_Player::setConditions()	150
<i>Sets conditions.</i>	
OTS_Player::setCap()	149
<i>Sets capacity.</i>	
OTS_Player::setAccount()	149
<i>Assigns character to account.</i>	
OTS_Player::setCustomField()	150
<i>Writes custom field.</i>	
OTS_Player::setDepot()	151
<i>Sets slot content.</i>	
OTS_Player::setExperience()	152
<i>Sets experience points.</i>	
OTS_Player::setDirection()	151
<i>Sets looking direction.</i>	
OTS_GuildRanks_List::current()	112
<i>Returns current row.</i>	
OTS_GuildRanks_List::count()	111
<i>Returns number of ranks on list in current criterium.</i>	
OTS_Container	64
<i>Container item representation.</i>	
OTS_Accounts_List::valid()	63
<i>Checks if there are any rows left.</i>	
OTS_Accounts_List::setOffset()	63
<i>Sets OFFSET.</i>	
OTS_Container::addItem()	64
<i>Adds item to container.</i>	

OTS Container::count()	64
<i>Number of items inside container.</i>	
OTS Container::next()	65
<i>Moves to next item.</i>	
OTS Container::key()	65
<i>Current cursor position.</i>	
OTS Container::current()	65
<i>Returns current item.</i>	
OTS Accounts List::setLimit()	62
<i>Sets LIMIT.</i>	
OTS Accounts List::rewind()	62
<i>Select accounts from database.</i>	
OTS Accounts List	60
<i>List of accounts.</i>	
OTS Account::unblock()	59
<i>Unblocks account.</i>	
OTS Account::setPassword()	59
<i>Sets account's password.</i>	
OTS Accounts List::count()	60
<i>Returns number of accounts on list in current criterium.</i>	
OTS Accounts List::current()	61
<i>Returns current row.</i>	
OTS Accounts List::next()	62
<i>Moves to next row.</i>	
OTS Accounts List::key()	61
<i>Current cursor position.</i>	
OTS Accounts List::deleteAccount()	61
<i>Deletes account.</i>	
OTS Container::removeItem()	66
<i>Removes given item from current container.</i>	
OTS Container::rewind()	66
<i>Resets internal items array pointer.</i>	
OTS DB ODBC::SQLquote()	73
<i>IOTS_DB method.</i>	
OTS DB ODBC::SQLquery()	72
<i>IOTS_DB method.</i>	
OTS DB ODBC::limit()	72
<i>LIMIT/OFFSET clause for queries.</i>	
OTS DB ODBC::tableName()	73
<i>Query-quoted table name.</i>	
OTS DB PostgreSQL	74
<i>PostgreSQL connection interface.</i>	
OTS DB PostgreSQL::SQLquery()	76
<i>IOTS_DB method.</i>	
OTS DB PostgreSQL::limit()	75
<i>LIMIT/OFFSET clause for queries.</i>	
OTS DB PostgreSQL::fieldName()	75
<i>Query-quoted field name.</i>	
OTS DB ODBC::fieldName()	71
<i>Query-quoted field name.</i>	
OTS DB ODBC	70
<i>ODBC connection interface.</i>	
OTS DB MySQL::fieldName()	68

Query-quoted field name.	67
OTS_DB_MySQL	67
MySQL connection interface.	
OTS_Container::valid()	66
Checks if there are any items left.	
OTS_DB_MySQL::limit()	68
LIMIT/OFFSET clause for queries.	
OTS_DB_MySQL::SQLquery()	69
IOTS_DB method.	
OTS_DB_MySQL::tableName()	70
Query-quoted table name.	
OTS_DB_MySQL::SQLquote()	69
IOTS_DB method.	
OTS_Account::setPACCDays()	58
Sets PACC days count.	
OTS_Account::setGroup()	58
Assigns account to group.	
OTS_GuildRanks_List.php	34
OTS_GuildRank.php	33
OTS_Guild.php	32
OTS_Guilds_List.php	35
OTS_InfoRespond.php	36
OTS_Players_List.php	39
OTS_Player.php	38
OTS_Item.php	37
OTS_Groups_List.php	31
OTS_Group.php	30
OTS_Container.php	25
OTS_Accounts_List.php	24
OTS_Account.php	23
OTS_DB_MySQL.php	26
OTS_DB_ODBC.php	27
OTS_DB_SQLite.php	29
OTS_DB_PostgreSQL.php	28
OTS_SQLite_Results.php	40
OTS_Account	49
OTServ account abstraction.	
OTS_Account::isBlocked()	55
Checks if account is blocked.	
OTS_Account::getPlayers()	55
List of characters on account.	
OTS_Account::getPassword()	55
Account's password.	
OTS_Account::isLoaded()	56
Checks if object is loaded.	
OTS_Account::load()	56
Loads account with given number.	
OTS_Account::setEMail()	57
Sets account's email.	
OTS_Account::setCustomField()	57
Writes custom field.	
OTS_Account::save()	56
Updates account in database.	

OTS Account::getPACCDays()	54
<i>PACC days.</i>	
OTS Account::getId()	54
<i>Account number.</i>	
OTS Account::createEx()	51
<i>Creates new account.</i>	
OTS Account::create()	50
<i>Creates new account.</i>	
OTS Account::block()	49
<i>Blocks account.</i>	
OTS Account::find()	52
<i>Loads account by it's e-mail address.</i>	
OTS Account::getCustomField()	52
<i>Reads custom field.</i>	
OTS Account::getGroup()	53
<i>Returns group of this account.</i>	
OTS Account::getEmail()	53
<i>E-mail address.</i>	
OTS DB PostgreSQL::SQLquote()	76
<i>IOTS_DB method.</i>	
OTS DB PostgreSQL::tableName()	76
<i>Query-quoted table name.</i>	
OTS Guild::load()	99
<i>Loads guild with given id.</i>	
OTS Guild::listRequests()	99
<i>Returns list of players that requested membership.</i>	
OTS Guild::listInvites()	99
<i>Returns list of invited players.</i>	
OTS Guild::request()	100
<i>Requests membership in guild for player player.</i>	
OTS Guild::save()	100
<i>Saves guild in database.</i>	
OTS Guild::setInvitesDriver()	102
<i>Assigns invites handler.</i>	
OTS Guild::setCustomField()	101
<i>Writes custom field.</i>	
OTS Guild::setCreationData()	101
<i>Sets guild creation data.</i>	
OTS Guild::isLoading()	98
<i>Checks if object is loaded.</i>	
OTS Guild::invite()	98
<i>Invites player to guild.</i>	
OTS Guild::getCreationData()	95
<i>Guild creation data.</i>	
OTS Guild::find()	95
<i>Loads guild by it's name.</i>	
OTS Guild::deleteRequest()	94
<i>Deletes request from player.</i>	
OTS Guild::getCustomField()	96
<i>Reads custom field.</i>	
OTS Guild::getGuildRanks()	96
<i>Reads all ranks that are in this guild.</i>	
OTS Guild::getOwner()	97

<i>Returns owning player of this player.</i>	
OTS_Guild::getName()	97
<i>Guild name.</i>	
OTS_Guild::getId()	97
<i>Guild ID.</i>	
OTS_Guild::setName()	102
<i>Sets players's name.</i>	
OTS_Guild::setOwner()	103
<i>Assigns guild to owner.</i>	
OTS_GuildRank::save()	108
<i>Saves rank in database.</i>	
OTS_GuildRank::load()	108
<i>Loads rank with given id.</i>	
OTS_GuildRank::isLoaded()	107
<i>Checks if object is loaded.</i>	
OTS_GuildRank::setCustomField()	108
<i>Writes custom field.</i>	
OTS_GuildRank::setGuild()	109
<i>Assigns rank to guild.</i>	
OTS_GuildRanks_List	111
<i>List of guild ranks.</i>	
OTS_GuildRank::setName()	110
<i>Sets rank's name.</i>	
OTS_GuildRank::setLevel()	110
<i>Sets rank's access level within guild.</i>	
OTS_GuildRank::getPlayers()	107
<i>Reads all players who has this rank set.</i>	
OTS_GuildRank::getName()	107
<i>Rank name.</i>	
OTS_GuildRank::find()	104
<i>Loads rank by it's name.</i>	
OTS_GuildRank	104
<i>OTServ guild rank abstraction.</i>	
OTS_Guild::setRequestsDriver()	103
<i>Assigns requests handler.</i>	
OTS_GuildRank::getCustomField()	105
<i>Reads custom field.</i>	
OTS_GuildRank::getGuild()	106
<i>Returns guild of this rank.</i>	
OTS_GuildRank::getLevel()	106
<i>Rank's access level.</i>	
OTS_GuildRank::getId()	106
<i>Rank ID.</i>	
OTS_Guild::deleteInvite()	94
<i>Deletes invitation for player to guild.</i>	
OTS_Guild::acceptRequest()	93
<i>Accepts player.</i>	
OTS_Group::getMaxDepotItems()	83
<i>Maximum count of items in depot.</i>	
OTS_Group::getId()	82
<i>Group ID.</i>	
OTS_Group::getFlags()	82
<i>Rights flags.</i>	

OTS_Group::getMaxVIPList()	83
<i>Maximum count of players in VIP list.</i>	
OTS_Group::getName()	83
<i>Group name.</i>	
OTS_Group::load()	84
<i>Loads group with given id.</i>	
OTS_Group::isLoaded()	84
<i>Checks if object is loaded.</i>	
OTS_Group::getPlayers()	84
<i>List of characters in given group.</i>	
OTS_Group::getCustomField()	81
<i>Reads custom field.</i>	
OTS_Group::getAccess()	81
<i>Access level.</i>	
OTS_DB_SQLite::limit()	78
<i>LIMIT/OFFSET clause for queries.</i>	
OTS_DB_SQLite::fieldName()	78
<i>Query-quoted field name.</i>	
OTS_DB_SQLite	77
<i>SQLite connection interface.</i>	
OTS_DB_SQLite::SQLquery()	79
<i>IOTS_DB method.</i>	
OTS_DB_SQLite::SQLquote()	79
<i>IOTS_DB method.</i>	
OTS_Group	80
<i>OTServ user group abstraction.</i>	
OTS_DB_SQLite::tableName()	80
<i>Query-quoted table name.</i>	
OTS_Group::save()	85
<i>Saves account in database.</i>	
OTS_Group::setAccess()	85
<i>Sets access level.</i>	
OTS_Groups_List::rewind()	90
<i>Select groups from database.</i>	
OTS_Groups_List::next()	90
<i>Moves to next row.</i>	
OTS_Groups_List::key()	90
<i>Current cursor position.</i>	
OTS_Groups_List::setLimit()	91
<i>Sets LIMIT.</i>	
OTS_Groups_List::setOffset()	91
<i>Sets OFFSET.</i>	
OTS_Guild::acceptInvite()	93
<i>Finalise invitation.</i>	
OTS_Guild	92
<i>OTServ guild abstraction.</i>	
OTS_Groups_List::valid()	92
<i>Checks if there are any rows left.</i>	
OTS_Groups_List::deleteGroup()	89
<i>Deletes group.</i>	
OTS_Groups_List::current()	89
<i>Returns current row.</i>	
OTS_Group::setMaxDepotItems()	87

<i>Sets maximum count of items in depot.</i>	
OTS_Group::setFlags()	86
<i>Sets rights flags.</i>	
OTS_Group::setCustomField()	86
<i>Writes custom field.</i>	
OTS_Group::setMaxVIPList()	87
<i>Sets maximum count of players in VIP list.</i>	
OTS_Group::setName()	88
<i>Sets group's name.</i>	
OTS_Groups_List::count()	89
<i>Returns number of groups on list in current criterium.</i>	
OTS_Groups_List	88
<i>List of groups.</i>	
OTS.php	22
<i>This file contains main toolkit class.</i>	

P

POT::SLOT_LEGS	181
<i>Legs slot.</i>	
POT::SLOT_NECKLACE	181
<i>Necklace slot.</i>	
POT::SLOT_RIGHT	182
<i>Right hand slot.</i>	
POT::SLOT_RING	182
<i>Ring slot.</i>	
POT::SLOT_LEFT	181
<i>Left hand slot.</i>	
POT::SLOT_HEAD	180
<i>Head slot.</i>	
POT::SLOT_ARMOR	179
<i>Armor slot.</i>	
POT::SLOT_BACKPACK	179
<i>Backpack slot.</i>	
POT::SLOT_FEET	180
<i>Boots slot.</i>	
POT::VOCATION_DRUID	182
<i>Druid.</i>	
POT::VOCATION_KNIGHT	183
<i>Knight.</i>	
POT::getInstance()	186
<i>Singleton.</i>	
POT::loadClass()	186
<i>Loads POT class file.</i>	
POT::serverStatus()	186
<i>Queries server status.</i>	
POT::setPOTPath()	188
<i>Set POT directory.</i>	
POT::createObject()	185
<i>Creates OTServ DAO class instance.</i>	
POT::connect()	184
<i>Connects to database.</i>	

POT::VOCATION_NONE	183
<i>None vocation.</i>	
POT::VOCATION_PALADIN	183
<i>Paladin.</i>	
POT::VOCATION_SORCERER	184
<i>Sorcerer.</i>	
POT::SLOT_AMMO	179
<i>Ammunition slot.</i>	
POT::SKILL_SWORD	178
<i>Sword fighting.</i>	
POT::DB_PGSQL	174
<i>PostgreSQL driver.</i>	
POT::DB_SQLITE	174
<i>SQLite driver.</i>	
POT::DIRECTION_EAST	174
<i>East.</i>	
POT::DIRECTION_NORTH	175
<i>North.</i>	
POT::DB_ODBC	173
<i>ODBC driver.</i>	
POT::DB_MYSQL	173
<i>MySQL driver.</i>	
PHP 5.0	3
POT class preview	5
POT	173
<i>Main POT class.</i>	
POT::DIRECTION_SOUTH	175
<i>South.</i>	
POT::DIRECTION_WEST	175
<i>West.</i>	
POT::SKILL_FISHING	177
<i>Fishing.</i>	
POT::SKILL_FIST	178
<i>Fist fighting.</i>	
POT::SKILL_SHIELDING	178
<i>Shielding.</i>	
POT::SKILL_DISTANCE	177
<i>Distance fighting.</i>	
POT::SKILL_CLUB	176
<i>Club fighting.</i>	
POT::SEX_FEMALE	175
<i>Female gender.</i>	
POT::SEX_MALE	176
<i>Male gender.</i>	
POT::SKILL_AXE	176
<i>Axe fighting.</i>	
POT	1

Q

Quick start	6
-----------------------------	---

R

README	196
----------------------------------	-----

S

Server online status	14
--	----