

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 NotLoaded.php	17
IOTS DAO.php	18
IOTS DB.php	19
OTS.php	20
OTS Account.php	21
OTS Accounts List.php	22
OTS Container.php	23
OTS DB MySQL.php	24
OTS DB SQLite.php	25
OTS Group.php	26
OTS Groups List.php	27
OTS InfoRespond.php	28
OTS Item.php	29
OTS Player.php	30
OTS Players List.php	31
OTS SQLite Results.php	32
Package POT Classes	33
Class E OTS NotLoaded	33
Class IOTS DAO	33
Constructor construct	34
Class IOTS DB	34
Constructor construct	34
Method fieldName	35
Method lastInsertId	35
Method limit	36
Method SQLquery	36
Method SQLquote	36
Method tableName	37
Class OTS Account	37
Constructor construct	38
Method block	38
Method create	38
example: account.php	38
Method find	40

Method getCustomField	40
Method getEmail	41
Method getGroup	41
Method getId	41
Method getPACCDays	42
Method getPassword	42
Method getPlayers	42
Method isBlocked	43
Method isLoaded	43
Method load	43
Method save	44
Method setCustomField	44
Method setEmail	45
Method setGroup	45
Method setPACCDays	46
Method setPassword	46
Method unblock	47
Class OTS_Accounts_List	47
Constructor construct	47
Method count	48
Method current	48
Method deleteAccount	48
Method key	49
Method next	49
Method rewind	50
Method setLimit	50
Method setOffset	50
Method valid	51
Class OTS_Container	51
Method addItem	51
Method count	52
Method current	52
Method key	53
Method next	53
Method removeItem	53
Method rewind	54
Method valid	54
Class OTS_DB_MySQL	54
Constructor construct	55
Method fieldName	55
Method limit	56
Method SQLquery	56
Method SQLquote	57
Method tableName	57
Class OTS_DB_SQLite	58
Constructor construct	58
Method fieldName	59
Method limit	59
Method regexp	59

Method SQLQuery	60
Method SQLQuote	60
Method tableName	61
Class OTS_Group	61
Constructor construct	62
Method getAccess	62
Method getCustomField	62
Method getFlags	63
Method getId	63
Method getMaxDepotItems	64
Method getMaxVIPList	64
Method getName	64
Method getPlayers	65
Method isLoaded	65
Method load	65
Method save	66
Method setAccess	66
Method setCustomField	67
Method setFlags	67
Method setMaxDepotItems	68
Method setMaxVIPList	68
Method setName	69
Class OTS_Groups_List	69
Constructor construct	69
Method count	70
Method current	70
Method deleteGroup	70
Method key	71
Method next	71
Method rewind	71
Method setLimit	72
Method setOffset	72
Method valid	73
Class OTS_InfoRespond	73
Method getClientVersion	73
Method getEmail	74
Method getIP	74
Method getLocation	74
Method getMapAuthor	75
Method getMapHeight	75
Method getMapName	75
Method getMapWidth	76
Method getMaxPlayers	76
Method getMonstersCount	76
Method getMOTD	77
Method getName	77
Method getOnlinePlayers	77
Method getOwner	78
Method getPlayersPeak	78

Method getPort	78
Method getServer	79
Method getServerVersion	79
Method getTSPQVersion	79
Method getUptime	80
Method getURL	80
Class OTS_Item	80
Constructor construct	81
Method count	81
Method getAttributes	81
Method getCount	82
Method getId	82
Method setAttributes	82
Method setCount	83
Class OTS_Player	83
Constructor construct	84
Method find	84
Method getAccount	85
Method getCap	85
Method getConditions	85
Method getCustomField	86
Method getDepot	86
Method getDirection	87
Method getExperience	87
Method getGroup	88
Method getGuildNick	88
Method getHealth	88
Method getHealthMax	89
Method getId	89
Method getLastIP	89
Method getLastLogin	90
Method getLevel	90
Method getLookAddons	90
Method getLookBody	91
Method getLookFeet	91
Method getLookHead	92
Method getLookLegs	92
Method getLookType	92
Method getLossExperience	93
Method getLossMana	93
Method getLossSkills	93
Method getMagLevel	94
Method getMana	94
Method getManaMax	94
Method getManaSpent	95
Method getName	95
Method getPosX	95
Method getPosY	96
Method getPosZ	96

Method getPremiumEnd	97
Method getRankId	97
Method getRedSkullTime	97
Method getSex	98
Method getSkill	98
Method getSkillTries	99
Method getSlot	99
Method getSoul	100
Method getTownId	100
Method getVocation	100
Method hasRedSkull	101
Method isLoading	101
Method isSaveSet	101
Method load	102
Method save	102
Method setAccount	102
Method setCap	103
Method setConditions	103
Method setCustomField	104
Method setDepot	104
Method setDirection	105
Method setExperience	106
Method setGroup	106
Method setGuildNick	106
Method setHealth	107
Method setHealthMax	107
Method setLastIP	108
Method setLastLogin	108
Method setLevel	108
Method setLookAddons	109
Method setLookBody	109
Method setLookFeet	110
Method setLookHead	110
Method setLookLegs	111
Method setLookType	111
Method setLossExperience	111
Method setLossMana	112
Method setLossSkills	112
Method setMagLevel	113
Method setMana	113
Method setManaMax	113
Method setManaSpent	114
Method setName	114
Method setPosX	115
Method setPosY	115
Method setPosZ	116
Method setPremiumEnd	116
Method setRankId	116
Method setRedSkull	117

Method setRedSkullTime	117
Method setSave	118
Method setSex	118
Method setSkill	118
Method setSkillTries	119
Method setSlot	119
Method setSoul	120
Method setTownId	120
Method setVocation	121
Method unsetRedSkull	121
Method unsetSave	121
Class OTS Players List	122
Constructor construct	122
Method count	123
Method current	123
Method deletePlayer	123
Method key	124
Method next	124
Method rewind	124
Method setLimit	125
Method setOffset	125
Method valid	125
Class POT	126
Class Constant DB_MYSQL	126
Class Constant DB_SQLITE	126
Class Constant DIRECTION_EAST	127
Class Constant DIRECTION_NORTH	127
Class Constant DIRECTION_SOUTH	127
Class Constant DIRECTION_WEST	128
Class Constant SEX_FEMALE	128
Class Constant SEX_MALE	128
Class Constant SKILL_AXE	129
Class Constant SKILL_CLUB	129
Class Constant SKILL_DISTANCE	129
Class Constant SKILL_FISHING	130
Class Constant SKILL_FIST	130
Class Constant SKILL_SHIELDING	130
Class Constant SKILL_SWORD	131
Class Constant SLOT_AMMO	131
Class Constant SLOT_ARMOR	132
Class Constant SLOT_BACKPACK	132
Class Constant SLOT_FEET	132
Class Constant SLOT_HEAD	133
Class Constant SLOT_LEFT	133
Class Constant SLOT_LEGS	133
Class Constant SLOT_NECKLACE	134
Class Constant SLOT_RIGHT	134
Class Constant SLOT_RING	135
Class Constant VOCATION_DRUID	135

Class Constant VOCATION_KNIGHT	135
Class Constant VOCATION_NONE	136
Class Constant VOCATION_PALADIN	136
Class Constant VOCATION_SORCERER	136
Method connect	136
example: connect.php	136
Method createObject	138
Method getInstance	138
Method loadClass	138
Method serverStatus	139
example: example	139
Method setPOTPath	140
example: fakeroot.php	140
compat.php	142
Appendices	143
Appendix A - Class Trees	144
POT	144
Appendix B - README/CHANGELOG/INSTALL	147
CHANGELOG	148
README	148
INSTALL	149
NEWS	150

POT

This is documentenation 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_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](#)

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
- **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_SQLite.php

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

OTS_Group.php

- **Package** POT
- **Author** Wrzasq < wrzasq@gmail.com>
- **Version** 0.0.1
- **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_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_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 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 295]

Blocks account.

Blocks account.

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

int function OTS_Account::create([\$min = 1], [\$max = 99999999]) [line 62]

account.php

```

1  <?php
2
3  /**
4   * @ignore
5   * @package examples
6   * @author Wrzasq <wrzasq@gmail.com>
7   * @copyright 2007 (C) by Wrzasq
8   * @license http://www.gnu.org/licenses/lgpl-3.0.txt GNU Lesser General Public License, Version 3
9   */
10
11 // to not repeat all that stuff
12 include('quickstart.php');
13
14 // creates new OTS_Account object
15 $account = $ots->createObject('Account');
16
17 // generates new account number
18 $number = $account->create();
19
20 /*
21 to generate number from 111111 to 999999 use:
22 $number = $account->create(111111, 999999);
23 */
24
25 // sets account info
26 $account->setPassword('secret'); // $account->setPassword( md5('secret') );
27 $account->setEMail('foo@example.com');
28 $account->unblock(); // remember to unblock!
29 $account->setPACCDays(0);
30 $account->save();
31
32 // give user his number
33 echo 'Your account number is: ', $number;
34
35 ?>

```

Function Parameters:

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

Creates new account.

Creates new account.

Create new account in given range (1 - 9999999 by default).

Remember! This method sets blocked flag to true after account creation!

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

void function OTS_Account::find(\$email) [*line 127*]

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 342*]

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 247*]

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 191*]

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 173*]

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 308*]

PACC days.

PACC days.

- **Version** 0.0.3
- **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 220*]

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 391*]

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 274*]

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::isLoading() [*line 144*]

Checks if object is loaded.

Checks if object is loaded.

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

void function OTS_Account::load(\$id) [*line 114*]

Function Parameters:

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

Loads account with given number.

Loads account with given number.

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

void function OTS_Account::save() [line 155]

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 368]

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 262]

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 208]

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 324]

Function Parameters:

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

Sets PACC days count.

Sets PACC days count.

- **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 235]

Function Parameters:

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

Sets account's password.

Sets account's password.

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

void function OTS_Account::unlock() [line 287]

Unblocks account.

Unblocks 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 its 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 against other values.

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

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

Function Parameters:

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

Query-quoted table name.

Query-quoted table name.

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

Class OTS_DB_SQLite

[line 19]

SQLite connection interface.

SQLite connection interface.

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

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

Function Parameters:

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

Creates database connection.

Creates database connection.

Connects to SQLite database on given arguments.

List of parameters for this drivers:

- *database* - database name.

- **Version** 0.0.1
- **See** [POT::connect\(\)](#)
- **Access** public

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

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

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

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

Function Parameters:

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

LIMIT/OFFSET clause for queries.

LIMIT/OFFSET clause for queries.

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

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

Function Parameters:

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

- *string* **\$content** String to test.

REGEXP operator for SQLite

REGEXP operator for SQLite

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

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

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

IOTS_DB method.

IOTS_DB method.
 Overwrites PDO method.

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

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

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

IOTS_DB method.

IOTS_DB method.
 Overwrites PDO method - we won't use quoting 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|bool 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_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 1206]

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 1471]

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_OTTS_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_OTTS_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_OTTS_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 1121*]

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 1147*]

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 1173*]

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

int function OTS_Player::getRankId() [*line 1069*]

Guild rank ID.

Guild rank ID.

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

int function OTS_Player::getRedSkullTime() [*line 982*]

Red skulled time remained.

Red skulled time remained.

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

- **Throws** E_OTS_NotLoaded If player is not loaded.
- **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 1257*]

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 1289*]

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 1342*]

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 1095*]

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 account in database.

Saves account 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 1232]

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 player is not loaded.
- **Since** 0.0.3
- **Access** public

void function OTS_Player::setDepot(\$depot, [\$item = null], [\$pid = 0], [\$depot_id = 0]) [line 1526]

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 1136]

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 1162]
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 1188]
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::setRankId(\$rank_id) [line 1084]

Function Parameters:

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

Sets guild rank ID.

Sets guild rank ID.

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

void function OTS_Player::setRedSkull() [*line 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 1275]

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 1307]

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 1396]

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 1110]

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

POT::DB_MYSQL

= 1 *[line 28]*

MySQL driver.
MySQL driver.

- **Version** 0.0.1

POT::DB_SQLITE

= 2 *[line 32]*

SQLite driver.
SQLite driver.

- **Version 0.0.1**

POT::DIRECTION_EAST

= 1 [*line 71*]

East.

East.

- **Version 0.0.1**

POT::DIRECTION_NORTH

= 0 [*line 67*]

North.

North.

- **Version 0.0.1**

POT::DIRECTION_SOUTH

= 2 [*line 75*]

South.

South.

- **Version 0.0.1**

POT::DIRECTION_WEST

= 3 *[line 79]*

West.

West.

- **Version 0.0.1**

POT::SEX_FEMALE

= 0 *[line 37]*

Female gender.

Female gender.

- **Version 0.0.1**

POT::SEX_MALE

= 1 *[line 41]*

Male gender.

Male gender.

- **Version 0.0.1**

POT::SKILL_AXE

= 3 *[line 108]*

Axe fighting.

Axe fighting.

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

POT::SKILL_CLUB

= 1 *[line 94]*

Club fighting.

Club fighting.

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

POT::SKILL_DISTANCE

= 4 *[line 115]*

Distance fighting.

Distance fighting.

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

POT::SKILL_FISHING

= 6 [*line 129*]

Fishing.

Fishing.

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

POT::SKILL_FIST

= 0 [*line 87*]

Fist fighting.

Fist fighting.

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

POT::SKILL_SHIELDING

= 5 [*line 122*]

Shielding.

Shielding.

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

POT::SKILL_SWORD

= 2 [*line 101*]

Sword fighting.
Sword fighting.

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

POT::SLOT_AMMO

= 10 [*line 200*]

Ammunition slot.
Ammunition slot.

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

POT::SLOT_ARMOR

= 4 *[line 158]*

Armor slot.

Armor slot.

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

POT::SLOT_BACKPACK

= 3 *[line 151]*

Backpack slot.

Backpack slot.

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

POT::SLOT_FEET

= 8 *[line 186]*

Boots slot.

Boots slot.

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

POT::SLOT_HEAD

= 1 [*line 137*]

Head slot.

Head slot.

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

POT::SLOT_LEFT

= 6 [*line 172*]

Left hand slot.

Left hand slot.

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

POT::SLOT_LEGS

= 7 [*line 179*]

Legs slot.

Legs slot.

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

POT::SLOT_NECKLACE

= 2 [*line 144*]

Necklace slot.
Necklace slot.

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

POT::SLOT_RIGHT

= 5 [*line 165*]

Right hand slot.
Right hand slot.

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

POT::SLOT_RING

= 9 [*line 193*]

Ring slot.

Ring slot.

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

POT::VOCATION_DRUID

= 2 [*line 54*]

Druid.

Druid.

- **Version** 0.0.1

POT::VOCATION_KNIGHT

= 4 [*line 62*]

Knight.

Knight.

- **Version** 0.0.1

POT::VOCATION_NONE

= 0 *[line 46]*

None vocation.

None vocation.

- **Version 0.0.1**

POT::VOCATION_PALADIN

= 3 *[line 58]*

Paladin.

Paladin.

- **Version 0.0.1**

POT::VOCATION_SORCERER

= 1 *[line 50]*

Sorcerer.

Sorcerer.

- **Version 0.0.1**

void function POT::connect(\$driver, \$params) *[line 319]*

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 than one OTServ installed on one database.

- **Version** 0.0.1
- **Throws** Exception When driver is not supported.

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

IOTS_DAO function POT::createObject(\$class) [*line 362*]

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 207*]

Singleton.

Singleton.

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

void function POT::loadClass(\$class) [*line 279*]

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 380]

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 238]

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_NotLoaded

- Exception
 - [E_OTS_NotLoaded](#)

IOTS_DAO

- [IOTS_DAO](#)

IOTS_DB

- [IOTS_DB](#)

OTS_Account

- [OTS_Account](#)

OTS_Accounts_List

- [OTS_Accounts_List](#)

OTS_DB_MySQL

- PDO
 - [OTS_DB_MySQL](#)

OTS_DB_SQLite

- PDO
 - [OTS_DB_SQLite](#)

OTS_Group

- [OTS_Group](#)

OTS_Groups_List

- [OTS_Groups_List](#)

OTS_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 account group support. <wrzasq>
- * Added support for depot_id field (it is reserved in OTServ for further use). <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_Player::__construct\(\)](#) 84
 Sets database connection handler.
[constructor OTS_Item::__construct\(\)](#) 81
 Creates item of given ID.
[constructor OTS_Players_List::__construct\(\)](#) 122
 Sets database connection handler.
[compat.php](#) 142
 POT compatibility assurance package.
[CHANGELOG](#) 148
[constructor OTS_Groups_List::__construct\(\)](#) 69
 Sets database connection handler.
[constructor OTS_Group::__construct\(\)](#) 62
 Sets database connection handler.
[constructor OTS_Account::__construct\(\)](#) 38
 Sets database connection handler.
[constructor IOTS_DB::__construct\(\)](#) 34
 Connection parameters.
[constructor OTS_Accounts_List::__construct\(\)](#) 47
 Sets database connection handler.
[constructor OTS_DB_MySQL::__construct\(\)](#) 55
 Creates database connection.
[constructor OTS_DB_SQLite::__construct\(\)](#) 58
 Creates database connection.
[constructor IOTS_DAO::__construct\(\)](#) 34
 DAO objects must be initialized with a database.

D

[DAO objects](#) 9

E

[E_OTS_NotLoaded](#) 33
 Occurs when code attempts to access property of not loaded object.
[E_OTS_NotLoaded.php](#) 17

I	
IOTS_DB::SQLquery()	36
<i>Evaluates query.</i>	
IOTS_DB::SQLquote()	36
<i>Query-quoted string value.</i>	
IOTS_DB::tableName()	37
<i>Query-quoted table name.</i>	
INSTALL	149
IOTS_DB::limit()	36
<i>LIMIT/OFFSET clause for queries.</i>	
IOTS_DB::lastInsertId()	35
<i>ID of last created record.</i>	
IOTS_DB.php	19
IOTS_DAO	33
<i>OTserv database object.</i>	
IOTS_DB	34
<i>OTServ database handler interface.</i>	
IOTS_DB::fieldName()	35
<i>Query-quoted field name.</i>	
IOTS_DAO.php	18

N	
NEWS	150

O	
OTS_Player::getManaMax()	94
<i>Maximum mana.</i>	
OTS_Player::getMana()	94
<i>Current mana.</i>	
OTS_Player::getManaSpent()	95
<i>Mana spent.</i>	
OTS_Player::getName()	95
<i>Player name.</i>	
OTS_Player::getPosY()	96
<i>Y map coordinate.</i>	
OTS_Player::getPosX()	95
<i>X map coordinate.</i>	
OTS_Player::getMagLevel()	94
<i>Magic level.</i>	
OTS_Player::getLossSkills()	93
<i>Percentage of skills lost after dead.</i>	
OTS_Player::getLookLegs()	92
<i>Legs color.</i>	
OTS_Player::getLookHead()	92
<i>Hair color.</i>	
OTS_Player::getLookType()	92
<i>Outfit.</i>	
OTS_Player::getLossExperience()	93

<i>Percentage of experience lost after dead.</i>	
OTS_Player::getLossMana()	93
<i>Percentage of used mana lost after dead.</i>	
OTS_Player::getPosZ()	96
<i>Z map coordinate.</i>	
OTS_Player::getPremiumEnd()	97
<i>Player's Premium Account expiration timestamp.</i>	
OTS_Player::getVocation()	100
<i>Player proffesion.</i>	
OTS_Player::getTownId()	100
<i>Residence town's ID.</i>	
OTS_Player::hasRedSkull()	101
<i>Checks if player has red skull.</i>	
OTS_Player::isLoading()	101
<i>Checks if object is loaded.</i>	
OTS_Player::isSaveSet()	101
<i>Checks if save flag is set.</i>	
OTS_Player::getSoul()	100
<i>Soul points.</i>	
OTS_Player::getSlot()	99
<i>Returns items tree from given slot.</i>	
OTS_Player::getRedSkullTime()	97
<i>Red skulled time remained.</i>	
OTS_Player::getRankId()	97
<i>Guild rank ID.</i>	
OTS_Player::getSex()	98
<i>Player gender.</i>	
OTS_Player::getSkill()	98
<i>Returns player's skill.</i>	
OTS_Player::getSkillTries()	99
<i>Returns player's skill's tries for next level.</i>	
OTS_Player::getLookFeet()	91
<i>Boots color.</i>	
OTS_Player::getLookBody()	91
<i>Body color.</i>	
OTS_Player	83
<i>OTServ character abstraction.</i>	
OTS_Item::setCount()	83
<i>Sets count of item.</i>	
OTS_Player::find()	84
<i>Loads player by it's name.</i>	
OTS_Player::getAccount()	85
<i>Returns account of this player.</i>	
OTS_Player::getCap()	85
<i>Capacity.</i>	
OTS_Item::setAttributes()	82
<i>Sets item attributes.</i>	
OTS_Item::getId()	82
<i>Returns item type.</i>	
OTS_Item	80
<i>Single item representation.</i>	
OTS_InfoRespond::getURL()	80
<i>Returns server website.</i>	

<u>OTS_Item::count()</u>	81
<i>Count value for current item.</i>	
<u>OTS_Item::getAttributes()</u>	81
<i>Returns item custom attributes.</i>	
<u>OTS_Item::getCount()</u>	82
<i>Returns count of item.</i>	
<u>OTS_Player::getConditions()</u>	85
<i>Conditions.</i>	
<u>OTS_Player::getCustomField()</u>	86
<i>Reads custom field.</i>	
<u>OTS_Player::getLastIP()</u>	89
<i>Last login IP.</i>	
<u>OTS_Player::getId()</u>	89
<i>Player ID.</i>	
<u>OTS_Player::getLastLogin()</u>	90
<i>Last login timestamp.</i>	
<u>OTS_Player::getLevel()</u>	90
<i>Experience level.</i>	
<u>OTS_Player::getLookAddons()</u>	90
<i>Addons.</i>	
<u>OTS_Player::getHealthMax()</u>	89
<i>Maximum HP.</i>	
<u>OTS_Player::getHealth()</u>	88
<i>Current HP.</i>	
<u>OTS_Player::getDirection()</u>	87
<i>Looking direction.</i>	
<u>OTS_Player::getDepot()</u>	86
<i>Returns items tree from given depot.</i>	
<u>OTS_Player::getExperience()</u>	87
<i>Experience points.</i>	
<u>OTS_Player::getGroup()</u>	88
<i>Returns group of this player.</i>	
<u>OTS_Player::getGuildNick()</u>	88
<i>Guild nick.</i>	
<u>OTS_Player::load()</u>	102
<i>Loads player with given id.</i>	
<u>OTS_Player::save()</u>	102
<i>Saves account in database.</i>	
<u>OTS_Player::setSex()</u>	118
<i>Sets player gender.</i>	
<u>OTS_Player::setSave()</u>	118
<i>Sets save flag.</i>	
<u>OTS_Player::setSkill()</u>	118
<i>Sets skill value.</i>	
<u>OTS_Player::setSkillTries()</u>	119
<i>Sets skill's tries for next level.</i>	
<u>OTS_Player::setSoul()</u>	120
<i>Sets soul points.</i>	
<u>OTS_Player::setSlot()</u>	119
<i>Sets slot content.</i>	
<u>OTS_Player::setRedSkullTime()</u>	117
<i>Sets red skulled time remained.</i>	
<u>OTS_Player::setRedSkull()</u>	117

<i>Sets red skull flag.</i>	
OTS_Player::setPosY()	115
<i>Sets Y map coordinate.</i>	
OTS_Player::setPosX()	115
<i>Sets X map coordinate.</i>	
OTS_Player::setPosZ()	116
<i>Sets Z map coordinate.</i>	
OTS_Player::setPremiumEnd()	116
<i>Sets player's Premium Account expiration timestamp.</i>	
OTS_Player::setRankId()	116
<i>Sets guild rank ID.</i>	
OTS_Player::setTownId()	120
<i>Sets residence town's ID.</i>	
OTS_Player::setVocation()	121
<i>Sets player proffesion.</i>	
OTS_Players_List::rewind()	124
<i>Select players from database.</i>	
OTS_Players_List::next()	124
<i>Moves to next row.</i>	
OTS_Players_List::setLimit()	125
<i>Sets LIMIT.</i>	
OTS_Players_List::setOffset()	125
<i>Sets OFFSET.</i>	
OTS_Players_List::valid()	125
<i>Checks if there are any rows left.</i>	
OTS_Players_List::key()	124
<i>Current cursor position.</i>	
OTS_Players_List::deletePlayer()	123
<i>Deletes player.</i>	
OTS_Player::unsetSave()	121
<i>Unsets save flag.</i>	
OTS_Player::unsetRedSkull()	121
<i>Unsets red skull flag.</i>	
OTS_Players_List	122
<i>List of players.</i>	
OTS_Players_List::count()	123
<i>Returns number of characters on list in current criterium.</i>	
OTS_Players_List::current()	123
<i>Returns current row.</i>	
OTS_Player::setName()	114
<i>Sets players's name.</i>	
OTS_Player::setManaSpent()	114
<i>Sets mana spent.</i>	
OTS_Player::setGuildNick()	106
<i>Sets guild nick.</i>	
OTS_Player::setGroup()	106
<i>Assigns character to group.</i>	
OTS_Player::setHealth()	107
<i>Sets current HP.</i>	
OTS_Player::setHealthMax()	107
<i>Sets maximum HP.</i>	
OTS_Player::setLastIP()	108
<i>Sets last login IP.</i>	

OTS_Player::setExperience()	106
<i>Sets experience points.</i>	
OTS_Player::setDirection()	105
<i>Sets looking direction.</i>	
OTS_Player::setCap()	103
<i>Sets capacity.</i>	
OTS_Player::setAccount()	102
<i>Assigns character to account.</i>	
OTS_Player::setConditions()	103
<i>Sets conditions.</i>	
OTS_Player::setCustomField()	104
<i>Writes custom field.</i>	
OTS_Player::setDepot()	104
<i>Sets slot content.</i>	
OTS_Player::setLastLogin()	108
<i>Sets last login timestamp.</i>	
OTS_Player::setLevel()	108
<i>Sets experience level.</i>	
OTS_Player::setLossSkills()	112
<i>Sets percentage of skills lost after dead.</i>	
OTS_Player::setLossMana()	112
<i>Sets percentage of used mana lost after dead.</i>	
OTS_Player::setMagLevel()	113
<i>Sets magic level.</i>	
OTS_Player::setMana()	113
<i>Sets current mana.</i>	
OTS_Player::setManaMax()	113
<i>Sets maximum mana.</i>	
OTS_Player::setLossExperience()	111
<i>Sets percentage of experience lost after dead.</i>	
OTS_Player::setLookType()	111
<i>Sets outfit.</i>	
OTS_Player::setLookBody()	109
<i>Sets body color.</i>	
OTS_Player::setLookAddons()	109
<i>Sets addons.</i>	
OTS_Player::setLookFeet()	110
<i>Sets boots color.</i>	
OTS_Player::setLookHead()	110
<i>Sets hair color.</i>	
OTS_Player::setLookLegs()	111
<i>Sets legs color.</i>	
OTS_InfoRespond::getUptime()	80
<i>Returns server uptime.</i>	
OTS_InfoRespond::getTSPQVersion()	79
<i>Returns version of root element.</i>	
OTS_Accounts_List::deleteAccount()	48
<i>Deletes account.</i>	
OTS_Accounts_List::current()	48
<i>Returns current row.</i>	
OTS_Accounts_List::key()	49
<i>Current cursor position.</i>	
OTS_Accounts_List::next()	49

<i>Moves to next row.</i>	
OTS Accounts List::setLimit()	50
<i>Sets LIMIT.</i>	
OTS Accounts List::rewind()	50
<i>Select accounts from database.</i>	
OTS Accounts List::count()	48
<i>Returns number of accounts on list in current criterium.</i>	
OTS Accounts List	47
<i>List of accounts.</i>	
OTS Account::setGroup()	45
<i>Assigns account to group.</i>	
OTS Account::setEMail()	45
<i>Sets account's email.</i>	
OTS Account::setPACCDays()	46
<i>Sets PACC days count.</i>	
OTS Account::setPassword()	46
<i>Sets account's password.</i>	
OTS Account::unblock()	47
<i>Unblocks account.</i>	
OTS Accounts List::setOffset()	50
<i>Sets OFFSET.</i>	
OTS Accounts List::valid()	51
<i>Checks if there are any rows left.</i>	
OTS Container::valid()	54
<i>Checks if there are any items left.</i>	
OTS Container::rewind()	54
<i>Resets internal items array pointer.</i>	
OTS DB MySQL	54
<i>MySQL connection interface.</i>	
OTS DB MySQL::fieldName()	55
<i>Query-quoted field name.</i>	
OTS DB MySQL::limit()	56
<i>LIMIT/OFFSET clause for queries.</i>	
OTS Container::removeItem()	53
<i>Removes given item from current container.</i>	
OTS Container::next()	53
<i>Moves to next item.</i>	
OTS Container::addItem()	51
<i>Adds item to container.</i>	
OTS Container	51
<i>Container item representation.</i>	
OTS Container::count()	52
<i>Number of items inside container.</i>	
OTS Container::current()	52
<i>Returns current item.</i>	
OTS Container::key()	53
<i>Current cursor position.</i>	
OTS Account::setCustomField()	44
<i>Writes custom field.</i>	
OTS Account::save()	44
<i>Updates account in database.</i>	
OTS Item.php	29
OTS InfoRespond.php	28

OTS_Player.php	30
OTS_Players_List.php	31
OTS_SQLite_Results.php	32
OTS_Groups_List.php	27
OTS_Group.php	26
OTS_Accounts_List.php	22
OTS_Account.php	21
OTS_Container.php	23
OTS_DB_MySQL.php	24
OTS_DB_SQLite.php	25
OTS_Account	37
<i>OTServ account abstraction.</i>	
OTS_Account::block()	38
<i>Blocks account.</i>	
OTS_Account::getPlayers()	42
<i>List of characters on account.</i>	
OTS_Account::getPassword()	42
<i>Account's password.</i>	
OTS_Account::isBlocked()	43
<i>Checks if account is blocked.</i>	
OTS_Account::isLoading()	43
<i>Checks if object is loaded.</i>	
OTS_Account::load()	43
<i>Loads account with given number.</i>	
OTS_Account::getPACCDays()	42
<i>PACC days.</i>	
OTS_Account::getId()	41
<i>Account number.</i>	
OTS_Account::find()	40
<i>Loads account by it's e-mail address.</i>	
OTS_Account::create()	38
<i>Creates new account.</i>	
OTS_Account::getCustomField()	40
<i>Reads custom field.</i>	
OTS_Account::getEmail()	41
<i>E-mail address.</i>	
OTS_Account::getGroup()	41
<i>Returns group of this account.</i>	
OTS_DB_MySQL::SQLquery()	56
<i>IOTS_DB method.</i>	
OTS_DB_MySQL::SQLquote()	57
<i>IOTS_DB method.</i>	
OTS_InfoRespond	73
<i>Wrapper for 'info' respond's DOMDocument.</i>	
OTS_Groups_List::valid()	73
<i>Checks if there are any rows left.</i>	
OTS_InfoRespond::getClientVersion()	73
<i>Returns dedicated version of client.</i>	
OTS_InfoRespond::getEmail()	74
<i>Returns owner e-mail.</i>	
OTS_InfoRespond::getLocation()	74
<i>Returns server location.</i>	
OTS_InfoRespond::getIP()	74

<i>Returns server IP.</i>	
OTS_Groups_List::setOffset()	72
<i>Sets OFFSET.</i>	
OTS_Groups_List::setLimit()	72
<i>Sets LIMIT.</i>	
OTS_Groups_List::deleteGroup()	70
<i>Deletes group.</i>	
OTS_Groups_List::current()	70
<i>Returns current row.</i>	
OTS_Groups_List::key()	71
<i>Current cursor position.</i>	
OTS_Groups_List::next()	71
<i>Moves to next row.</i>	
OTS_Groups_List::rewind()	71
<i>Select groups from database.</i>	
OTS_InfoRespond::getMapAuthor()	75
<i>Returns map author.</i>	
OTS_InfoRespond::getMapHeight()	75
<i>Returns map height.</i>	
OTS_InfoRespond::getPlayersPeak()	78
<i>Returns record of online players.</i>	
OTS_InfoRespond::getOwner()	78
<i>Returns owner name.</i>	
OTS_InfoRespond::getPort()	78
<i>Returns server port.</i>	
OTS_InfoRespond::getServer()	79
<i>Returns server attribute.</i>	
OTS_InfoRespond::getServerVersion()	79
<i>Returns server version.</i>	
OTS_InfoRespond::getOnlinePlayers()	77
<i>Returns current amount of players online.</i>	
OTS_InfoRespond::getName()	77
<i>Returns server name.</i>	
OTS_InfoRespond::getMapWidth()	76
<i>Returns map width.</i>	
OTS_InfoRespond::getMapName()	75
<i>Returns map name.</i>	
OTS_InfoRespond::getMaxPlayers()	76
<i>Returns maximum amount of players online.</i>	
OTS_InfoRespond::getMonstersCount()	76
<i>Returns number of all monsters on map.</i>	
OTS_InfoRespond::getMOTD()	77
<i>Returns server's Message Of The Day</i>	
OTS_Groups_List::count()	70
<i>Returns number of groups on list in current criterium.</i>	
OTS_Groups_List	69
<i>List of groups.</i>	
OTS_Group	61
<i>OTServ user group abstraction.</i>	
OTS_DB_SQLite::tableName()	61
<i>Query-quoted table name.</i>	
OTS_Group::getAccess()	62
<i>Access level.</i>	

OTS_Group::getCustomField()	62
<i>Reads custom field.</i>	
OTS_Group::getFlags()	63
<i>Rights flags.</i>	
OTS_DB_SQLite::SQLquote()	60
<i>IOTS_DB method.</i>	
OTS_DB_SQLite::SQLquery()	60
<i>IOTS_DB method.</i>	
OTS_DB_SQLite	58
<i>SQLite connection interface.</i>	
OTS_DB_MySQL::tableName()	57
<i>Query-quoted table name.</i>	
OTS_DB_SQLite::fieldName()	59
<i>Query-quoted field name.</i>	
OTS_DB_SQLite::limit()	59
<i>LIMIT/OFFSET clause for queries.</i>	
OTS_DB_SQLite::regexp()	59
<i>REGEXP operator for SQLite</i>	
OTS_Group::getId()	63
<i>Group ID.</i>	
OTS_Group::getMaxDepotItems()	64
<i>Maximum count of items in depot.</i>	
OTS_Group::setFlags()	67
<i>Sets rights flags.</i>	
OTS_Group::setCustomField()	67
<i>Writes custom field.</i>	
OTS_Group::setMaxDepotItems()	68
<i>Sets maximum count of items in depot.</i>	
OTS_Group::setMaxVIPList()	68
<i>Sets maximum count of players in VIP list.</i>	
OTS_Group::setName()	69
<i>Sets group's name.</i>	
OTS_Group::setAccess()	66
<i>Sets access level.</i>	
OTS_Group::save()	66
<i>Saves account in database.</i>	
OTS_Group::getName()	64
<i>Group name.</i>	
OTS_Group::getMaxVIPList()	64
<i>Maximum count of players in VIP list.</i>	
OTS_Group::getPlayers()	65
<i>List of characters in given group.</i>	
OTS_Group::isLoaded()	65
<i>Checks if object is loaded.</i>	
OTS_Group::load()	65
<i>Loads group with given id.</i>	
OTS.php	20
<i>This file contains main toolkit class.</i>	

P

POT::SLOT_NECKLACE	134
------------------------------------	-----

<i>Necklace slot.</i>	
POT::SLOT_RIGHT	134
<i>Right hand slot.</i>	
POT::SLOT_RING	135
<i>Ring slot.</i>	
POT::VOCATION_DRUID	135
<i>Druid.</i>	
POT::SLOT_LEGS	133
<i>Legs slot.</i>	
POT::SLOT_LEFT	133
<i>Left hand slot.</i>	
POT::SLOT_BACKPACK	132
<i>Backpack slot.</i>	
POT::SLOT_FEET	132
<i>Boots slot.</i>	
POT::SLOT_HEAD	133
<i>Head slot.</i>	
POT::VOCATION_KNIGHT	135
<i>Knight.</i>	
POT::VOCATION_NONE	136
<i>None vocation.</i>	
POT::loadClass()	138
<i>Loads POT class file.</i>	
POT::serverStatus()	139
<i>Queries server status.</i>	
POT::setPOTPath()	140
<i>Set POT directory.</i>	
POT::getInstance()	138
<i>Singleton.</i>	
POT::createObject()	138
<i>Creates OTServ DAO class instance.</i>	
POT::VOCATION_PALADIN	136
<i>Paladin.</i>	
POT::VOCATION_SORCERER	136
<i>Sorcerer.</i>	
POT::connect()	136
<i>Connects to database.</i>	
POT::SLOT_ARMOR	132
<i>Armor slot.</i>	
POT::SLOT_AMMO	131
<i>Ammunition slot.</i>	
POT::DIRECTION_EAST	127
<i>East.</i>	
POT::DIRECTION_NORTH	127
<i>North.</i>	
POT::DIRECTION_SOUTH	127
<i>South.</i>	
POT::DB_SQLITE	126
<i>SQLite driver.</i>	
POT::DB_MYSQL	126
<i>MySQL driver.</i>	
PHP 5.0	3
POT class preview	5

POT	126
<i>Main POT class.</i>	
POT::DIRECTION WEST	128
<i>West.</i>	
POT::SEX FEMALE	128
<i>Female gender.</i>	
POT::SKILL FIST	130
<i>Fist fighting.</i>	
POT::SKILL SHIELDING	130
<i>Shielding.</i>	
POT::SKILL SWORD	131
<i>Sword fighting.</i>	
POT::SKILL FISHING	130
<i>Fishing.</i>	
POT::SKILL DISTANCE	129
<i>Distance fighting.</i>	
POT::SEX MALE	128
<i>Male gender.</i>	
POT::SKILL AXE	129
<i>Axe fighting.</i>	
POT::SKILL CLUB	129
<i>Club fighting.</i>	
POT	1

Q

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

R

README	148
------------------------	-----

S

Server online status	14
--------------------------------------	----