POT

Contents

POT

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

PHP OTServ Toolkit

There are several reasons why POT was created:

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

POT has been created for latest SVN release, it won't work with old database structure as well as with broken database - it ralies on database foreign key contraints, triggers etc.

System requirements

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

What POT is

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

What POT is not

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

What about XML?

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

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

How to use

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

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

Link

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

You can use following code for that:

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

POT class preview

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

What it is

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

Creating instance of POT class

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

__autoload() and POT classes

PHP5 provides nice autoloading mechanism. You can combine POT class loading mechanism with it. For example:

```
<?php

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

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

function __autoload($class)
{
    // checks if it's POT class
    if( preg_match('/^I?OTS_/',  $class) !=0)
    {
        POT::getInstance()->  loadClass(  $class);
    }
/*
    // possibly call your own __autoload() handler
    else
    {
        here comes your stuff...
    }
*/
}

?>
```

DAO classes

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

Quick start

Quick start guide.

Putting this all together

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

```
<?php

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

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

// for further POT classes
function __autoload($class)
{
    // checks if it's POT class
    if( preg_match('/^I?OTS_/',  $class) !=0)
    {
        POT::getInstance()->    loadClass(  $class);
    }
/*
    // possibly call your own __autoload() handler
    else
    {
        here comes your stuff...
    }
*/
}

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

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

?>
```

Account creation

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

```php
<?php

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

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

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

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

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

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

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

?>
```

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:

```php
<?php

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

// to not repeat all that stuff
```

```
include( 'quickstart.php');

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

// loads player
$player->find('Wrzasq');

// checks if player exists
if( $player->isLoaded() )
{
    // prints character info
    echo 'Player \'' . $player->getName() . '\' has ' . $player->getLevel() . ' level.', "\n";

    // example of associated objects retriving
    echo 'Player \'' . $player->getName() . '\' is member of ' . $player->getGroup()->getName() . ' group.', "\n";
}
else
{
    echo 'Player does not exists.', "\n";
}

?>
```

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:

```
<?php

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

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

// creates new OTS_Player object
$players = $ots->createObject('Players_List');

// count of all players - Countable interface implemented
echo 'There are ' . count( $players) . ' players in our database.', "\n";

// sets limitation
$players->setLimit(10);
$players->setOffset(2);

// iterates throught selected players
foreach($players as $index => $player)
{
    // each returned item is instance of OTS_Player class
    echo (2 + $index) . ': ' . $player->getName(), "\n";
}

?>
```

Account number hack

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

Walkaround

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

```php
<?php

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

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

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

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

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

?>
```

Package POT Procedural Elements

IOTS_DAO.php

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

IOTS_DB.php

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

OTS.php

This file contains main toolkit class.

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

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

OTS_Account.php

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

OTS_DB_MySQL.php

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

OTS_DB_SQLite.php

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

OTS_Group.php

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

OTS_Groups_List.php

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

OTS_Player.php

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

OTS_SQLite_Results.php

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

Package POT Classes

Class IOTS_DAO

[line 21]

OTserv database object.

OTserv database object.

This insterface 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** 0.0.1+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 262]*

Blocks account.

Blocks account.

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

int function OTS_Account::create([$min = 1], [$max = 9999999]) *[line 62]*

account.php

```php
<?php

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

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

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

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

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

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

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

?>
```

Function Parameters:

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

Creates new account.

Creates new account.

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

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

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

void function OTS_Account::find($email) *[line 126]*

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

string|bool function OTS_Account::getEMail() *[line 214]*

E-mail address.

E-mail address.

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

int|bool function OTS_Account::getId() *[line 172]*

Account number.

Account number.

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

int|bool function OTS_Account::getPACCDays() *[line 272]*

PACC days.

PACC days.

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

string|bool function OTS_Account::getPassword() *[line 188]*

Account's password.

Account's password.

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

array|bool function OTS_Account::getPlayers() *[line 298]*

List of characters on account.

List of characters on account.

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

bool|null function OTS_Account::isBlocked() *[line 240]*

Checks if account is blocked.

Checks if account is blocked.

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

bool function OTS_Account::isLoaded() *[line 143]*

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

bool function OTS_Account::save() *[line 153]*

Updates account in database.

Updates account in database.

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

void function OTS_Account::setEMail($email) *[line 230]*

Function Parameters:

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

Sets account's email.

Sets account's email.

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

void function OTS_Account::setPACCDays($premdays, $pacc) *[line 288]*

Function Parameters:

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

Sets PACC days count.

Sets PACC days count.

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

void function OTS_Account::setPassword($password) *[line 204]*

Function Parameters:

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

Sets account's password.

Sets account's password.

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

void function OTS_Account::unblock() *[line 254]*

Unblocks account.

Unblocks account.

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

Class OTS_Accounts_List

[line 19]

List of accounts.

List of accounts.

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

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

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

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

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

Returns number of accounts on list in current criterium.

Returns number of accounts on list in current criterium.

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

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

Returns current row.

Returns current row.

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

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

Function Parameters:

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

Deletes account.

Deletes account.

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

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

Current cursor position.

Current cursor position.

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

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

Moves to next row.

Moves to next row.

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

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

Select accounts from database.

Select accounts from database.

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

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

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

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

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

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

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

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

Checks if there are any rows left.

Checks if there are any rows left.

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

Class OTS_DB_MySQL

[line 19]

MySQL connection interface.

MySQL connection interface.

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

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

Function Parameters:

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

Creates database connection.

Creates database connection.

Connects to MySQL database on given arguments.

List of parameters for this drivers:

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

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

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

Function Parameters:

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

Query-quoted field name.

Query-quoted field name.

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

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

Function Parameters:

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

LIMIT/OFFSET clause for queries.

LIMIT/OFFSET clause for queries.

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

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

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

Overwrites PDO method.

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

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

Function Parameters:

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

IOTS_DB method.

IOTS_DB method.

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

- **Version** 0.0.1
- **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:

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

IOTS_DB method.

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

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

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

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

Query-quoted table name.

Query-quoted table name.

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

Class OTS_Group

[line 19]

OTServ user group abstraction.

OTServ user group abstraction.

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

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

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

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

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

Access level.

Access level.

- **Version** 0.0.1

- **Access** public

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

Rights flags.

Rights flags.

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

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

Group ID.

Group ID.

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

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

Maximum count of items in depot.

Maximum count of items in depot.

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

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

Maximum count of players in VIP list.

Maximum count of players in VIP list.

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

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

Group name.

Group name.

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

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

List of characters in given group.

List of characters in given group.

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

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

Checks if object is loaded.

Checks if object is loaded.

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

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

Function Parameters:

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

Loads group with given id.

Loads group with given id.

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

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

Saves account in database.

Saves account in database.

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

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

Function Parameters:

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

Sets access level.

Sets access level.

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

void function OTS_Group::setFlags($flags) *[line 150]*
Function Parameters:

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

Sets rights flags.

Sets rights flags.

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

void function OTS_Group::setMaxDepotItems($maxdepotitems) *[line 202]*
Function Parameters:

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

Sets maximum count of items in depot.

Sets maximum count of items in depot.

- **Version** 0.0.1

- **Access** public

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

Function Parameters:

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

Sets maximum count of players in VIP list.

Sets maximum count of players in VIP list.

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

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

Function Parameters:

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

Sets group's name.

Sets group's name.

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

Class OTS_Groups_List

[line 19]

List of groups.

List of groups.

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

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

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

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

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

Returns number of groups on list in current criterium.

Returns number of groups on list in current criterium.

- **Version** 0.0.1

- **Access** public

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

Returns current row.

Returns current row.

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

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

Function Parameters:

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

Deletes group.

Deletes group.

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

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

Current cursor position.

Current cursor position.

- **Version** 0.0.1

- **Access** public

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

Moves to next row.

Moves to next row.

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

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

Select groups from database.

Select groups from database.

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

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

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

- **Version** 0.0.1

- **Access** public

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

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

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

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

Checks if there are any rows left.

Checks if there are any rows left.

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

Class OTS_Player

[line 21]

OTServ character abstraction.

OTServ character abstraction.

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

Constructor *void* function OTS_Player::__construct($db) *[line 51]*

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 82]*

Function Parameters:

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

Loads player by it's name.

Loads player by it's name.

- **Version** 0.0.1

- **Access** public

OTS_Account function OTS_Player::getAccount() *[line 180]*

Returns account of this player.

Returns account of this player.

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

int|*bool* function OTS_Player::getCap() *[line 782]*

Capacity.

Capacity.

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

mixed|*bool* function OTS_Player::getConditions() *[line 892]*

Conditions.

Conditions.

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

int|*bool* function OTS_Player::getDirection() *[line 522]*

Looking direction.

Looking direction.

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

int|*bool* function OTS_Player::getExperience() *[line 288]*

Experience points.

Experience points.

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

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

Returns group of this player.

Returns group of this player.

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

string|*bool* function OTS_Player::getGuildNick() *[line 976]*

Guild nick.

Guild nick.

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

int|*bool* function OTS_Player::getHealth() *[line 366]*

Current HP.

Current HP.

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

int|*bool* function OTS_Player::getHealthMax() *[line 392]*

Maximum HP.

Maximum HP.

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

int|*bool* function OTS_Player::getId() *[line 138]*

Player ID.

Player ID.

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

int|*bool* function OTS_Player::getLastIP() *[line 834]*

Last login IP.

Last login IP.

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

int|*bool* function OTS_Player::getLastLogin() *[line 808]*

Last login timestamp.

Last login timestamp.

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

int|*bool* function OTS_Player::getLevel() *[line 314]*

Experience level.

Experience level.

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

int|*bool* function OTS_Player::getLookAddons() *[line 678]*

Addons.

Addons.

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

int|*bool* function OTS_Player::getLookBody() *[line 548]*

Body color.

Body color.

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

int|*bool* function OTS_Player::getLookFeet() *[line 574]*

Boots color.

Boots color.

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

int|*bool* function OTS_Player::getLookHead() *[line 600]*

Hair color.

Hair color.

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

int|*bool* function OTS_Player::getLookLegs() *[line 626]*

Legs color.

Legs color.

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

int|*bool* function OTS_Player::getLookType() *[line 652]*

Outfit.

Outfit.

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

int|*bool* function OTS_Player::getLossExperience() *[line 1052]*

Percentage of experience lost after dead.

Percentage of experience lost after dead.

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

int|*bool* function OTS_Player::getLossMana() *[line 1077]*

Percentage of used mana lost after dead.

Percentage of used mana lost after dead.

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

int|*bool* function OTS_Player::getLossSkills() *[line 1102]*

Percentage of skills lost after dead.

Percentage of skills lost after dead.

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

int|*bool* function OTS_Player::getMagLevel() *[line 340]*

Magic level.

Magic level.

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

int|*bool* function OTS_Player::getMana() *[line 418]*

Current mana.

Current mana.

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

int|*bool* function OTS_Player::getManaMax() *[line 444]*

Maximum mana.

Maximum mana.

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

int|*bool* function OTS_Player::getManaSpent() *[line 470]*

Mana spent.

Mana spent.

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

string|*bool* function OTS_Player::getName() *[line 154]*

Player name.

Player name.

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

int|*bool* function OTS_Player::getPosX() *[line 704]*

X map coordinate.

X map coordinate.

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

int|*bool* function OTS_Player::getPosY() *[line 730]*

Y map coordinate.

Y map coordinate.

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

int|*bool* function OTS_Player::getPosZ() *[line 756]*

Z map coordinate.

Z map coordinate.

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

int|*bool* function OTS_Player::getRankId() *[line 1002]*

Guild rank ID.

Guild rank ID.

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

int|bool function OTS_Player::getRedSkullTime() *[line 918]*

Red skulled time remained.

Red skulled time remained.

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

int|bool function OTS_Player::getSex() *[line 236]*

Player gender.

Player gender.

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

int function OTS_Player::getSkill($skill) *[line 1130]*

Function Parameters:

- *int* **$skill** Skill ID.

Returns player's skill.

Returns player's skill.

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

int function OTS_Player::getSkillTries($skill) *[line 1154]*

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

int|*bool* function OTS_Player::getSoul() *[line 496]*

Soul points.

Soul points.

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

int|*bool* function OTS_Player::getTownId() *[line 1027]*

Residence town's ID.

Residence town's ID.

- **Version** 0.0.1

- **Access** public

int|*bool* function OTS_Player::getVocation() *[line 262]*

Player proffesion.

Player proffesion.

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

bool|*null* function OTS_Player::hasRedSkull() *[line 944]*

Checks if player has red skull.

Checks if player has red skull.

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

bool function OTS_Player::isLoaded() *[line 99]*

Checks if object is loaded.

Checks if object is loaded.

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

bool|*null* function OTS_Player::isSaveSet() *[line 860]*

Checks if save flag is set.

Checks if save flag is set.

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

void function OTS_Player::load($id) *[line 62]*

Function Parameters:

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

Loads player with given id.

Loads player with given id.

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

void function OTS_Player::save() *[line 109]*

Saves account in database.

Saves account in database.

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

void function OTS_Player::setAccount($account) *[line 198]*

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 798]*

Function Parameters:

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

Sets capacity.

Sets capacity.

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

void function OTS_Player::setConditions($conditions) *[line 908]*

Function Parameters:

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

Sets conditions.

Sets conditions.

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

void function OTS_Player::setDirection($direction) *[line 538]*

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 304]*

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 226]*

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 992]*

Function Parameters:

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

Sets guild nick.

Sets guild nick.

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

void function OTS_Player::setHealth($health) *[line 382]*

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 408]*

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 850]*

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 824]*

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 330]*

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 694]*

Function Parameters:

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

Sets addons.

Sets addons.

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

void function OTS_Player::setLookBody($lookbody) *[line 564]*

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 590]*

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 616]*

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 642]*

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 668]*

Function Parameters:

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

Sets outfit.

Sets outfit.

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

void function OTS_Player::setLossExperience($loss_experience) *[line 1068]*

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 1093]*

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 1118]*
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 356]*
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 434]*

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 460]*

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 486]*

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 170]*

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 720]*

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 746]*

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 772]*

Function Parameters:

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

Sets Z map coordinate.

Sets Z map coordinate.

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

void function OTS_Player::setRankId($rank_id) *[line 1018]*

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 966]*

Sets red skull flag.

Sets red skull flag.

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

void function OTS_Player::setRedSkullTime($redskulltime) *[line 934]*

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 882]*

Sets save flag.

Sets save flag.

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

void function OTS_Player::setSex($sex) *[line 252]*

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 1142]*

Function Parameters:

- *int* **$skill** Skill ID.
- *int* **$value** Skill value.

Sets skill value.

Sets skill value.

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

void function OTS_Player::setSkillTries($skill, $tries) *[line 1166]*

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

void function OTS_Player::setSoul($soul) *[line 512]*

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 1043]*

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 278]*

Function Parameters:

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

Sets player proffesion.

Sets player proffesion.

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

void function OTS_Player::unsetRedSkull() *[line 958]*

Unsets red skull flag.

Unsets red skull flag.

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

void function OTS_Player::unsetSave() *[line 874]*

Unsets save flag.

Unsets save flag.

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

Class OTS_Players_List

[line 19]

List of players.

List of players.

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

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

Function Parameters:

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

Sets database connection handler.

Sets database connection handler.

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

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

Returns number of characters on list in current criterium.

Returns number of characters on list in current criterium.

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

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

Returns current row.

Returns current row.

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

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

Function Parameters:

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

Deletes player.

Deletes player.

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

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

Current cursor position.

Current cursor position.

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

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

Moves to next row.

Moves to next row.

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

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

Select players from database.

Select players from database.

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

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

Function Parameters:

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

Sets LIMIT.

Sets LIMIT.

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

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

Function Parameters:

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

Sets OFFSET.

Sets OFFSET.

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

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

Checks if there are any rows left.

Checks if there are any rows left.

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

Class POT

[line 23]

Main POT class.

Main POT class.

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

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 104]*

Axe fighting.
Axe fighting.

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

POT::SKILL_CLUB

= 1 *[line 92]*

Club fighting.
Club fighting.

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

POT::SKILL_DISTANCE

= 4 *[line 110]*

Distance fighting.

Distance fighting.

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

POT::SKILL_FISHING

= 6 *[line 122]*

Fishing.

Fishing.

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

POT::SKILL_FIST

= 0 *[line 86]*

Fist fighting.

Fist fighting.

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

POT::SKILL_SHIELDING

= 5 *[line 116]*

Shielding.

Shielding.

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

POT::SKILL_SWORD

= 2 *[line 98]*

Sword fighting.

Sword fighting.

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

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

Constructor *void* function POT::__construct() *[line 179]*

Class initialization tools.

Class initialization tools.

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

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

void function POT::connect($driver, $params) *[line 236]*

connect.php

```php
<?php

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

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

```

// you can easily store such structure in config.php
$config = array(
    'driver' =>    POT::DB_MYSQL,
    'prefix' =>    '',
    'host' =>    'localhost',
    'user' =>    'wrzasq',
    'password' =>    '',
    'database' =>    'otserv'
);

// connects to database
$ots = POT::getInstance();
$ots->   connect(null, $config);
// could be: $ots->connect(POT::DB_MYSQL, $config);

?>
```

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, aplies when *$driver* method parameter is *null*
- *prefix* - optional, prefix for database tables, use if you have more then one OTServ installed on one database.

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

IOTS_DAO function POT::createObject($class) *[line 281]*

Function Parameters:

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

Creates OTServ DAO class instance.

Creates OTServ DAO class instance.
Currently it means Account, or Player object.

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

POT function POT::getInstance() *[line 129]*

Singleton.

Singleton.

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

void function POT::loadClass($class) *[line 194]*

autoload.php

```php
<?php

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

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

function __autoload($class)
{
    // checks if it's POT class
    if( preg_match('/^I?OTS_/', $class) != 0)
    {
        POT::getInstance()->   loadClass($class);
    }
/*
    // possibly call your own __autoload() handler
    else
    {
        here comes your stuff...
    }
*/
}

?>
```

Function Parameters:

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

Loads POT class file.

Loads POT class file.

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

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

void function POT::setPOTPath($path) *[line 160]*

fakeroot.php

```php
<?php

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

// this is the way you should work with POT if you moved main OTS.php file outside POT's directory
include('path/to/OTS.php');

// dont use 'new POT()'!!!
$ots = POT::getInstance();
$ots->   setPOTPath('../classes/');

/*
    here comes your stuff...
*/
```

```

?>
```

Function Parameters:

- *string* **$path** POT files path.

Set POT directory.

Set POT directory.

Use this method if you keep your POT package in different directory then this file.

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

Appendices

Appendix A - Class Trees

Package POT

IOTS_DAO

- IOTS_DAO

IOTS_DB

- IOTS_DB

OTS_Account

- OTS_Account

OTS_Accounts_List

- OTS_Accounts_List

OTS_DB_MySQL

- PDO
  - OTS_DB_MySQL

OTS_DB_SQLite

- PDO
  - OTS_DB_SQLite

OTS_Group

- OTS_Group

OTS_Groups_List

- OTS_Groups_List

OTS_Player

- OTS_Player

OTS_Players_List

- OTS_Players_List

POT

- POT

Appendix B - README/CHANGELOG/INSTALL

INSTALL

POT is a toolkit which means you don't literaly install it. You copy it's files and write code for it. All source files are located in classes/ subdirectory. Copy them to your script directory.

You can put main file - OTS.php in different directory then other files.

For information about how to include POT in your code see the documentation.

CHANGELOG

[SVN]
* Added skills support in OTS_Player class. <wrzasq>
* HTML documentation removed from SVN (pointless to update it all the time, you can re-create it with phpdoc and make). <wrzasq>
* Fixed `redskulltime` field name in OTS_Player. <wrzasq>
* Added find() to OTS_Account class to load accounts by their's e-mail addresses. <wrzasq>
* Documentation fixes. <wrzasq>
* Additional info/example. <wrzasq>

[0.0.1]
* Initial release. <wrzasq>

NEWS

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

README

POT (PHP OTServ Toolkit) is a PHP toolkit for scripts that work with OTServ database.

===== About =====

This toolkit provides a way for PHP programmers that don't know SQL langauge to work with OTServ database.

For installation help check INSTALL file.

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

===== Contact =====

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

===== Files =====

classes/ - POT class files.
documenation/ - phpDocumentor-generater documentation.
examples/ - example files for learning.
tutorials/ - phpDocumentor directory.
BUGS - known bugs.
CHANGELOG - changes history.
INSTALL - installation tutorial.
LICENSE - POT license (GNU LGPL v3), if you don't accept it - don't use any of those scripts.
NEWS - changes in current release.
README - this readme file.
RULES - rules to be followed during developing contributed code.
TODO - list of things to be done.
Makefile - make input, for documentation generation.
test.php - phpUnit test suite.

===== Makefile =====

Makefile contains some targets for make that can help in development. Makefile requires following command-line commands:

php: PHP CLI interface.
phpdoc: phpDocumentor.
phpunit: PHPUnit testing framework.

Possible targets:

all: default one, runs all other targets (in order: clean, check, documentation, pdf, otserv-aac, test).
clean: deletes documentation.
check: checks syntax of all PHP files.
documentation: generates HTML documentation.
pdf: generates PDF documentation.
otserv-aac: OTServ-AAC website documentation template used.
test: runs test suite.

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

===== Credits =====

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

Index