



## object relational mapping framework for php

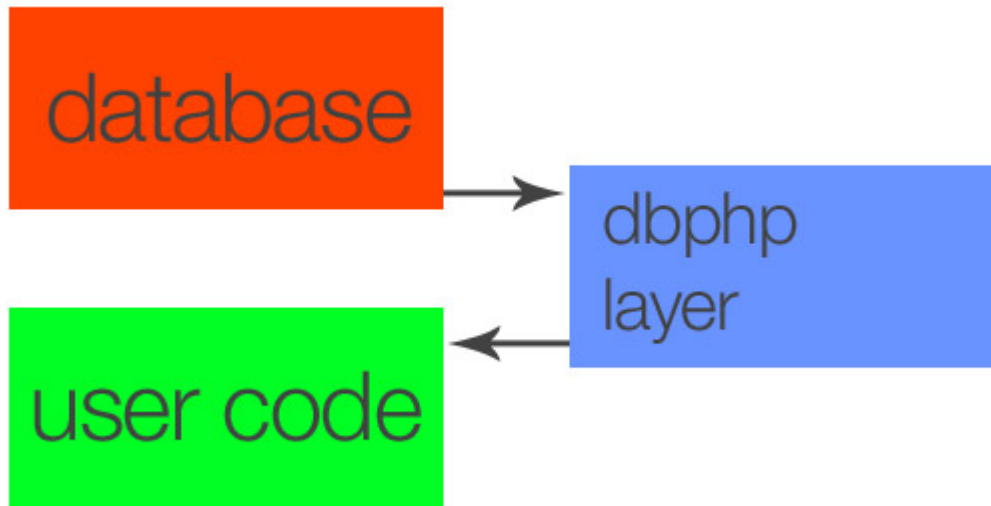
Title	dbphp user manual
Description	Technical manual for dbphp object relational mapping framework
Author	Mike Lang
Project page	<a href="http://www.dbphp.co.uk">http://www.dbphp.co.uk</a>
Version	0.1
Date	21/05/2009

## Contents

Getting Started.....	3
Object Relational Mapping .....	3
PHP version 4 and prior .....	3
Class Structures.....	3
Implementing dbphp .....	5
Functionality .....	7
Setup functions .....	7
getDbTable('string').....	7
getDbKey('string').....	7
setId('int').....	7
add('table_name','classVariable').....	7
Functionality (in order of usefulness) .....	7
persist() .....	7
delete().....	7
find(\$id).....	7
search(\$key, \$value) .....	7
getAll().....	7
getRandom() .....	7
enableDebug()/disableDebug().....	8
getError().....	8
show() .....	8
purgeTable().....	8
Open Source.....	9
Updates and Developments.....	9

## Getting Started

In order to use dbphp it is important to understand the principals upon which it relies. It acts as a layer of functionality between user classes and database information.



dbphp can either be included in existing projects by modifying the code or can be used to generate all new code.

### Object Relational Mapping

In order to allow for the easy flow of data between the code layer and the database layer all objects should be object relationally mapped (ORM). This means that each class concerned with handling the loading and storage of entities or information from the database, as far as possible should replicate the structure of the data held within the database.

There are a number of advantages to this approach. Implementing a functioning ORM structure also means that you will probably be taking full advantage of object orientation as well as taking advantage of the ability of objects to store their internal information privately and accessing them only via public get/set functions. Another huge advantage comes when you serialize your object to XML as it will retain its structure.

### PHP version 4 and prior

PHP is a highly flexible layout language for making websites. One of its strengths, and also its weaknesses, is that it lets you do just about anything with it, meaning that you can hack together any old rubbish, ignoring all the principals of good design and it will still work.

This is great for small projects, but when you need to implement any changes to your work you will throw your hands up in anguish. If you have ignored all the principals of good design you will probably be copying and pasting 100s of lines of repeated code and using the much maligned global variable.

### Class Structures

In an ideal world you should probably be designing your database around your information needs and then developing your code on top of this to reflect the structure of the database. As we do

indeed live in an ideal world we shall assume that this is exactly what you've done and that you're not desperately trying to hack a fix for some awful database structure you designed back in 2002.

For arguments sake our table looks like this and is called 'table'

Column	Type
Id	Int (auto increment)
first_name	Varchar(64)
last_name	Varchar(64)

Now in order to mirror the structure of this table in PHP you would make something that looked a bit like this:

```
class Table{
    private $id = null;
    private $firstName = null;
    private $lastName = null;

    public function getId(){
        return $this->id;
    }

    public function setId($id){
        $this->id = $id;
    }

    public function getFirstName(){
        return $this->firstName;
    }

    public function setFirstName($firstName){
        $this->firstName = $firstName;
    }

    public function getLastName(){
        return $this->lastName;
    }

    public function setLastName($lastName){
        $this->lastName = $lastName;
    }
}
```

This can all be done using the utilities found on dbphp.co.uk, but you're probably best of downloading development environment like Netbeans – from netbeans.org as this will allow you to do lots of other 'exciting' things as well as showing you where you've gone wrong with syntax highlighting.

## Implementing dbphp

So you've made your class, and if PHP were strictly typed like Java, it would look even nicer.

Just in case you are not sure on the instantiation thing; classes are not just nice neat receptacles in which to dump loads of functions so you know where they are. Classes are in fact capable of becoming objects in memory, complex versions of your average integer or string, and then all of the public functions defined within them can be utilized to do useful things.

To instantiate a copy of our class what we do is:

```
$myTable = new Table();
```

\$myTable can now have function calls made on it to add and retrieve information from its internal structure like:

```
$myTable->setFirstName('mike');
$string = $myTable->getFirstName();
```

\$string will now contain 'mike'

dbphp takes advantage of the availability of PHP's ability to extend parent classes and also execute code within constructor functions when a class is instantiated.

Assuming you have copied the dbphp libs directory (the default container for dbphp) to your server and you have included relevant files you can now extend the class that we have just made. You should have something that looks a bit like this.

```
<?php
    Include('libs/DbOneToOne.php');

    Class Table extends DbOneToOne {
        private $id = null;
        private $firstName = null.....
```

Now it's time to create the constructor, as soon as this is done you're all set to actually do something vaguely useful with it all. The constructor should be the same name as the class and then will be automatically executed by the 'compiler' (php isn't really compiled) when the class is instantiated.

For our class the dbphp constructor should look something like this

```
public function Table(){
    $this->setDbTable('table');
    $this->setDbKey('id');
    $this->setId($this->id);
    $this->add("first_name",$this->firstName);
    $this->add("last_name", $this->lastName);
}
```

Ok, that's great, you've done it, you've already created much nicer code than most of the existing PHP examples on the net. Now just drop this into the class we have created and whenever you instantiate it will tell the class which of its variables map to which columns in the database.

One important note at this stage is that the **id** column for the table must be set to **auto-increment** otherwise you will get an error from the database telling you it has no clue how to insert your data into the table – this is an issue that will be handled in future releases, but for now you can consider it all part of the rich tapestry of open source.

Full documentation on all functionality is provided in the next section.

## Functionality

### Setup functions

These functions are all set up in the constructor of the user class implementing dbphp functionality.

#### **setDbTable('string')**

Sets up the link between the class and the table

#### **setDbKey('string')**

Tells dbphp what the table key is (and assumes its auto-increment)

#### **setId('int')**

Tells dbphp where you are storing the class id variable and sends it to the parent

#### **add('table\_name','classVariable')**

Tells dbphp which class variables are linked to which database columns

### Functions (in order of usefulness)

#### **persist()**

Saves/updates a record back to the database – if there is no ID set it will create a new record else updates an existing one

#### **delete()**

Deletes the current record from the database. It will only delete a record if you have one loaded.

#### **find(\$id)**

Searches for a record corresponding to the \$id variable passed into it and populates the current class accordingly. Returns True or False depending on success or failure.

#### **search(\$key, \$value)**

Searches for any column/value match and populates the current class. Returns True/False depending on success or failure.

#### **getAll()**

Returns an array of objects of the current type of all values in the table. This can be used like this:

```
foreach($var->getAll() as $key => $val){
    $val->callObjectFunction();
}
```

#### **getRandom()**

Gets a random value from the table, seems silly, but it's really useful for picture galleries and all kinds of web apps

**enableDebug()/disableDebug()**

This is disabled by default, but when enabled will give you a screen print of what queries are being run along with the memory structure of the instantiated class

**getError()**

More debug functionality, this will return the last error generated by the database – this will be handled differently in future versions

**show()**

This just prints out the memory structure of the current instantiation.

**purgeTable()**

Should you want to erase all from your chosen table then this will do it for you.

## Open Source

Open source is fantastic, it's your chance to stick two fingers up at the evil capitalist oppressors, or if you're less radically minded, save a few quid on your software procurement budget. It should be noted that open source is actually about a reciprocal exchange of information, which means if you use it and you make it better, you should in turn return your contribution to the community.

An economy of knowledge based on freedom is clearly the way forward, which is why those crazy, radical, free thinking artists use Apple Macs (with Mac Os) because; Apple is in fact a giant benevolent brain trust whose only interest is world peace and free knowledge for all.

Ideology aside, you can involve yourself with dbphp (and many other worthy projects) on

**[sourceforge.net/projects/dbphp](http://sourceforge.net/projects/dbphp)**

If you have Netbeans (or Eclipse or anything else that handles SVN (subversion control system)) you can check out a copy of dbphp right away and play with it to your hearts content.

## Updates and Developments

Being firm proponents of web 2.0 and all future web iterations, as well as being a firm believer in the huge socio-intellectual benefits of **twitter**, **facebook** and other miscellaneous social network detritus, we've joined up!

**[twitter.com/dbphp](http://twitter.com/dbphp)**

Other useful links can be found on dbphp.co.uk which has all relevant links and documentation along with examples of implementation and other development information.

Have fun.