Mongo Mapper

The Mongo Object-Document-Mapper (ODM) is an implementation of the abstract Active Record Cursor class (cursor). Have a look into it for inherited method descriptions.

Namespace: `\DB\Mongo`
File location: `lib/db/mongo/mapper.php`

Instantiation

To use the Mongo ODM, create a valid MongoDB Connection (mongo#constructor) and follow this example:

```
$mapper = new \DB\Mongo\Mapper(\DB\Mongo $db, string $collection);
```

If you like to create a model class, you might like to wrap it up:

```
$f3->set('DB', new \DB\Mongo('mongodb://localhost:27017','fatfree'));

class User extends \DB\Mongo\Mapper {
    public function __construct() {
        parent::__construct( \Base::instance()->get('DB'), 'users' );
    }
}

$user = new User();
$user->load(array('_id' => new \MongoId('50ecaa466afa2f8c1c000004')));
// etc.
```

Syntax

$filter

The `$filter` argument for Mongo accepts the common mongo find array, see db.collection.find reference (http://docs.mongodb.org/manual/reference/method/db.collection.find/) and query-documents tutorial (http://docs.mongodb.org/manual/tutorial/query-documents/).

```
array([ array $find ]);
```

Search

Here is a simple example how to search in mongo collections:

```
$userList = $user->find(array('email'=> new \MongoRegex('/gmail/')));
// returns all users with an email address that contains GMAIL

// ends with gmail.com => /gmail\.com$/
// starts with john  => /^john/
```

Or just load a single user by its ID:

```
$user->load(array('_id'=> new \MongoId('507c35dd8fada716c89d0013')));
```

$option

The `$option` argument for Mongo accepts the following structure:

```
array(
    'group' => array (
        'keys',
        'initial',
        'reduce',
        'finalize'
    ),
    'order' => string $order,
    'limit' => integer $limit,
    'offset' => integer $offset
    )
```

For more details about `group`, go to the db.collection.group reference (http://docs.mongodb.org/manual/reference/method/db.collection.group/). And for more about `order`, and a look at the cursor.sort reference (http://docs.mongodb.org/manual/reference/method/cursor.sort/).

Methods

exists

Return TRUE if the given field is defined

```
bool exists( string $key )
```

set

Assign a value to a field

```
scalar|FALSE set( string $key, scalar $val )
```

This class takes advantage of the Magic class (magic) and ArrayAccess interface. It means you can set and get variables with direct access like this:

```
$mapper->foo = 'bar';
$mapper['foo'] = 'bar';
```

get

Retrieve the value of a given field

```
scalar|FALSE get( string $key )
```

clear

Delete a field

```
NULL clear( string $key )
```

cast

Return the fields of a mapper object as an associative array

```
array cast( [ object $obj = NULL ] )
```

select

Build query and execute

```
array select( [ string $fields = NULL [, array $filter = NULL [, array $options = NUL
L [, int $ttl = 0 ]]]] )
```

find

Return records that match a given criteria

```
array find( [ array $filter = NULL [, array $options = NULL [, int $ttl = 0 ]]] )
```

count

Count records that match a given criteria

```
int count( [ array $filter = NULL [, int $ttl = 0 ]] )
```

insert

Insert a new record

```
array insert()
```

update

Update the current record

```
array update()
```

erase

Delete the current record

```
bool erase( [ array $filter = NULL ] )
```

copyfrom

Hydrate the mapper object using an array

```
NULL copyfrom( array | string $var [, callback $func = NULL ] )
```

This function allows you to hydrate the mapper using an array (or the name of a hive variable containing an array).

`$func` is the callback function to apply to the hive array variable:

```
if ($func)  $var = $func($var);
```

copyto

Populate hive array variable with mapper fields

```
NULL copyto( string $key )
```

factory

Convert an array to a mapper object

```
protected object factory( array $row )
```

This *protected* method is used internally by the `select` method.