MongoDB PHP Library Documentation

Install

Prerequisites

The MongoDB PHP Library is a high-level abstraction for the MongoDB PHP driver. As such, you must install the *mongodb* extension to use the library.

Installing the MongoDB PHP Driver describes how to install the *mongodb* extension for PHP. Instructions for installing the driver for HHVM may be found in the Installation with HHVM article.

Procedure

Install the Library

The preferred method of installing MongoDB PHP Library is with Composer by running the following from your project root:

```
composer require mongodb/mongodb
```

While not recommended, you may also manually install the package via the source tarballs attached to the GitHub releases.

Configure Autoloading

Once you have installed the library, ensure that your application includes Composer’s autoloader. The `require_once` statement should point to Composer’s autoloader, as in the following example:

```
require_once __DIR__ . "/vendor/autoload.php";
```

Refer to Composer’s autoloading documentation for more information about setting up autoloading.

If you installed the library manually from a source tarball, you will also need to manually configure autoloading:

1. Map the top-level `MongoDB\` namespace to the `src/` directory using your preferred autoloader implementation.
2. Manually require the `src/functions.php` file, since PHP does not yet support function autoloading.

Library

MongoDB\Client Class

Definition

`MongoDB\Client`

This class serves as an entry point for the MongoDB PHP Library. It is the preferred class for connecting to a MongoDB server or cluster of servers and acts as a gateway for accessing individual databases and collections. `MongoDB\Client` is analogous to the driver's MongoDB\Driver\Manager class, which it composes.

Methods

- MongoDB\Client::__construct()
- MongoDB\Client::__get()
- MongoDB\Client::dropDatabase()
- MongoDB\Client::getManager()
- MongoDB\Client::getReadConcern()
- MongoDB\Client::getReadPreference()
- MongoDB\Client::getTypeMap()
- MongoDB\Client::getWriteConcern()
- MongoDB\Client::listDatabases()
- MongoDB\Client::selectCollection()
- MongoDB\Client::selectDatabase()

MongoDB\Database Class

Definition

`MongoDB\Database`

Provides methods for common operations on a database, such as executing database commands and managing collections.

You can construct a database directly using the driver's MongoDB\Driver\Manager class or select a database from the library's `MongoDB\Client` class. A database may also be cloned from an existing `MongoDB\Database` object via the `withOptions()` method.

`MongoDB\Database` supports the readConcern, readPreference, typeMap, and writeConcern options. If you omit an option, the database inherits the value from the Manager constructor argument or the `Client` object used to select the database.

Operations within the `MongoDB\Database` class inherit the Database's options.

Methods

- MongoDB\Database::__construct()
- MongoDB\Database::__get()
- MongoDB\Database::command()
- MongoDB\Database::createCollection()
- MongoDB\Database::drop()
- MongoDB\Database::dropCollection()
- MongoDB\Database::getDatabaseName()
- MongoDB\Database::getManager()
- MongoDB\Database::getReadConcern()
- MongoDB\Database::getReadPreference()
- MongoDB\Database::getTypeMap()
- MongoDB\Database::getWriteConcern()
- MongoDB\Database::listCollections()
- MongoDB\Database::selectCollection()
- MongoDB\Database::selectGridFSBucket()
- MongoDB\Database::withOptions()

MongoDB\Collection Class

Definition

`MongoDB\Collection`

Provides methods for common operations on collections and documents, including CRUD operations and index management.

You can construct collections directly using the driver's MongoDB\Driver\Manager class or select a collection from the library's `MongoDB\Client` or `MongoDB\Database` classes. A collection may also be cloned from an existing `MongoDB\Collection` object via the `withOptions()` method.

`MongoDB\Collection` supports the readConcern, readPreference, typeMap, and writeConcern options. If you omit an option, the collection inherits the value from the Manager constructor argument or the `Client` or `Database` object used to select the collection.

Operations within the `MongoDB\Collection` class inherit the collection's options.

Type Map Limitations

The aggregate (when not using a cursor), distinct, and findAndModify helpers do not support a `typeMap` option due to a driver limitation. The `aggregate()`, `distinct()`, `findOneAndReplace()`, `findOneAndUpdate()`, and `findOneAndDelete()` methods return BSON documents as *stdClass* objects and BSON arrays as arrays.

Methods

- MongoDB\Collection::__construct()
- MongoDB\Collection::aggregate()
- MongoDB\Collection::bulkWrite()
- MongoDB\Collection::count()
- MongoDB\Collection::createIndex()
- MongoDB\Collection::createIndexes()
- MongoDB\Collection::deleteMany()
- MongoDB\Collection::deleteOne()
- MongoDB\Collection::distinct()
- MongoDB\Collection::drop()
- MongoDB\Collection::dropIndex()
- MongoDB\Collection::dropIndexes()
- MongoDB\Collection::find()
- MongoDB\Collection::findOne()
- MongoDB\Collection::findOneAndDelete()
- MongoDB\Collection::findOneAndReplace()
- MongoDB\Collection::findOneAndUpdate()
- MongoDB\Collection::getCollectionName()
- MongoDB\Collection::getDatabaseName()
- MongoDB\Collection::getManager()
- MongoDB\Collection::getNamespace()
- MongoDB\Collection::getReadConcern()
- MongoDB\Collection::getReadPreference()
- MongoDB\Collection::getTypeMap()
- MongoDB\Collection::getWriteConcern()
- MongoDB\Collection::insertMany()
- MongoDB\Collection::insertOne()
- MongoDB\Collection::listIndexes()
- MongoDB\Collection::mapReduce()
- MongoDB\Collection::replaceOne()
- MongoDB\Collection::updateMany()
- MongoDB\Collection::updateOne()
- MongoDB\Collection::withOptions()

Write Result Classes

MongoDB\BulkWriteResult

Definition

`MongoDB\BulkWriteResult`

This class contains information about an executed bulk write operation. It encapsulates a MongoDB\Driver\WriteResult object and is returned from `MongoDB\Collection::bulkWrite()`.

Methods

- MongoDB\BulkWriteResult::getDeletedCount()
- MongoDB\BulkWriteResult::getInsertedCount()
- MongoDB\BulkWriteResult::getInsertedIds()
- MongoDB\BulkWriteResult::getMatchedCount()
- MongoDB\BulkWriteResult::getModifiedCount()
- MongoDB\BulkWriteResult::getUpsertedCount()
- MongoDB\BulkWriteResult::getUpsertedIds()
- MongoDB\BulkWriteResult::isAcknowledged()

MongoDB\DeleteResult

Definition

`MongoDB\DeleteResult`

This class contains information about an executed delete operation. It encapsulates a MongoDB\Driver\WriteResult object and is returned from `MongoDB\Collection::deleteMany()` or `MongoDB\Collection::deleteOne()`.

Methods

- MongoDB\DeleteResult::getDeletedCount()
- MongoDB\DeleteResult::isAcknowledged()

MongoDB\InsertManyResult

Definition

`MongoDB\InsertManyResult`

This class contains information about an executed bulk insert operation. It encapsulates a MongoDB\Driver\WriteResult object and is returned from `MongoDB\Collection::insertMany()`.

Methods

- MongoDB\InsertManyResult::getInsertedCount()
- MongoDB\InsertManyResult::getInsertedIds()
- MongoDB\InsertManyResult::isAcknowledged()

MongoDB\InsertOneResult

Definition

`MongoDB\InsertOneResult`

This class contains information about an executed insert operation. It encapsulates a MongoDB\Driver\WriteResult object and is returned from `MongoDB\Collection::insertOne()`.

Methods

- MongoDB\InsertOneResult::getInsertedCount()
- MongoDB\InsertOneResult::getInsertedId()
- MongoDB\InsertOneResult::isAcknowledged()

MongoDB\UpdateResult

Definition

`MongoDB\UpdateResult`

This class contains information about an executed update or replace operation. It encapsulates a MongoDB\Driver\WriteResult object and is returned from `MongoDB\Collection::replaceOne()`, `MongoDB\Collection::updateMany()`, or `MongoDB\Collection::updateOne()`.

Methods

- MongoDB\UpdateResult::getMatchedCount()
- MongoDB\UpdateResult::getModifiedCount()
- MongoDB\UpdateResult::getUpsertedCount()
- MongoDB\UpdateResult::getUpsertedId()

- MongoDB\UpdateResult::isAcknowledged()

MongoDB\Collection::find()

Definition

`MongoDB\Collection::find`

Finds documents matching the query.

```
function find($filter = [], array $options = []): MongoDB\Driver\Cursor
```

This method has the following parameters:

Parameter	Type	Description
`$filter`	array|object	Optional. The filter criteria that specifies the documents to query.
`$options`	array	Optional. An array specifying the desired options.

The `$options` parameter supports the following options:

Option	Type	Description
`projection`	array|object	Optional. The projection specification to determine which fields to include in the returned documents. See Project Fields to Return from Query and Projection Operators in the MongoDB manual.
`sort`	array|object	Optional. The sort specification for the ordering of the results.
`skip`	Integer	Optional. Number of documents to skip. Defaults to `0`.
`limit`	integer	Optional. The maximum number of documents to return. If unspecified, then defaults to no limit. A limit of `0` is equivalent to setting no limit. A negative limit is similar to a positive limit but closes the cursor after returning a single batch of results. As such, with a negative limit, if the limited result set does not fit into a single batch, the number of documents received will be less than the specified limit. By passing a negative limit, the client indicates to the server that it will not ask for a subsequent batch via getMore.
`batchSize`	Integer	Optional. The number of documents to return in the first batch. Defaults to `101`. A batchSize of `0` means that the cursor will be

		established, but no documents will be returned in the first batch. Unlike the previous wire protocol version, a batchSize of `1` for the `find` command does not close the cursor.
`collation`	array|object	Optional. Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks. When specifying collation, the `locale` field is mandatory; all other collation fields are optional. For descriptions of the fields, see Collation Document. If the collation is unspecified but the collection has a default collation, the operation uses the collation specified for the collection. If no collation is specified for the collection or for the operation, MongoDB uses the simple binary comparison used in prior versions for string comparisons. This option is available in MongoDB 3.4+ and will result in an exception at execution time if specified for an older server version.
`comment`	String	Optional. A comment to attach to the query to help interpret and trace query `profile` data.
`cursorType`	Integer	Optional. Indicates the type of cursor to use. `cursorType` supports the following values: • `MongoDB\Operation\Find::NON_TAILABLE` (*default*) • `MongoDB\Operation\Find::TAILABLE`
`hint`	string|array|object	Optional. The index to use. Specify either the index name as a string or the index key pattern as a document. If specified, then the query system will only consider plans using the hinted index. New in version 1.2.
`maxAwaitTimeMS`	integer	Optional. Positive integer denoting the time limit in milliseconds for the server to block a getMore operation if no data is available. This option should only be used if cursorType is TAILABLE_AWAIT. New in version 1.2.
`maxTimeMS`	Integer	Optional. The cumulative time limit in milliseconds for processing operations on the cursor. MongoDB aborts the operation at the earliest following interrupt point.
`readConcern`	MongoDB\Driver\ReadConcern	Optional. Read concern to use for the operation. Defaults to the collection's read concern. This is not supported for server versions prior to 3.2 and will result in an exception at execution time if used.
`readPreference`	MongoDB\Driver\ReadPrefe	Optional. Read preference to use for the operation. Defaults to the collection's read preference.

	rence	
`max`	array|object	Optional. The exclusive upper bound for a specific index. New in version 1.2.
`maxScan`	Integer	Optional. Maximum number of documents or index keys to scan when executing the query. New in version 1.2.
`min`	array|object	Optional. The inclusive lower bound for a specific index. New in version 1.2.
`oplogReplay`	Boolean	Optional. Internal use for replica sets. To use `oplogReplay`, you must include the following condition in the filter: `{ ts: { $gte: <timestamp> } }` The MongoDB\BSON\Timestamp class reference describes how to represent MongoDB's BSON timestamp type with PHP.
`noCursorTimeout`	Boolean	Optional. Prevents the server from timing out idle cursors after an inactivity period (10 minutes).
`returnKey`	Boolean	Optional. If true, returns only the index keys in the resulting documents. New in version 1.2.
`showRecordId`	Boolean	Optional. Determines whether to return the record identifier for each document. If true, adds a field $recordId to the returned documents. New in version 1.2.
`Snapshot`	Boolean	Optional. Prevents the cursor from returning a document more than once because of an intervening write operation. New in version 1.2.
`allowPartialResults`	Boolean	Optional. For queries against a sharded collection, returns partial results from the **mongos** if some shards are unavailable instead of throwing an error.
`typeMap`	Array	Optional. The type map to apply to cursors, which determines how BSON documents are converted to PHP values. Defaults to the collection's type map.
`Modifiers`	array|object	Optional. Meta operators that modify the output or behavior of a query. Use of these operators is deprecated in favor of named options.

Return Values

A MongoDB\Driver\Cursor object.

Errors/Exceptions

`MongoDB\Exception\UnsupportedException` if options are used and not supported by the selected server (e.g. `collation`, `readConcern`, `writeConcern`).

`MongoDB\Exception\InvalidArgumentException` for errors related to the parsing of parameters or options.

MongoDB\Driver\Exception\RuntimeException for other errors at the driver level (e.g. connection errors).

Behavior

When evaluating query criteria, MongoDB compares types and values according to its own comparison rules for BSON types, which differs from PHP's comparison and type juggling rules. When matching a special BSON type the query criteria should use the respective BSON class in the driver (e.g. use MongoDB\BSON\ObjectId to match an ObjectId).

Examples

The following example finds restaurants based on the `cuisine` and `borough` fields and uses a projection to limit the fields that are returned. It also limits the results to 5 documents.

```
$collection = (new MongoDB\Client)->test->restaurants;

$cursor = $collection->find(
    [
        'cuisine' => 'Italian',
        'borough' => 'Manhattan',
    ],
    [
        'limit' => 5,
        'projection' => [
            'name' => 1,
            'borough' => 1,
            'cuisine' => 1,
        ],
    ]
);

foreach ($cursor as $restaurant) {
   var_dump($restaurant);
};
```

The output would then resemble:

```
object(MongoDB\Model\BSONDocument)#10 (1) {
  ["storage":"ArrayObject":private]=>
  array(4) {
    ["_id"]=>
    object(MongoDB\BSON\ObjectId)#8 (1) {
      ["oid"]=>
      string(24) "576023c6b02fa9281da3f983"
    }
    ["borough"]=>
    string(9) "Manhattan"
    ["cuisine"]=>
    string(7) "Italian"
    ["name"]=>
    string(23) "Isle Of Capri Resturant"
  }
}
object(MongoDB\Model\BSONDocument)#13 (1) {
  ["storage":"ArrayObject":private]=>
  array(4) {
    ["_id"]=>
    object(MongoDB\BSON\ObjectId)#12 (1) {
      ["oid"]=>
      string(24) "576023c6b02fa9281da3f98d"
    }
    ["borough"]=>
    string(9) "Manhattan"
    ["cuisine"]=>
    string(7) "Italian"
    ["name"]=>
    string(18) "Marchis Restaurant"
  }
}
object(MongoDB\Model\BSONDocument)#8 (1) {
  ["storage":"ArrayObject":private]=>
  array(4) {
    ["_id"]=>
    object(MongoDB\BSON\ObjectId)#10 (1) {
      ["oid"]=>
      string(24) "576023c6b02fa9281da3f99b"
    }
    ["borough"]=>
    string(9) "Manhattan"
    ["cuisine"]=>
    string(7) "Italian"
    ["name"]=>
    string(19) "Forlinis Restaurant"
  }
}
object(MongoDB\Model\BSONDocument)#12 (1) {
  ["storage":"ArrayObject":private]=>
  array(4) {
    ["_id"]=>
    object(MongoDB\BSON\ObjectId)#13 (1) {
      ["oid"]=>
      string(24) "576023c6b02fa9281da3f9a8"
    }
    ["borough"]=>
    string(9) "Manhattan"
    ["cuisine"]=>
    string(7) "Italian"
    ["name"]=>
    string(22) "Angelo Of Mulberry St."
  }
}
object(MongoDB\Model\BSONDocument)#10 (1) {
  ["storage":"ArrayObject":private]=>
  array(4) {
    ["_id"]=>
    object(MongoDB\BSON\ObjectId)#8 (1) {
      ["oid"]=>
      string(24) "576023c6b02fa9281da3f9b4"
    }
    ["borough"]=>
    string(9) "Manhattan"
    ["cuisine"]=>
    string(7) "Italian"
    ["name"]=>
    string(16) "V & T Restaurant"
  }
}
```