PHP and Non-Relational Databases

Reza Salehi
CLOUD CONSULTANT

@zaalion linkedin.com/in/rezasalehi2008

Overview

Quick reminder

A few NoSQL databases to use with PHP

Configuring PHP to use MongoDB

Performing CRUD

Demo: PHP and MongoDB

PHP and NoSQL Databases

PHP can work with several database engines...

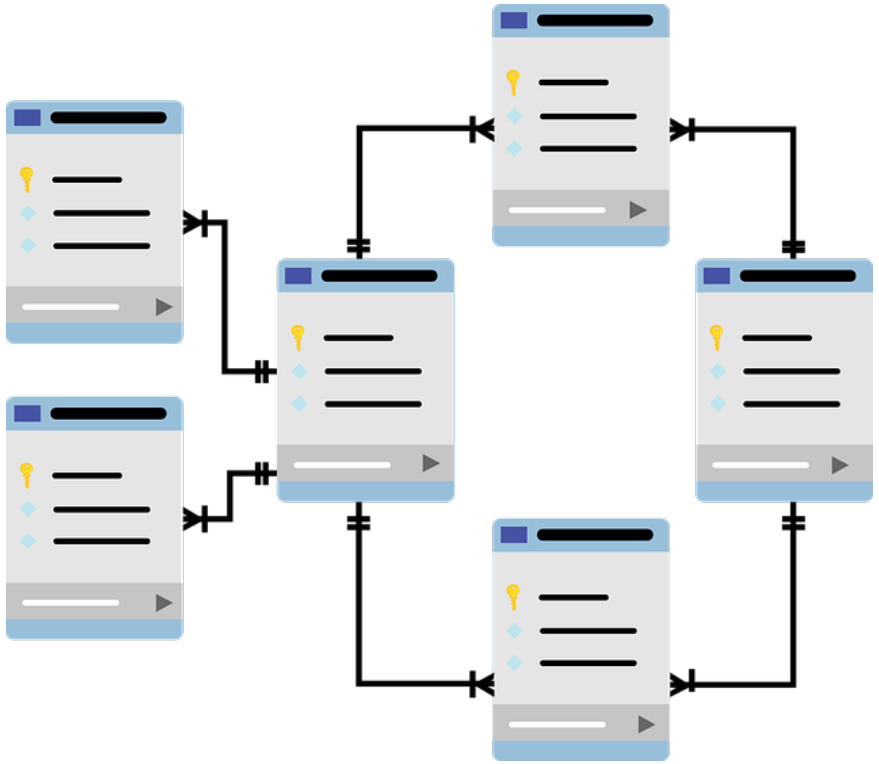
Database to Discuss

Relational

Data is structured into multiple related tables with fixed columns.

NoSQL

Data is structured in means other than tabular relations. Many offerings out there!



```
{
    "person": {
        "name": "John Smith",
        "address": {
            "streetAddress": "#1, 1 sample street",
            "city" : "Boston",
            "country": "USA"
        }
    }
}
```

NoSQL Databases

Flat files	Document	Key-value pair
Graph	Cache	Wide Column

PHP Supports a Few NoSQL Databases

Change language: English

Edit Report a Bug

MongoDB driver

Unlike the mongo extension, this extension is developed atop the » libmongoc and » libbson libraries. It provides a minimal API for core driver functionality: commands, queries, writes, connection management, and BSON serialization.

Userland PHP libraries that depend on this extension may provide higher level APIs, such as query builders, individual command helper methods, and GridFS. Application developers should consider using this extension in conjunction with the » MongoDB PHP library, which implements the same higher level APIs found in MongoDB drivers for other languages. This separation of concerns allows the driver to focus on essential features for which an extension implementation is paramount for performance.

- Installing/Configuring
  - Requirements
  - Installation
  - Runtime Configuration
  - Predefined Constants
- Tutorials
  - Using the PHP Library for MongoDB (PHPLIB)
  - Application Performance Monitoring (APM)
- Driver Architecture and Internals — Explains the driver architecture, and special features
  - Architecture — Architecture Overview
  - Connections — Connection handling and persistence

Vendor Specific Database Extensions

- CUBRID
- DB++
- dBase
- filePro
- Firebird/InterBase
- FrontBase
- IBM DB2
- Informix
- Ingres
- MaxDB
- Mongo
- » MongoDB
- mSQL
- Mssql
- MySQL
- OCI8
- Paradox
- PostgreSQL
- SQLite
- SQLite3
- SQLSRV
- Sybase
- tokyo_tyrant

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.PHP.html

Amazon DynamoDB
Developer Guide

- What Is Amazon DynamoDB?
- Setting Up DynamoDB
- Accessing DynamoDB
- Getting Started with DynamoDB
- Getting Started with the AWS SDKs
  - Java and DynamoDB
  - JavaScript and DynamoDB
  - Node.js and DynamoDB
  - .NET and DynamoDB
  - PHP and DynamoDB
    - Step 1: Create a Table
    - Step 2: Load Sample Data
    - Step 3: Create, Read, Update, and Delete an Item
    - Step 4: Query and Scan the Data
    - Step 5: (Optional) Delete the Table
    - Summary
  - Python and DynamoDB
  - Ruby and DynamoDB
- Programming with DynamoDB
- Working with DynamoDB

PHP and DynamoDB

PDF | Kindle | RSS

In this tutorial, you use the AWS SDK for PHP to write simple programs to perform the following Amazon DynamoDB operations:

- Create a table called `Movies` and load sample data in JSON format.
- Perform create, read, update, and delete operations on the table.
- Run simple queries.

As you work through this tutorial, you can refer to the AWS SDK for PHP Developer Guide. The Amazon DynamoDB section in the *AWS SDK for PHP API Reference* describes the parameters and results for DynamoDB operations.

Tutorial Prerequisites

- Download and run DynamoDB on your computer. For more information, see Setting Up DynamoDB Local (Downloadable Version).

  Note

  You use the downloadable version of DynamoDB in this tutorial. For information about how to run the same code against the DynamoDB service, see the Summary.

- Set up an AWS access key to use the AWS SDKs. For more information, see Setting Up DynamoDB (Web Service).
- Set up the AWS SDK for PHP:
  - Install PHP.
  - Install the SDK for PHP.

  For more information, see Getting Started in the *AWS SDK for PHP Getting Started Guide*.

https://docs.microsoft.com/en-us/azure/cosmos-db/create-graph-php

Filter by title

- Cassandra, MongoDB, and other APIs
  - Cassandra API
  - MongoDB API
  - Gremlin API
    - Overview
    - Quickstarts
      - Gremlin console
      - .NET
      - Java
      - Node.js
      - Python
      - PHP
    - Tutorials
    - Concepts
    - How-to guides
    - Reference
  - Table API
  - Etcd API
- How-to guides
- References for SQL API SDKs
- Resources

Download PDF

Quickstart: Create a graph database in Azure Cosmos DB using PHP and the Azure portal

01/05/2019 • 9 minutes to read • +1

PHP

This quickstart shows how to use PHP and the Azure Cosmos DB Gremlin API to build a console app by cloning an example from GitHub. This quickstart also walks you through the creation of an Azure Cosmos DB account by using the web-based Azure portal.

Azure Cosmos DB is Microsoft's globally distributed multi-model database service. You can quickly create and query document, table, key-value, and graph databases, all of which benefit from the global distribution and horizontal scale capabilities at the core of Azure Cosmos DB.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin. Alternatively, you can Try Azure Cosmos DB for free without an Azure subscription, free of charge and commitments.

In addition:

- PHP 5.6 or newer
- Composer

Create a database account

Before you can create a graph database, you need to create a Gremlin (Graph) database account with Azure Cosmos DB.

Is this page helpful?

Yes No

In this article

- Prerequisites
- Create a database account
- Add a graph
- Clone the sample application
- Review the code
- Update your connection information
- Run the console app
- Review and add sample data
- Review SLAs in the Azure portal
- Clean up resources
- Next steps


mongoDB®

PHP and MongoDB

Change language: English

Edit Report a Bug

MongoDB driver

Unlike the mongo extension, this extension is developed atop the » libmongoc and » libbson libraries. It provides a minimal API for core driver functionality: commands, queries, writes, connection management, and BSON serialization.

Userland PHP libraries that depend on this extension may provide higher level APIs, such as query builders, individual command helper methods, and GridFS. Application developers should consider using this extension in conjunction with the » MongoDB PHP library, which implements the same higher level APIs found in MongoDB drivers for other languages. This separation of concerns allows the driver to focus on essential features for which an extension implementation is paramount for performance.

- Installing/Configuring
  - Requirements
  - Installation
  - Runtime Configuration
  - Predefined Constants
- Tutorials
  - Using the PHP Library for MongoDB (PHPLIB)
  - Application Performance Monitoring (APM)
- Driver Architecture and Internals — Explains the driver architecture, and special features
  - Architecture — Architecture Overview
  - Connections — Connection handling and persistence

Vendor Specific Database Extensions

- CUBRID
- DB++
- dBase
- filePro
- Firebird/InterBase
- FrontBase
- IBM DB2
- Informix
- Ingres
- MaxDB
- Mongo
- » MongoDB
- mSQL
- Mssql
- MySQL
- OCI8
- Paradox
- PostgreSQL
- SQLite
- SQLite3
- SQLSRV
- Sybase
- tokyo_tyrant

https://docs.mongodb.com/drivers/php

Close ×

MONGODB DRIVERS

- C Driver
- C++ Driver
- C# and .NET Driver
- Go Driver
- Java Drivers
- Node.js Driver
- Perl Driver
- **PHP Driver**
  - PHP Libraries, Frameworks, and Tools
- Python Drivers
- Ruby Driver
- Ruby Mongoid ODM
- Scala Driver
- Swift Driver
- Rust Driver
- About Compatibility Tables
- Use Cases

MongoDB PHP Driver

On this page

- Introduction
- Installation
- Connect to MongoDB Atlas
- Compatibility
- See Also

Introduction

The PHP driver consists of two components, the MongoDB extension and library.

The extension provides a low-level API and mainly serves to integrate libmongoc and libbson with PHP.

While it is possible to use the extension alone, users are strongly encouraged to use the extension and library together. The library provides a high-level API consistent with other MongoDB language drivers.

- Tutorials
- Extension Architecture and Internals
- Documentation
  - Library
  - Extension
- Changelog
  - Library
  - Extension
- Source Code
  - Library
  - Extension

Installation

First, make sure you have a recent version of PHP installed on your system. See the official PHP manual for download and installation instructions.

Change language: English
Edit Report a Bug

Installing the MongoDB PHP Driver with PECL

Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here:
» https://pecl.php.net/package/mongodb

Linux, Unix, and macOS users may run the following command to install the driver:

```
$ sudo pecl install mongodb
```

If your system has multiple version of PHP installed (e.g. macOS default, Homebrew, » XAMPP), note that that each version of PHP has its own pecl command and `php.ini` file.

Installing the driver via PECL will use bundled versions of » libbson and » libmongoc and attempt to automatically configure them.

Note: If the build process fails to find an SSL library, check that the development packages (e.g. *libssl-dev*) and » pkg-config are both installed. If that does not resolve the problem, consider using the manual installation process.

Finally, add the following line to your `php.ini` file:

Installation
» Installing the MongoDB PHP Driver with PECL
Installing the MongoDB PHP Driver on macOS with Homebrew
Installing the MongoDB PHP Driver on Windows
Building the MongoDB PHP Driver from source

Change language: English

Edit Report a Bug

Installing the MongoDB PHP Driver on Windows

Precompiled binaries for each release are available from » PECL for a variety of combinations of versions, thread safety, and VC libraries. Extract the archive and put `php_mongodb.dll` in your PHP extension directory ("ext" by default).

Add the following line to your `php.ini` file:

```
extension=php_mongodb.dll
```

Note: Additional DLL dependencies for Windows Users

In order for this extension to work, there are DLL files that must be available to the Windows system PATH. For information on how to do this, see the FAQ entitled "How do I add my PHP directory to the PATH on Windows". Although copying DLL files from the PHP folder into the Windows system directory also works (because the system directory is by default in the system's PATH), this is not recommended. *This extension requires the following files to be in the* PATH*:* `libsasl.dll`

User Contributed Notes 2 notes

add a note

▲ 0 ▼ luchontandil3 at gmail dot com 3 months ago

Installation

Installing the MongoDB PHP Driver with PECL

Installing the MongoDB PHP Driver on macOS with Homebrew

» Installing the MongoDB PHP Driver on Windows

Building the MongoDB PHP Driver from source

https://www.php.net/manual/en/class.mongoclient.php

The MongoClient class

(PECL mongo >=1.3.0)

Warning This extension that defines this class is d recated. stead, the MongoDB extensi should be us Alternatives t his class include:

- MongoDB\Driver\Manager

Introduction

A connection manager for PHP and MongoDB.

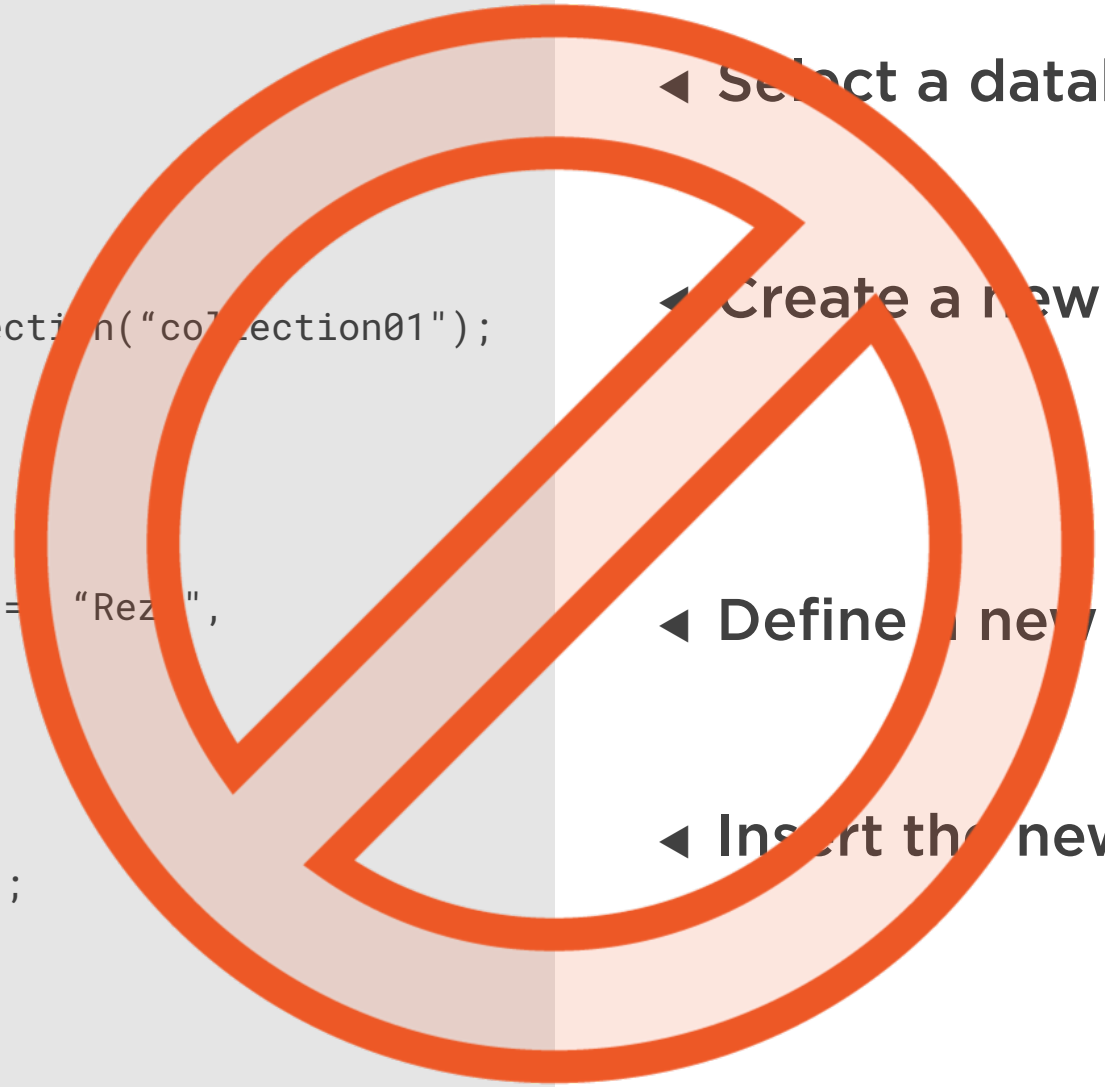
This class is used to create and manage connections. A typical is:

Example #1 MongoClient basic usage

```
<?php

$m = new MongoClient(); // connect
```

Core Classes
» MongoClient
MongoDB
MongoCollection
MongoCursor
MongoCursorInterface
MongoCommandCursor

```php
<?php

$mClient = new MongoClient(); // connect

$db = $mClient->sampleDB;


// Create a collection

$collection = $db->createCollection("collection01");



$document = array("firstname" => "Reza",

      "lastname" => "Salehi");


$collection->insert($document);

?>
```



- Create MongoDB client
- Select a database
- Create a new collection
- Define a new document
- Insert the new document

```php
<?php

// connect

$manager = new MongoDB\Driver\Manager("connection
string here");



// Create a bulk-write object

$bulk = new MongoDB\Driver\BulkWrite;




$newPerson = array("Firstname" => $firstname,…)



$id = $bulk->insert($newPerson);

$result = $manager->
executeBulkWrite('PersonalDB.Person', $bulk);

?>
```

- Create MongoDB manager client
- Create a bulk-write object
- Define a new document
- Insert the new document

```
$manager = new MongoDB\Driver\Manager(“"mongodb+srv://..”)
```

Create a Manager object

```php
$bulk = new MongoDB\Driver\BulkWrite;

$id = $bulk->insert($newPerson);
```

Inserting a Document

```
$query = new MongoDB\Driver\Query([]);
$cursor = $manager->executeQuery("PersonalDB.Person", $query);
```

Select All Documents

```php
$bulk = new MongoDB\Driver\BulkWrite;


$bulk->update(['id' => 112],
    ['$set' => ["Firstname" => $firstname]],
    ['multi' => false, 'upsert' => false]
```

Update a Document

```
$bulk = new MongoDB\Driver\BulkWrite;
$bulk->delete(['Pid'=>intval($Pid)], ['limit' => 1]);
```

Delete a Document

```
$result = $manager->executeBulkWrite('PersonalDB.Person',
$bulk);
```

Run ExecuteBulkWrite

Change language: English

Edit Report a Bug

The MongoDB\Driver\Manager class

(mongodb >=1.0.0)

Introduction

The **MongoDB\Driver\Manager** is the main entry point to the extension. It is responsible for maintaining connections to MongoDB (be it standalone server, replica set, or sharded cluster).

No connection to MongoDB is made upon instantiating the Manager. This means the **MongoDB\Driver\Manager** can always be constructed, even though one or more MongoDB servers are down.

Any write or query can throw connection exceptions as connections are created lazily. A MongoDB server may also become unavailable during the life time of the script. It is therefore important that all actions on the Manager to be wrapped in try/catch statements.

Class synopsis

```
final MongoDB\Driver\Manager {

   /* Methods */
   final public __construct ([ string $uri = "mongodb://127.0.0.1/" [, array $uriOptions = array() [, array
   $driverOptions = array() ]]] )
   final public createClientEncryption ( array $options ) : MongoDB\Driver\ClientEncryption
```

MongoDB\Driver
» MongoDB\Driver\Manager
MongoDB\Driver\Command
MongoDB\Driver\Query
MongoDB\Driver\BulkWrite
MongoDB\Driver\Session
MongoDB\Driver\ClientEncryption
MongoDB\Driver\WriteConcern
MongoDB\Driver\ReadPreference
MongoDB\Driver\ReadConcern
MongoDB\Driver\Cursor
MongoDB\Driver\CursorId
MongoDB\Driver\CursorInterface
MongoDB\Driver\Server
MongoDB\Driver\WriteConcernError
MongoDB\Driver\WriteError
MongoDB\Driver\WriteResult

Demo

Setting up MongoDB

- MongoDB Atlas

Demo

Working with MongoDB in PHP

Summary

A few NoSQL databases to use with PHP

Configuring PHP to use MongoDB

Performing CRUD

Demo: PHP and MongoDB