

---

# php-dynamodb Documentation

*Release stable*

Feb 08, 2022



---

## Contents

---

<b>1</b>	<b>Quickstart</b>	<b>3</b>
<b>2</b>	<b>User Guide</b>	<b>5</b>
2.1	Overview . . . . .	5
2.1.1	Installation . . . . .	5
2.1.2	Running DynamoDB Locally . . . . .	5
2.1.3	Contributing . . . . .	5
2.2	Executing Operations . . . . .	6
2.2.1	The DynamoDB Client Factory . . . . .	6
2.2.2	The Marshaler Factory . . . . .	6
2.2.3	The DynamoDB Adapter . . . . .	6
2.2.4	The Operation Factory . . . . .	7
2.3	Working with Tables . . . . .	7
2.3.1	Listing Tables . . . . .	7
2.3.2	Creating Tables . . . . .	7
2.3.3	Deleting Tables . . . . .	8
2.4	Working with Items . . . . .	8



**php-dynamodb** is a PHP library that can be used to interact with Amazon DynamoDB. It provides a layer of abstraction between your code and the DynamoDB-related classes made available by the [AWS SDK for PHP](#).



# CHAPTER 1

## Quickstart

```
1  <?php declare(strict_types=1);
2
3  require dirname(__DIR__) . '/vendor/autoload.php';
4
5  use Guillermoandrae\DynamoDb\Constant\AttributeTypes;
6  use Guillermoandrae\DynamoDb\Constant\KeyTypes;
7  use Guillermoandrae\DynamoDb\DynamoDbAdapter;
8
9  // create a new adapter
10 $adapter = new DynamoDbAdapter();
11
12 try {
13     $tableName = 'myTable';
14
15     // create a table
16     $adapter->useTable($tableName)->createTable([
17         'year' => [AttributeTypes::NUMBER, KeyTypes::HASH],
18         'title' => [AttributeTypes::STRING, KeyTypes::RANGE],
19     ]);
20
21     // add an item to the table
22     $adapter->useTable($tableName)->insert([
23         'year' => 2015,
24         'title' => 'The Big New Movie',
25         'info' => [
26             'plot' => 'Nothing happens at all',
27             'rating' => 0,
28         ],
29     ]);
30
31     // fetch an item from the table
32     $item = $adapter->useTable($tableName)->find([
33         'year' => 2015,
34         'title' => 'The Big New Movie'
```

(continues on next page)

(continued from previous page)

```
35     });
36
37     printf('Added item: %s - %s' . PHP_EOL, $item['year'], $item['title']);
38
39     print_r($item);
40
41     // delete the table
42     $adapter->useTable($tableName)->deleteTable();
43
44 } catch (\Exception $ex) {
45     die($ex->getMessage() . PHP_EOL);
46 }
```



## 2.1 Overview

**php-dynamodb** is a PHP library that can be used to interact with [Amazon DynamoDB](#). It provides a layer of abstraction between your code and the DynamoDB-related classes made available by the [AWS SDK for PHP](#).

### 2.1.1 Installation

The recommended way to install this library is through [Composer](#):

```
composer install guillermoandreae/php-dynamodb
```

### 2.1.2 Running DynamoDB Locally

To aid in your development, you can run the following commands to manage DynamoDB locally:

```
composer install-db # downloads and installs DynamoDB locally
composer start-db # starts DynamoDB locally
composer stop-db # stops DynamoDB locally
composer restart-db # calls stop-db then start-db
```

### 2.1.3 Contributing

To find out how to contribute to this project, please refer to the [CONTRIBUTING](#) file in the project's GitHub repository.

## 2.2 Executing Operations

You can easily execute DynamoDB operations using a straightforward API.

### 2.2.1 The DynamoDB Client Factory

To create an instance of the AWS `DynamoDbClient` class, you can use `php-dynamodb`'s `DynamoDbClientFactory`. By default, it will use local credentials to create an instance of the client, but you can provide your own options to create a client that can connect to a DynamoDB table in your AWS account. You can use the `DynamoDbClientFactory` to create clients that can be passed to the DynamoDB adapter.

#### Example

```
use Guillermoandreae\DynamoDb\Factory\DynamoDbClientFactory;

$client = DynamoDbClientFactory::factory();
```

or

#### Example

```
use Guillermoandreae\DynamoDb\Factory\DynamoDbClientFactory;

$client = DynamoDbClientFactory::factory([
    'region' => '<your region>',
    'version' => 'latest',
    'endpoint' => '<your endpoint>',
    'credentials' => [
        'key' => '<your key>',
        'secret' => '<your secret>',
    ],
]);
```

### 2.2.2 The Marshaler Factory

A `Marshaler` object is needed to process requests and results. The `MarshalerFactory` creates instances of the AWS `Marshaler` that can be passed to the DynamoDB adapter.

#### Example

```
use Guillermoandreae\DynamoDb\Factory\MarshalerFactory;

$marshaller = MarshalerFactory::factory()
```

### 2.2.3 The DynamoDB Adapter

#### Example

```
use Guillermoandreae\DynamoDb\DynamoDbAdapter;

$adapter = new DynamoDbAdapter();
```

## 2.2.4 The Operation Factory

### Example

```
use Guillermoandrae\DynamoDb\Factory\DynamoDbClientFactory;
use Guillermoandrae\DynamoDb\Factory\MarshallerFactory;
use Guillermoandrae\DynamoDb\Factory\OperationFactory;

OperationFactory::registerClient(DynamoDbClientFactory::factory());
OperationFactory::registerMarshaller(MarshallerFactory::factory());
$operation = OperationFactory::factory('list-tables', 'myTable');
$operation->execute();
```

## 2.3 Working with Tables

The following table operations are supported by **php-dynamodb**.

### 2.3.1 Listing Tables

To see a list of the tables that exist in your database, use the `listTables()` method. It will return an indexed array that stores the names of the existing tables.

#### Example

```
$tables = $adapter->listTables();
foreach ($tables as $table) {
    echo $table . PHP_EOL;
}
```

### 2.3.2 Creating Tables

Simple table creation can be accomplished using the adapter's `createTable()` method. The method takes the table name as an optional parameter (you can, instead, use the `useTable()` method and pass it the table name) and an optional array that specifies the table's key schema.

#### Example

```
$result = $adapter->useTable('myTable')->createTable();
```

More complex table creation can be accomplished using the `CreateTableOperation` class, an instance of which can be created using the `OperationFactory`.

#### Example

```
use Guillermoandrae\DynamoDb\Constant\AttributeTypes;
use Guillermoandrae\DynamoDb\Constant\KeyTypes;
use Guillermoandrae\DynamoDb\DynamoDbAdapter;

$operation = new CreateTableOperation('myTable', [
    'name' => [AttributeTypes::STRING, KeyTypes::HASH],
    'year' => [AttributeTypes::NUMBER, KeyTypes::RANGE],
]);
```

(continues on next page)

(continued from previous page)

```
$operation->setReadCapacityUnits(10);  
$result = $operation->execute();
```

---

**Note:** By default, php-dynamodb will use 5 read capacity units and 5 write capacity units when creating tables.

---

### 2.3.3 Deleting Tables

To delete a table in your database, use the `deleteTable()` method. It will return a boolean value that indicates whether or not the deletion was successful.

**Example**

```
$result = $adapter->deleteTable('myTable');
```

## 2.4 Working with Items

Coming soon!