php-dynamodb Documentation

Release latest

Feb 08, 2022

Contents

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

```php
<?php declare(strict_types=1);

require dirname(__DIR__) . '/vendor/autoload.php';

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

// create a new adapter
$adapter = new DynamoDbAdapter();

try {
    $tableName = 'myTable';

    // create a table
    $adapter->useTable($tableName)->createTable([
        'year' => [AttributeTypes::NUMBER, KeyTypes::HASH],
        'title' => [AttributeTypes::STRING, KeyTypes::RANGE],
    ]);

    // add an item to the table
    $adapter->useTable($tableName)->insert([
        'year' => 2015,
        'title' => 'The Big New Movie',
        'info' => [
            'plot' => 'Nothing happens at all',
            'rating' => 0,
        ],
    ]);

    // fetch an item from the table
    $item = $adapter->useTable($tableName)->find([
        'year' => 2015,
        'title' => 'The Big New Movie'
```

(continues on next page)

(continued from previous page)

```
    ]);

    printf('Added item: %s - %s' . PHP_EOL, $item['year'], $item['title']);

    print_r($item);

    // delete the table
    $adapter->useTable($tableName)->deleteTable();

} catch (\Exception $ex) {
    die($ex->getMessage() . PHP_EOL);
}
```

CHAPTER 2

User Guide

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 guillermoandrae/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 Guillermoandrae\DynamoDb\Factory\DynamoDbClientFactory;

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

or

Example

```
use Guillermoandrae\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 Guillermoandrae\DynamoDb\Factory\MarshalerFactory;

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

2.2.3 The DynamoDB Adapter

Example

```
use Guillermoandrae\DynamoDb\DynamoDbAdapter;

$adapter = new DynamoDbAdapter();
```

2.2.4 The Operation Factory

Example

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

OperationFactory::registerClient(DynamoDbClientFactory::factory());
OperationFactory::registerMarshaler(MarshalerFactory::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!