Baleen Migrations Documentation

Release 0.7.0

Gabriel Somoza <gabriel@strategery.io>

April 25, 2016

Contents

Baleen Migrations is a project that seeks to abstract the domain of performing migrations of any kind into a single independent package. Its goal is to excel at one single task: provide an intuitive, well-tested framework to migrate "something" from point A to point B (or vice-versa if going down).

In other words, we take care of WHICH migrations are run and the mechanism to run them. Everything else is the responsibility of the implementation:

- WHAT is going to be migrated? It could be a database, images, documents, etc.
- HOW its going to be migrated? You can wrap each migration into DB transactions. Or not, its up to you.
- What to do when a migration fails? We'll let you know WHEN it happens, but its up to you to decide what to do (e.g. cancel the transaction / re-throw the exception, etc).

Contents:

CHAPTER 1

Getting Started

Before you get started its important you understand what this particular library is about. Make sure you read the introduction and the following architecture overview.

1.1 Overview

The Baleen projects aims to make migrations easier for anyone. All projects under the "Baleen umbrella" are meant to be related to migrations in one way or another.

Baleen Migrations is the core of the Baleen project. It provides a domain model on top of which other migration libraries can run their *migration scripts*.

In other words: Baleen Migrations alone is NOT supposed to be an end-user solution. Its meant to be used by other repositories such as Doctrine Migrations, and it is those repositories that would become the end-user libraries that framework users consume.

So if you came to this repository to find a new migrations solution then you're not quite in the right place yet. Instead, please refer to the implementations list of libraries - we hope there you'll find what you're looking for.

1.2 Installation

Installing Baleen Migrations with Composer is very easy:

```
composer install baleen/migrations
```

Then make sure you require the Composer autoloader:

```
<?php
require __DIR__ . '/vendor/autoload.php';

// do Baleen stuff here.
```

You're all set!

Refer to the Examples to see Baleen Migrations in action.

CHAPTER 2

Domain Model

2.1 Glossary

Baleen Migrations has been carefully engineered with the goal of satisfying as many types migration as possible. Whether its a database migration, image transformations or even source code, this module aims to help in the process of pragmatically transform virtually anything from one state to the other, taking it through a series of pre-defined steps (migrations).

The purpose of this document is to standardise the use of certain terms throughout the context of this project, as well as to provide information about the different models that make up the "migrations" domain.

Ecosystem Refers to a group of objects related to each other that work towards a single purpose: migrating a *Target Resource* from point A to point B. In this project, some of those objects are (in no particular order):

- *Timeline*
- *VersionRepository*
- *Storage*
- *Delta*
- *Migration*

A single application can have any number of Migration Ecosystems, but its most common to find only one or two per project.

Migration (noun) A migration is a class that can execute an incremental (when migrating / "going up") or decremental (when reverting / "going down") transformation on a target resource. All classes that implement `MigrationInterface` are considered a migration.

Migrate (verb) Verb used to refer to the process of running one or more migrations.

Migrated (adj.) If something is "migrated" then it means that the migration's `up()` method has executed successfully and the corresponding version is (or will immediately be) stored in the *Storage*. The opposite is true if something has NOT been migrated.

Target Resource During a migrationm the target resource or migration target is WHAT's being migrated. It can be a database, an image, other files, etc. There should only be one target resource for any given Migration Ecosystem.

VersionRepository The VersionRepository is an object that knows where to find *migrations*. It must use a factory to instantiate each migration.

Storage The Storage object is used to persist information about which Versions have already been migrated.

Timeline The Timeline is an object that is in charge of executing migrations in bulk using certain pre-defined algorithms. It holds an immutable *Delta Collection*.

Delta A Delta is a lightweight entity that is persisted into the *Storage* and holds state information about a *Migration*. Currently the only state information being saved is whether the Delta is *migrated* or not. If a Delta is present in the *Storage* then it means that Delta has already been migrated.

Delta Collection A special object to represent and sort a set of Collection. Its important to note that it behaves as an ordered set of elements.

2.2 Migration

A migration in Baleen is a class that implements `MigrationInterface`. It can optionally implement one or more additional interfaces to describe special *capabilities*.

2.2.1 MigrationInterface

All *migration* classes must implement `MigrationInterface`, which declares the `up()` and `down()` methods. Each of those methods should modify the *target resource* symmetrically:

- UP: Describes what happens when the resource moves up (or forwards) in the Timeline.
- DOWN: Describes how to revert / rollback the resource if `up` has already been executed.

The following snippet illustrates a simple migration that creates (or destroys) a table. Note how the up and down methods are perfectly symmetrical to each other:

```
<?php
class v001_AddHireDateToStaff implements MigrationInterface
{

    // a constructor would go here to set up the connection

    /**
      * Adds a "hire_date" column to the staff table.
      */
    public function up() {
        $this->connection->exec("ALTER TABLE staff ADD hire_date date");
    }

    /**
      * Removes the "hire_date" column from the staff table.
      */
    public function down() {
        $this->connection->exec("ALTER TABLE staff REMOVE hire_date date");
    }
}
```

2.2.2 Additional Capabilities

Additional interfaces can be used to indicate special features available for a certain migration. Baleen supplies a few common interfaces for commonly-used features, but more can be created to suit specific needs.

OptionsAwareInterface

A migration that implements this interface must receive an `Options` object after being instantiated. The `Options` object will include contextual information that could be useful for the migration. Refer to the `Options` class docu-

mentation for more information.

TransactionAwareInterface

A migration that implements this interface must implement a set of methods that will aid in generating transactions. Useful for example to wrap `up()` or `down()` commands into database transactions. The methods declared in this interface are:

- `begin()`: called right before executing the migration (i.e. the `up()` or `down()` method).
- `finish()`: called right after the migration finished executing without any exceptions.
- `abort(Exception)`: called if any exception is fired during the execution of the migration. The exception is passed as the only parameter. The `abort` method must recover the resource from the exception (e.g. roll-back the transaction) and optionally re-throw the exception if needed.

Other Capabilities

Additional interfaces can be specified to deal with special features required by any particular migration process. In order to make this easy to customise, migrations are run through a configurable *Command Bus* pattern.

2.2.3 Custom Migrations

Creating specialised Migration classes is very easy: all that's needed is a migration class that end-users must implement in their migrations.

If your abstract migration class has dependencies that need to be injected through its constructor then simply create a `Migration\Factory\FactoryInterface` instance and pass it to the *VersionRepository*:

```
<?php

$factory = new MyMigrationFactory(/** dependencies here **/); // instance of MigrationFactory
$repository->setMigrationFactory($factory);
// $repository->fetchAll() will call the factory to instantiate each Migration
```

The following section is an example of how an abstract PDO migration class can power database migrations.

2.2.4 Example

For the purposes of this example, imagine the following classes are each located on a separate file under a folder in your project called `./migrations`.

The first file declares a sample abstract class to incorporate common functionality. It is of course up to the user whether something like this is really required or not.

File `./migrations/AbstractPDOMigration.php`:

```
<?php
use Baleen\Migration\MigrationInterface;
use Baleen\Migration\Capabilities;
use Baleen\Migration\MigrationInterface;
use Baleen\Migration\RunOptions;

/**
 * You can be as creative as you want here. The only requirement is to implement
 * MigrationInterface.
```

```
 */
class AbstractPDOMigration
    implements MigrationInterface,
               Capabilities/OptionsAwareInterface,
               Capabilities/TransactionAwareInterface
{
    /** @var PDO **/
    protected $connection; // gets initialised in the constructor

    /** @var RunOptions **/
    protected $options;

    public function __construct (PDO $connection) {
        $this->connection = $connection;
    }

    public function begin() {
        $this->connection->beginTransaction();
    }

    public function finish() {
        $this->connection->commit();
    }

    public function abort() {
        $this->connection->rollBack();
    }

    public function setOptions(RunOptions $options) {
        $this->options = $options;
    }
}
```

And now the concrete migrations:

File `./migrations/v001_AddHireDateToStaff.php`:

```
<?php
class v001_AddHireDateToStaff extends AbstractPDOMigration
{
    public function up() {
        $this->connection->exec("ALTER TABLE staff ADD hire_date date");
    }

    public function down() {
        $this->connection->exec("ALTER TABLE staff REMOVE hire_date date");
    }
}
```

File `./migrations/v002_SeedJoeBloggs.php`:

```
<?php
class v002_SeedJoeBloggs extends AbstractPDOMigration
{
    public function up() {
        $this->connection->exec(
            "INSERT INTO staff (id, first, last) VALUES (23, 'Joe', 'Bloggs')"
        );
    }
```

```
    public function down() {
        $this->connection->exec("DELETE FROM staff WHERE id = 23");
    }
}

// ... etc - for the purposes of this example imagine there are 100 migrations
```

2.3 Timeline

A `Timeline` is an object comprised of a version collection that's ordered by *version*.

2.3.1 Public Methods

The Timeline is in charge of executing one or more migrations in any of four different ways:

`upTowards(target)` Executes all migrations in **ascending sequential order**, starting from the first "pending" version, up to *and including* the "target" version - and skipping any versions that have already been migrated. Calls each migration's `up()` method upon execution.

`downTowards(target)` Executes all migrations in **descending sequential order**, starting from the last migration and down until *and including* the "target" version. Calls each migration's `down()` method upon execution.

`goTowards(target)` Combines the previous two operations into one. Executes all migrations `up()` to and including the "target" version. And then executes all remaining versions `down()`, starting from the last available version and ending in the version immediately **after** the "target" version. Useful to make sure all versions before and including "target" are **migrated**, while all others are **pending**.

`runSingle(version, options)` Runs a single version's migration using the specified options. Can be called directly and useful mostly for testing purposes in a controlled environment.

2.3.2 Constructor

When being instantiated the Timeline requires a version collection and a *comparator* as arguments.

`VersionCollection $versions` The version collection must contain at least one *Delta*. All versions in the collection must have a *migration* assigned to them or otherwise an exception will be thrown.

Something to note is that the `VersionCollection` object will be cloned before being stored in the Timeline. Since the Timeline doesn't have any public methods available to access the internal version collection this effectively means that once the Timeline is created its collection cannot be altered.

`callable $comparator` (optional) The comparator must be a `callable` that receives two Versions as arguments and returns a number less than, equal to, or greater than zero (0) if the first version should be executed before than, is the same as, or should be executed after the second version (respectively).

The default comparator simply extracts the first number from each version and subtracts the second one from the first one. See class `Delta\Comparator\DefaultComparator` for the source code.

`MigrationBus $migrationBus` (optional) The `MigrationBus` (colloquially "bus") is an object that can handle a `MigrationCommand` ("command"). The bus consists if a series of `AbstractMiddleware` (middleware), at least one of which must be a `MigrationHandler` (handler). Each middleware object receives the command, handles it (e.g. logs some info), and then calls the next middleware. This goes on until the command arrives at the handler (which is also a middleware), who calls the `up()` or `down()` function in the command's migration and ends the chain.

A default `MigrationBus` is automatically created if none is specified.

2.3.3 Events

Events are an useful to tap into the migration behaviour quickly and cleanly. The Timeline is coupled with a `TimelineEmitter` class that fires domain events for each operation if an `EventDispatcher` is present. Events that will be fired are:

Before Collection, `DomainEventInterface::COLLECTION_BEFORE` Fired before the Timeline executes one of the *collection methods* (upTowards, downTowards, goTowards). The listener callback will receive a *CollectionEvent* object as a parameter, which includes information about the collection of versions, the *Options* and the *target* version.

After Collection, `DomainEventInterface::COLLECTION_AFTER` Same as above, but fired once the the collection method has finished executing all scheduled migrations.

Before Migration, `DomainEventInterface::MIGRATION_BEFORE` Fired before the Timeline executes a single migration, which occurs both as part of executing `runSingle` and also for each of the migrations executed by any *collection method*.

After Migration, `DomainEventInterface::MIGRATION_AFTER` Same as above, but fired once the migration has finished.

Baleen uses Symfony's Event Dispatcher component, so listening to timeline events is very easy:

```
<?php
use Baleen\Migrations\Event;

$dispatcher = $timeline->getEventDispatcher();

$dispatcher->addListener(
    DomainEventInterface::COLLECTION_BEFORE,
    function ($event, $name) {
        // do something
    }
);
$dispatcher->addListener( /* ... */ );
```

You can also inject your own dispatcher (useful for shared listeners e.g. across different Timeline instances):

```
<?php
use Symfony\Component\EventDispatcher;

$dispatcher = new EventDispatcher();
$dispatcher->addListener( /* ... */ );
$dispatcher->addListener( /* ... */ );

$timeline->setEventDispatcher($dispatcher);
```

2.3.4 Custom MigrationBus

As indicated in the *Timeline Constructor* section, a Timeline can receive a `MigrationBus` as an optional parameter. The `MigrationBus` is simply a specialised `CommandBus` that helps provide strong typing for the `MigrateCommand` in PHP 5. Its currently powered by the Tactician command bus library.

The Middleware attached to that bus will be used to execute each individual migration, which means behaviors can easily be customised if needed.

The default migration bus and its middleware stack is created by `MigrationBusFactory` - its very simple:

```
<?php
new MigrationBus([
    // injects Options into the migration
    new SetOptionsMiddleware(),
    // wraps each migration into a transaction (most commonly a DB transaction)
    new TransactionMiddleware(),
    // in charge of executing the MigrateCommand
    new MigrateHandler(),
]);
```

You can override the default `MigrationBus` by simply passing a different instance as a parameter for the Timeline's constructor.

Creating your own middleware is easy too: just create a new class that extends `AbstractMiddleware` and add it to the chain when you create the `MigrationBus`. See any of the default middleware classes for an example of what middleware can do.

CHAPTER 3

Examples

3.1 Usage Example

The following example illustrates how you can use Baleen migrations from within your code to trigger a set of *existing* migrations.

```
<?php
#!/bin/env/php

require __DIR__ . '/vendor/autoload.php';

// The repository is in charge of loading all available migrations.
$repository = new DirectoryRepositoryMapper(__DIR__ . '/migrations');
$available = $repository->getAllAvailableMigrations();

/* The Storage retrieves a list of versions that have already been run.
   Here we're loading from a file, but it could also be a DB table, API call, etc. */
// requires package "baleen/storage-flysystem"
$storage = new FlyStorage(__DIR__ . '/versions.txt');
$migrated = $storage->fetchAll();

$timelineFactory = new TimelineFactory();

/* The Timeline sorts migrations in order and runs them based on their status */
$timeline = $timelineFactory->create($available, $migrated);

// Say we want to make sure all migrations up to and including v015 are UP:
$timeline->upTowards('v015');

// Now lets revert all migrations down to version 13 (inclusive)
$timeline->downTowards('v013'); // will revert 15, 14 and 13 - in that order

/* You can also run a single migration in any direction and pass custom arguments
   to the Migration. */
use Baleen\Migration\Options;
$options = (new Options())->withCustom([
    'notifyEmail' => 'jon@doe.me',
]);
$timeline->runSingle('v100', $options);

/* Delta 'v100' will receive an instance of RunOptions through the setOptions
   method. You can also pass RunOptions to most of the other Timeline methods. */
```

CHAPTER 4

Implementations

Since Baleen Migrations is just taking its first steps there are still no major end-user libraries / implementations that are already using it. You could be the first one to build one!

- Baleen CLI: command line (CLI) wrapper to execute simple migrations. Still in active development.
- Pimcore Migrations: a migrations library for Pimcore 3.x.
- Doctrine Migrations (pre-alpha): a soon-to-be PR for Doctrine Migrations. Still in progress :)

CHAPTER 5

Reference

5.1 Baleen\Migrations

5.1.1 Baleen\Migrations\Event

Baleen\Migrations\Event\CanDispatchEventsTrait

trait `CanDispatchEventsTrait`

Allows classes to OPTIONALLY receive an EventDispatcher to dispatch events. Making it optional puts responsibility on the implementation: if the EventDispatcher is not available then no events are dispatched.

property `eventDispatcher`

protected null|EventDispatcherInterface

`getEventDispatcher`()

Returns null|EventDispatcherInterface

`setEventDispatcher`(*EventDispatcherInterface $eventDispatcher*)

Parameters

- **`$eventDispatcher`** (*`EventDispatcherInterface`*) –

`dispatchEvent`(*$name*, *Event $event*)

Dispatches an event if an EventDispatcher is available.

Parameters

- **`$name`** –
- **`$event`** (*`Event`*) –

Returns Event

Baleen\Migrations\Event\HasEmitterTrait

trait `HasEmitterTrait`

Trait to be used by classes that fire events through a specialised emitter. For example, the Timeline class will use this trait to fire events using the TimelineEmitter.

property `emitter`

protected PublisherInterface

setEventDispatcher(*EventDispatcherInterface $eventDispatcher*)
Set the EventDispatcher for the emitter. This is public to allow attaching a previously existing EventDispatcher.

Parameters

- **$eventDispatcher** (*EventDispatcherInterface*) –

getEmitter()
Returns the emitter for the class. Creates one if necessary.

Returns PublisherInterface

setEmitter(*PublisherInterface $emitter*)
Sets the emitter for the class.

Parameters

- **$emitter** (*PublisherInterface*) –

createEmitter()
Must create and return a default emitter.

Returns PublisherInterface

getEventDispatcher()
Get the event dispatcher from the emitter.

Returns EventDispatcherInterface

Baleen\Migrations\Event\Timeline

Baleen\Migrations\Service\Runner\Event\Collection\CollectionEvent

class CollectionEvent
Class CollectionEvent.

property collection
protected Linked

property options
protected OptionsInterface

property target
protected DeltaInterface

property progress
protected Progress

__construct(*DeltaInterface $target*, *OptionsInterface $options*, *Linked $versions*, *Progress $progress = null*)
CollectionEvent constructor.

Parameters

- **$target** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$versions** (*Linked*) –
- **$progress** (Progress) –

getOptions()

Returns OptionsInterface

getCollection()

Returns Linked

getTarget()

Returns DeltaInterface

getProgress()

Returns Progress

Baleen\Migrations\Service\Runner\Event\Migration\MigrationEvent

class MigrationEvent
Class MigrationEvent.

property options
protected OptionsInterface

property version
protected DeltaInterface

property progress
protected Progress

__construct(*DeltaInterface $version*, *OptionsInterface $options*, *Progress $progress = null*)
MigrationEvent constructor.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$progress** (Progress) –

getOptions()

Returns OptionsInterface

getVersion()

Returns DeltaInterface

getProgress()

Returns Progress

Baleen\Migrations\Shared\Event\Progress

class Progress
Class Progress.

property total
protected

property current
protected

__construct(*$total*, *$current*)
Progress constructor.

Parameters

- **$total** (*int*) –
- **$current** (*int*) –

getTotal()

Returns int

getCurrent()

Returns int

update(*$current*)

Parameters

- **$current** (*int*) –

5.1.2 Baleen\Migrations\Exception

Baleen\Migrations\Exception\BaleenException

class BaleenException

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Baleen\Migrations\Exception\CollectionException

class CollectionException
Class CollectionException.

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Baleen\Migrations\Exception\InvalidArgumentException

class InvalidArgumentException

property message
protected

property code
protected

property file
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

`getFile`()

`getLine`()

`getTrace`()

`getPrevious`()

`getTraceAsString`()

`__toString`()

Baleen\Migrations\Exception\MigrationBusException

class `MigrationBusException`
Class MigrationBusException.

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Baleen\Migrations\Exception\MigrationException

class MigrationException

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Baleen\Migrations\Exception\MigrationExceptionInterface

class MigrationExceptionInterface

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

`getFile`()

`getLine`()

`getTrace`()

`getPrevious`()

`getTraceAsString`()

`__toString`()

Baleen\Migrations\Exception\MigrationMissingException

class `MigrationMissingException`

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –

- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Baleen\Migrations\Exception\Migration\VersionRepository\RepositoryException

class RepositoryException

Class RepositoryException.

property message

protected

property code

protected

property file

protected

property line

protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Baleen\Migrations\Exception\ResolverException

class `ResolverException`
Class ResolverException

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

`getFile`()

`getLine`()

`getTrace`()

`getPrevious`()

`getTraceAsString`()

`__toString`()

Baleen\Migrations\Exception\TimelineException

class `TimelineException`
Class TimelineException.

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

__construct (*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup ()

getMessage ()

getCode ()

getFile ()

getLine ()

getTrace ()

getPrevious ()

getTraceAsString ()

__toString ()

5.1.3 Baleen\Migrations\Migration

Baleen\Migrations\Migration\AbstractMigration

class AbstractMigration

A simple migration base class. To be used for examples and tests or extended by more complex classes.

getOptions ()

Returns OptionsInterface

setOptions (*OptionsInterface $options*)

Parameters

- **$options** (OptionsInterface) –

up ()

down ()

Baleen\Migrations\Migration\Capabilities

Baleen\Migrations\Migration\Capabilities\OptionsAwareInterface

interface OptionsAwareInterface

Interface OptionsAwareInterface.

setOptions (*OptionsInterface $options*)

Receive an OptionsInterface instance

Parameters

- **$options** (OptionsInterface) –

Returns void

Baleen\Migrations\Migration\Capabilities\TransactionAwareInterface

interface `TransactionAwareInterface`
Indicates a migration can be handled within a transaction (commonly used in database migrations) and provides methods for the different stages of a transaction.

`begin`()
Called when the transaction should be begun.

Returns void

`finish`()
Called when the transaction should be finished.

Returns void

`abort`(*Exception $e*)
Called when the transaction should be cancelled.

Parameters

- **`$e`** (*`Exception`*) –

Returns void

Baleen\Migrations\Migration\Command

Baleen\Migrations\Migration\Command\Middleware

Baleen\Migrations\Migration\Command\Middleware\AbstractMiddleware
class `AbstractMiddleware`
Enforces command type checking, to make sure that all commands ran by these Middleware classes are able to handle MigrateCommand.

`execute`(*$command*, *$next*)

Parameters

- **`$command`** (*`object`*) –
- **`$next`** (*`callable`*) –

Returns mixed

`doExecute`(*MigrateCommand $command*, *$next*)
Concrete handling of the MigrateCommand.

Parameters

- **`$command`** (`MigrateCommand`) –
- **`$next`** (*`callable`*) –

Returns mixed

Baleen\Migrations\Migration\Command\Middleware\SetOptionsMiddleware
class `SetOptionsMiddleware`
Checks if a migration is an instance of OptionsAwareInterface and if so sends it the options available in the command.

doExecute (*MigrateCommand $command*, *$next*)
{@inheritDoc}

Parameters

- **$command** (MigrateCommand) –
- **$next** –

execute (*$command*, *$next*)

Parameters

- **$command** (*object*) –
- **$next** (*callable*) –

Returns mixed

Baleen\Migrations\Migration\Command\Middleware\TransactionMiddleware
class TransactionMiddleware
Wraps the migration in a transaction if the migration implements TransactionAwareInterface.

doExecute (*MigrateCommand $command*, *$next*)
{@inheritDoc}

Parameters

- **$command** (MigrateCommand) –
- **$next** –

execute (*$command*, *$next*)

Parameters

- **$command** (*object*) –
- **$next** (*callable*) –

Returns mixed

Baleen\Migrations\Migration\Command\MigrateCommand

class MigrateCommand
Class MigrateCommand.

__construct (*MigrationInterface $migration*, *OptionsInterface $options*)

Parameters

- **$migration** (MigrationInterface) –
- **$options** (OptionsInterface) –

getMigration ()

Returns MigrationInterface

setMigration (*MigrationInterface $migration*)

Parameters

- **$migration** (MigrationInterface) –

getOptions ()

Returns OptionsInterface

setOptions (*OptionsInterface $options*)

Parameters

- **$options** (OptionsInterface) –

Baleen\Migrations\Migration\Command\MigrateHandler

class MigrateHandler

Class MigrateHandler.

doExecute (*MigrateCommand $command*, *$next*)

{@inheritDoc}

Parameters

- **$command** (MigrateCommand) –
- **$next** –

execute (*$command*, *$next*)

Parameters

- **$command** (*object*) –
- **$next** (*callable*) –

Returns mixed

Baleen\Migrations\Migration\Command\MigrationBus

class MigrationBus

__construct (*$middleware*)

Parameters

- **$middleware** –

Baleen\Migrations\Migration\Command\MigrationBusFactory

class MigrationBusFactory

Class CommandBusFactory.

create ()

Baleen\Migrations\Migration\Factory

Baleen\Migrations\Migration\Factory\FactoryInterface

interface FactoryInterface

Interface for a Migrations Factory.

create (*$class*, *$args* = [ ])

Creates a Migration based on a class name.

Parameters

- **$class** (*string*) – The FQN of the migration class to be instantiated.
- **$args** (*array*) – Constructor parameters.

Returns BaleenMigrationsMigrationMigrationInterface

Baleen\Migrations\Migration\Factory\SimpleFactory

class SimpleFactory

create (*$class*, *$args* = [ ])

Parameters

- **$class** –
- **$args** –

Baleen\Migrations\Migration\MigrationInterface

interface MigrationInterface

Interface to be implemented by all migration classes in the end-project.

up ()

down ()

Baleen\Migrations\Migration\Options

class Options

__construct (*$direction = self::DIRECTION_UP*, *$forced = false*, *$dryRun = false*, *$exceptionOnSkip = true*, *$custom* = [ ])

Parameters

- **$direction** –
- **$forced** (*bool*) –
- **$dryRun** (*bool*) –
- **$exceptionOnSkip** (*bool*) –
- **$custom** (*array*) –

setDirection (*$direction*)

setDirection

Parameters

- **$direction** –

getDirection ()

getDirection

Returns string

withDirection (*$direction*)

Parameters

- **$direction** (*string*) –

Returns static

isDirectionUp()

Returns bool

isDirectionDown()

Returns bool

isForced()

Returns bool

withForced(*$forced*)
withForced

Parameters

- **$forced** –

Returns static

isDryRun()

Returns bool

withDryRun(*$dryRun*)
withDryRun

Parameters

- **$dryRun** (*bool*) –

Returns static

isExceptionOnSkip()

Returns bool

withExceptionOnSkip(*$exceptionOnSkip*)

Parameters

- **$exceptionOnSkip** (*bool*) –

Returns static

getCustom()

Returns array

withCustom(*$custom*)

Parameters

- **$custom** (*array*) –

Returns static

isSameValueAs(*OptionsInterface $options*)
Compares the current instance with another instance of options to see if they contain the same values.

Parameters

- **$options** (OptionsInterface) –

Returns bool

Baleen\Migrations\Migration\OptionsInterface

interface `OptionsInterface`
Options value object. Used to configure the migration jobs and provide information about them to the migration.

`getDirection`()
The direction that we're migrating

Returns string

`withDirection`(*$direction*)
MUST return a new OptionsInterface instance with the same property values as the current one except for the new direction.

Parameters

- **`$direction`** (*string*) –

Returns static

`isDirectionUp`()

Returns bool

`isDirectionDown`()

Returns bool

`isForced`()

Returns bool

`withForced`(*$forced*)
MUST return a new OptionsInterface instance with the same property values as the current one except for the new value for the "forced" property.

Parameters

- **`$forced`** –

Returns static

`isDryRun`()

Returns bool

`withDryRun`(*$dryRun*)
MUST return a new OptionsInterface instance with the same property values as the current one except for the new value for the "dryRun" property.

Parameters

- **`$dryRun`** (*bool*) –

Returns static

`isExceptionOnSkip`()

Returns bool

`withExceptionOnSkip`(*$exceptionOnSkip*)
MUST return a new OptionsInterface instance with the same property values as the current one except for the new value for the "exceptionOnSkip" property.

Parameters

- **$exceptionOnSkip** (*bool*) –

Returns static

getCustom()

Returns array

withCustom(*$custom*)
MUST return a new OptionsInterface instance with the same property values as the current one except for the new value for the "custom" array.

Parameters

- **$custom** (*array*) –

Returns static

isSameValueAs(*OptionsInterface $options*)
Returns true if the current object is the same as the parameter.

Parameters

- **$options** (OptionsInterface) –

Returns boolean

5.1.4 Baleen\Migrations\Storage

5.1.5 Baleen\Migrations\Timeline

Baleen\Migrations\Timeline\AbstractTimeline

class AbstractTimeline
Encapsulates the lower-level methods of a Timeline, leaving the actual timeline logic to the extending class.

property emitter
protected PublisherInterface

__construct(*Linked $versions*, *MigrationBus $migrationBus = null*)

Parameters

- **$versions** (*Linked*) –
- **$migrationBus** (MigrationBus) – A CommandBus that will be used to run each individual migration.

shouldMigrate(*DeltaInterface $version*, *OptionsInterface $options*)
Returns true if the operatin is forced, or if the direction is the opposite to the state of the migration.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –

Returns bool

createEmitter()
Must create and return a default specialised dispatcher.

Returns BaleenMigrationsSharedEventPublisherInterface

doRun (*MigrationInterface $migration*, *OptionsInterface $options*)

Parameters

- **$migration** (MigrationInterface) –
- **$options** (OptionsInterface) –

Returns bool

runCollection (*DeltaInterface $goalVersion*, *OptionsInterface $options*, *Linked $collection*)
Executes migrations against a collection

Parameters

- **$goalVersion** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$collection** (*Linked*) –

Returns Linked

getVersions ()
getVersions

Returns Linked

setEventDispatcher (*EventDispatcherInterface $eventDispatcher*)
Set the EventDispatcher for the emitter. This is public to allow attaching a previously existing EventDispatcher.

Parameters

- **$eventDispatcher** (*EventDispatcherInterface*) –

getEmitter ()
Returns the emitter for the class. Creates one if necessary.

Returns PublisherInterface

setEmitter (*PublisherInterface $emitter*)
Sets the emitter for the class.

Parameters

- **$emitter** (*PublisherInterface*) –

getEventDispatcher ()
Get the event dispatcher from the emitter.

Returns EventDispatcherInterface

upTowards (*$version*, *OptionsInterface $options*)
Runs all versions up, starting from the oldest and until (and including) the specified version.

Parameters

- **$version** (*string|DeltaInterface*) –
- **$options** (OptionsInterface) –

downTowards (*$version*, *OptionsInterface $options*)
Runs all versions down, starting from the newest and until (and including) the specified version.

Parameters

- **$version** (*string|DeltaInterface*) –

- **$options** (OptionsInterface) –

goTowards (*$goalVersion*, *OptionsInterface $options*)
Runs migrations up/down so that all versions *before and including* the specified version are "up" and all versions *after* the specified version are "down".

Parameters

- **$goalVersion** (*string|DeltaInterface*) –
- **$options** (OptionsInterface) –

runSingle (*DeltaInterface $version*, *OptionsInterface $options*, *Progress $progress*)
Runs a single migration in the specified direction.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$progress** (Progress) –

Returns DeltaInterface|false

Baleen\Migrations\Timeline\TimelineEmitter

class TimelineEmitter
Dispatches Timeline events.

property eventDispatcher
protected null|EventDispatcherInterface

dispatchCollectionBefore (*DeltaInterface $targetVersion*, *OptionsInterface $options*, *Linked $versions*, *Progress $progress = null*)

Parameters

- **$targetVersion** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$versions** (*Linked*) –
- **$progress** (Progress) –

Returns SymfonyComponentEventDispatcherEvent|void

dispatchCollectionAfter (*DeltaInterface $targetVersion*, *OptionsInterface $options*, *Linked $versions*, *Progress $progress = null*)
dispatchCollectionAfter.

Parameters

- **$targetVersion** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$versions** (*Linked*) –
- **$progress** (Progress) –

Returns SymfonyComponentEventDispatcherEvent|void

dispatchMigrationBefore (*DeltaInterface $version*, *OptionsInterface $options*, *Progress $progress = null*)
dispatchMigrationBefore.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$progress** (Progress) –

Returns SymfonyComponentEventDispatcherEvent|void

dispatchMigrationAfter(*DeltaInterface $version*, *OptionsInterface $options*, *Progress $progress = null*)

dispatchMigrationAfter.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$progress** (Progress) –

Returns SymfonyComponentEventDispatcherEvent|void

getEventDispatcher()

Returns null|EventDispatcherInterface

setEventDispatcher(*EventDispatcherInterface $eventDispatcher*)

Parameters

- **$eventDispatcher** (*EventDispatcherInterface*) –

dispatchEvent(*$name*, *Event $event*)

Dispatches an event if an EventDispatcher is available.

Parameters

- **$name** –
- **$event** (*Event*) –

Returns Event

Baleen\Migrations\Timeline\TimelineFactory

class TimelineFactory

__construct(*ResolverInterface $resolver = null*, *ComparatorInterface $comparator = null*, *EventDispatcher $dispatcher = null*)

Parameters

- **$resolver** (*ResolverInterface*) –
- **$comparator** (*ComparatorInterface*) –
- **$dispatcher** (*EventDispatcher*) –

create(*$available*, *$migrated* = [])

Creates a Timeline instance with all available versions. Those versions that have already been migrated will be marked accordingly.

Parameters

- **$available** (*array|Linked*) –

- **$migrated** (*array|Migrated*) –

Returns Timeline

prepareCollection (*$available*, *$migrated* = [ ])
Sets versions in $this->availableVersions to migrated if they appear in $this->migratedVersions.

Parameters

- **$available** (*array|Linked*) –
- **$migrated** (*array|Migrated*) –

Returns Linked

Baleen\Migrations\Timeline\TimelineInterface

interface TimelineInterface
The Timeline is responsible of emitting MigrateCommands based on how the user wants to navigate the timeline (e.g. travel to a specific version). It takes into account the current state.

upTowards (*$version*, *OptionsInterface $options*)
Runs all versions up, starting from the oldest and until (and including) the specified version.

Parameters

- **$version** (*string|DeltaInterface*) –
- **$options** (OptionsInterface) –

downTowards (*$version*, *OptionsInterface $options*)
Runs all versions down, starting from the newest and until (and including) the specified version.

Parameters

- **$version** (*string|DeltaInterface*) –
- **$options** (OptionsInterface) –

goTowards (*$goalVersion*, *OptionsInterface $options*)
Runs migrations up/down so that all versions *before and including* the specified version are "up" and all versions *after* the specified version are "down".

Parameters

- **$goalVersion** (*string|DeltaInterface*) –
- **$options** (OptionsInterface) –

runSingle (*DeltaInterface $version*, *OptionsInterface $options*, *Progress $progress*)
Runs a single migration in the specified direction.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$progress** (Progress) –

Returns DeltaInterface|false

getVersions ()
getVersions

Returns Linked

5.1.6 Baleen\Migrations\Timeline

class `Timeline`

property `emitter`
protected PublisherInterface

`upTowards` (*$goalVersion*, *OptionsInterface $options* = *null*)

Parameters

- **`$goalVersion`** (*`DeltaInterface|string`*) –
- **`$options`** (`OptionsInterface`) –

Returns Sortable A collection of modified versions

`downTowards` (*$goalVersion*, *OptionsInterface $options* = *null*)

Parameters

- **`$goalVersion`** (*`DeltaInterface|string`*) –
- **`$options`** (`OptionsInterface`) –

Returns Sortable A collection of modified versions

`goTowards` (*$goalVersion*, *OptionsInterface $options* = *null*)
Runs migrations up/down so that all versions *before and including* the specified version are "up" and all versions *after* the specified version are "down".

Parameters

- **`$goalVersion`** –
- **`$options`** (`OptionsInterface`) –

Returns Linked A collection of versions that were *changed* during the process. Note that this collection may significantly defer from what would be obtained by $this->getVersions()

`runSingle` (*DeltaInterface $version*, *OptionsInterface $options*, *Progress $progress* = *null*)

Parameters

- **`$version`** (*`DeltaInterface`*) –
- **`$options`** (`OptionsInterface`) –
- **`$progress`** (`Progress`) – Provides contextual information about current progress if this migration is one of many that are being run in batch.

Returns DeltaInterface|false

`__construct` (*Linked $versions*, *MigrationBus $migrationBus* = *null*)

Parameters

- **`$versions`** (*`Linked`*) –
- **`$migrationBus`** (`MigrationBus`) – A CommandBus that will be used to run each individual migration.

`shouldMigrate` (*DeltaInterface $version*, *OptionsInterface $options*)
Returns true if the operatin is forced, or if the direction is the opposite to the state of the migration.

Parameters

- **$version** (*DeltaInterface*) –
- **$options** (OptionsInterface) –

Returns bool

createEmitter()
Must create and return a default specialised dispatcher.

Returns BaleenMigrationsSharedEventPublisherInterface

doRun(*MigrationInterface $migration*, *OptionsInterface $options*)

Parameters

- **$migration** (MigrationInterface) –
- **$options** (OptionsInterface) –

Returns bool

runCollection(*DeltaInterface $goalVersion*, *OptionsInterface $options*, *Linked $collection*)
Executes migrations against a collection

Parameters

- **$goalVersion** (*DeltaInterface*) –
- **$options** (OptionsInterface) –
- **$collection** (*Linked*) –

Returns Linked

getVersions()
getVersions

Returns Linked

setEventDispatcher(*EventDispatcherInterface $eventDispatcher*)
Set the EventDispatcher for the emitter. This is public to allow attaching a previously existing EventDispatcher.

Parameters

- **$eventDispatcher** (*EventDispatcherInterface*) –

getEmitter()
Returns the emitter for the class. Creates one if necessary.

Returns PublisherInterface

setEmitter(*PublisherInterface $emitter*)
Sets the emitter for the class.

Parameters

- **$emitter** (*PublisherInterface*) –

getEventDispatcher()
Get the event dispatcher from the emitter.

Returns EventDispatcherInterface

CHAPTER 6

Indices and Tables

- genindex

Symbols

A

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

U

V

W