DMS Developer Documentation

Release 0.8

DMS

Jul 21, 2017

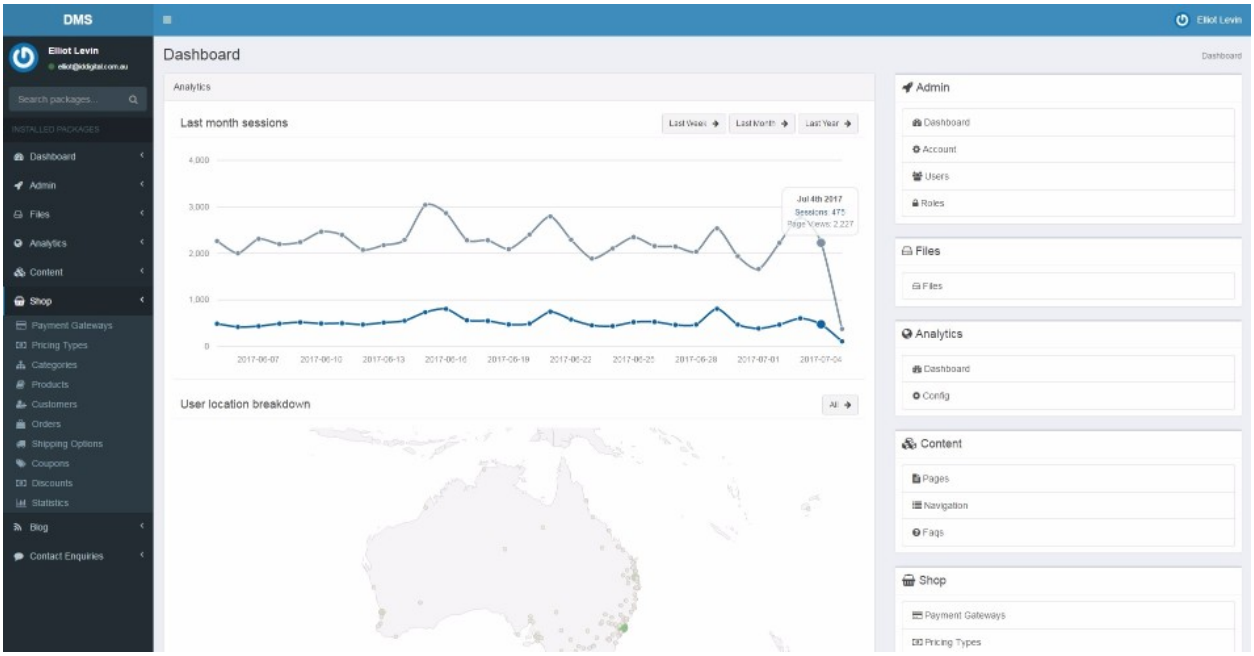
Contents

DMS is PHP7 application framework that helps to develop maintainable and robust apps of any scale.

- Integrates with Laravel
- Promotes separation of concerns and defines clear architectural boundaries
- Build strongly typed, rich or anemic domain models to suit your application
- Fully-featured data mapper ORM
- Provides an integrated and powerful CMS/Backend framework
- Extensive scaffolding to speed up application development
- Reusuable functionality can be installed via composer packages

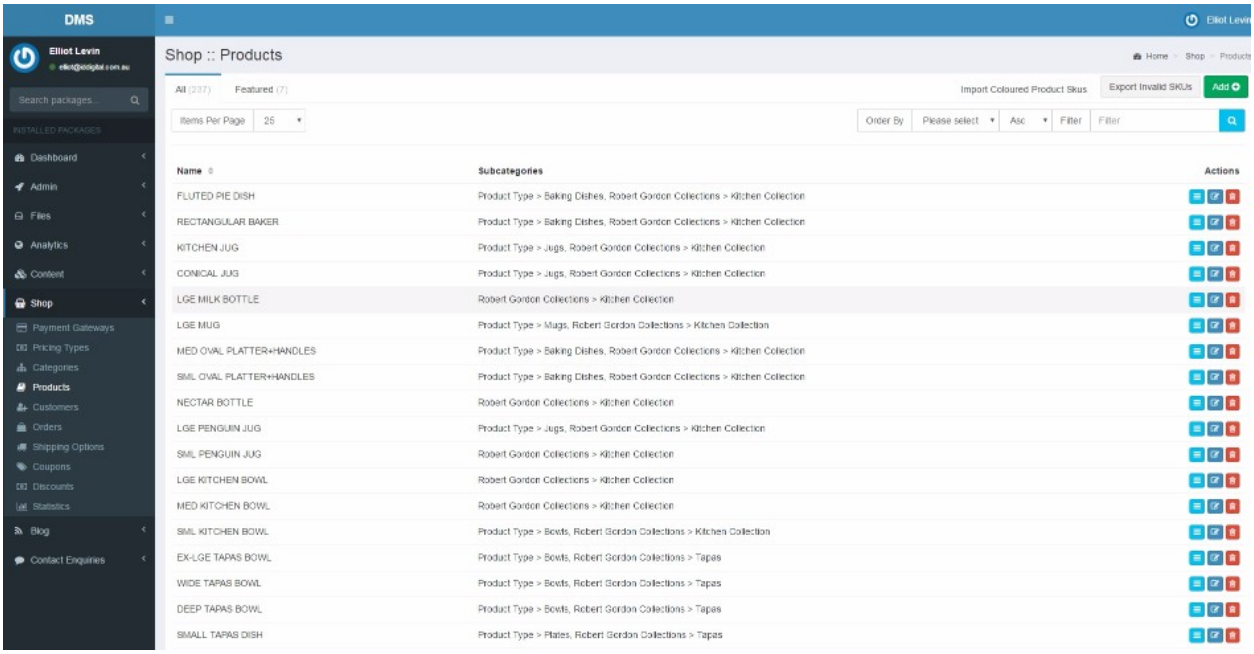
CHAPTER 1

Dashboard



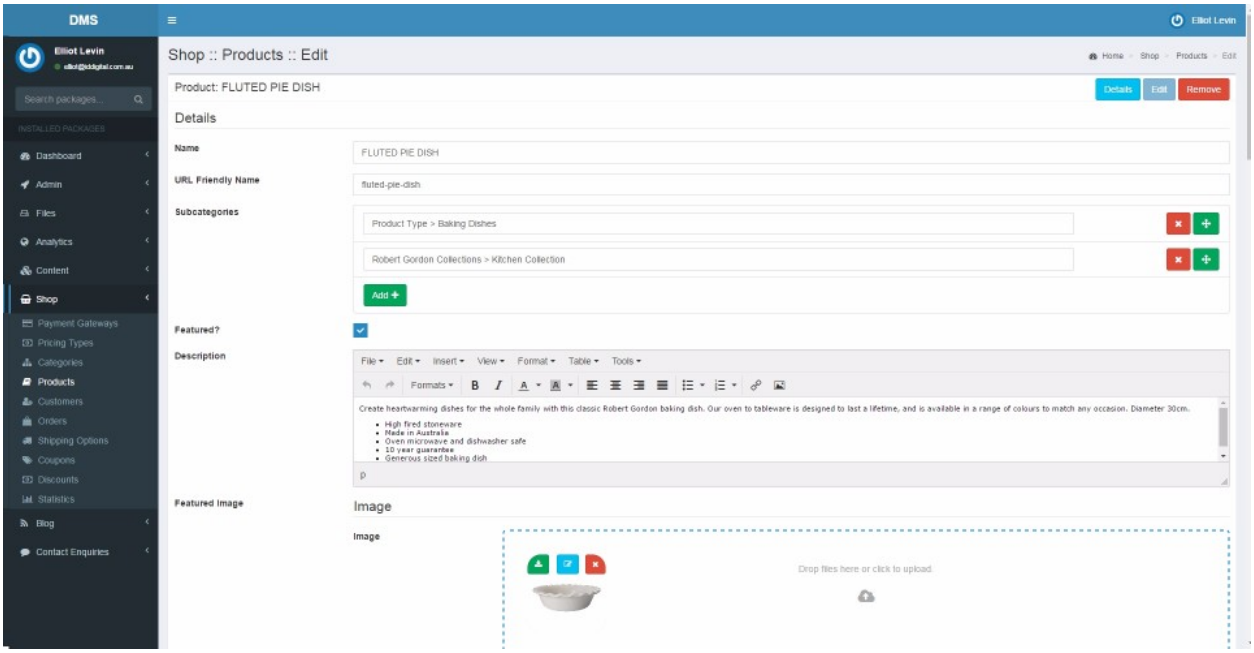
CHAPTER 2

Overview



CHAPTER 3

Edit



CHAPTER 4

Code

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

namespace Dms\Package\Shop\Domain\Entities\Order;

use Dms\Common\Structure\DateTime\DateTime;
use Dms\Common\Structure\Money\Money;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;
use Dms\Core\Model\ValueObjectCollection;
use Dms\Core\Util\IClock;
use Dms\Package\PaymentGateway\Gateway\Response\TransactionReference;
use Dms\Package\Shop\Coupon\Core\Coupon;
use Dms\Package\Shop\Domain\Entities\Pricing\PricingType;
use Dms\Package\Shop\Domain\Exception\ShopException;

/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class Order extends Entity
{
    const ORDER_NUMBER = 'orderNumber';
    const STATUS = 'status';
    const CUSTOMER_DETAILS = 'customerDetails';
    const PRICING_TYPE = 'pricingType';
    const ITEMS = 'items';
    const DISCOUNT = 'discount';
    const SHIPPING_DETAILS = 'shippingDetails';
    const TAX_FEE = 'taxFee';
    const COUPON = 'coupon';
    const NOTES = 'notes';
    const TRANSACTION_REFERENCE = 'transactionReference';
    const AMOUNT_REFUNDED = 'amountRefunded';
    const CREATED_AT = 'createdAt';
    const METADATA = 'metadata';
```

```
    /**
     * @var string
     */
    public $orderNumber;

    /**
     * @var OrderStatus
     */
    public $status;

    /**
     * @var CustomerDetails
     */
    public $customerDetails;

    /**
     * @var PricingType
     */
    public $pricingType;

    /**
     * @var ValueObjectCollection|OrderItem[]
     */
    public $items;

    /**
     * @var Money|null
     */
    public $discount;

    /**
     * @var ShippingDetails|null
     */
    public $shippingDetails;

    /**
     * @var Money|null
     */
    public $taxFee;

    /**
     * @var Coupon|null
     */
    public $coupon;

    /**
     * @var DateTime
     */
    public $createdAt;

    /**
     * @var TransactionReference|null
     */
    public $transactionReference;

    /**
     * @var Money
     */
```

```
    public $amountRefunded;

    /**
     * @var ValueObjectCollection|OrderNote[]
     */
    public $notes;

    /**
     * @var \ArrayObject
     */
    public $metadata;

    /**
     * Order constructor.
     *
     * @param string                    $orderNumber
     * @param OrderStatus               $status
     * @param CustomerDetails           $customerDetails
     * @param IClock                    $clock
     * @param PricingType               $pricingType
     * @param OrderItem[]               $items
     * @param Money|null                $discount
     * @param ShippingDetails           $shippingDetails
     * @param Money|null                $taxFee
     * @param Coupon|null               $coupon
     * @param TransactionReference|null $transactionReference
     * @param array                     $metadata
     */
    public function __construct(
        string $orderNumber,
        OrderStatus $status,
        CustomerDetails $customerDetails,
        IClock $clock,
        PricingType $pricingType,
        array $items,
        Money $discount = null,
        ShippingDetails $shippingDetails = null,
        Money $taxFee = null,
        Coupon $coupon = null,
        TransactionReference $transactionReference = null,
        array $metadata = []
    ) {
        parent::__construct();
        $this->orderNumber          = $orderNumber;
        $this->status               = $status;
        $this->customerDetails      = $customerDetails;
        $this->pricingType          = $pricingType;
        $this->items                = OrderItem::collection($items);
        $this->discount             = $discount;
        $this->shippingDetails      = $shippingDetails;
        $this->taxFee               = $taxFee;
        $this->coupon               = $coupon;
        $this->transactionReference = $transactionReference;
        $this->amountRefunded       = new Money(0, $pricingType->currency);
        $this->notes                = OrderNote::collection();
        $this->createdAt            = new DateTime($clock->utcNow());
        $this->metadata             = new \ArrayObject($metadata);
    }
```

```
    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->orderNumber)->asString();

        $class->property($this->status)->asObject(OrderStatus::class);

        $class->property($this->customerDetails)->asObject(CustomerDetails::class);

        $class->property($this->pricingType)->asObject(PricingType::class);

        $class->property($this->items)->asType(OrderItem::collectionType());

        $class->property($this->shippingDetails)->nullable()->
↪asObject(ShippingDetails::class);

        $class->property($this->discount)->nullable()->asObject(Money::class);

        $class->property($this->taxFee)->nullable()->asObject(Money::class);

        $class->property($this->coupon)->nullable()->asObject(Coupon::class);

        $class->property($this->transactionReference)->nullable()->
↪asObject(TransactionReference::class);

        $class->property($this->amountRefunded)->asObject(Money::class);

        $class->property($this->notes)->asType(OrderNote::collectionType());

        $class->property($this->createdAt)->asObject(DateTime::class);

        $class->property($this->metadata)->asObject(\ArrayObject::class);
    }

    /**
     * @return Money
     */
    public function getSubtotal() : Money
    {
        $amountOfMoney = new Money(0, $this->pricingType->currency);

        foreach ($this->items as $item) {
            /** @var OrderItem $item */
            $amountOfMoney = $amountOfMoney->add($item->getTotalPrice());
        }

        return $amountOfMoney;
    }

    /**
     * @return Money
     */
    public function getTotal() : Money
```

```
    {
        $amountOfMoney = $this->getSubtotal();

        if ($this->discount) {
            $amountOfMoney = $amountOfMoney->subtract($this->discount);
        }

        if ($this->shippingDetails && $this->shippingDetails->shippingFee) {
            $amountOfMoney = $amountOfMoney->add($this->shippingDetails->shippingFee);
        }

        if ($this->taxFee) {
            $amountOfMoney = $amountOfMoney->add($this->taxFee);
        }

        return $amountOfMoney;
    }

    /**
     * @return bool
     */
    public function requiresShipping() : bool
    {
        return $this->items->any(function (OrderItem $item) {
            return $item->requiresShipping;
        });
    }

    /**
     * @return Money
     * @throws ShopException
     */
    public function getMaxRefundAmount() : Money
    {
        if (!$this->transactionReference) {
            throw new ShopException('The order has no transaction');
        }

        return $this->transactionReference->amount->subtract($this->amountRefunded);
    }
}
```

CHAPTER 5

User Guide

Prologue

The DMS framework was borne out of frustration from multiple attempts at creating maintainable applications using the Laravel framework. Although possible, the Laravel structure does not encourage clear separation of concerns which could cause problems when building complex applications.

A common requirement of projects is for content and functionality to be manageable through an admin panel. These can be tedious and time consuming to set up on a per-project basis.

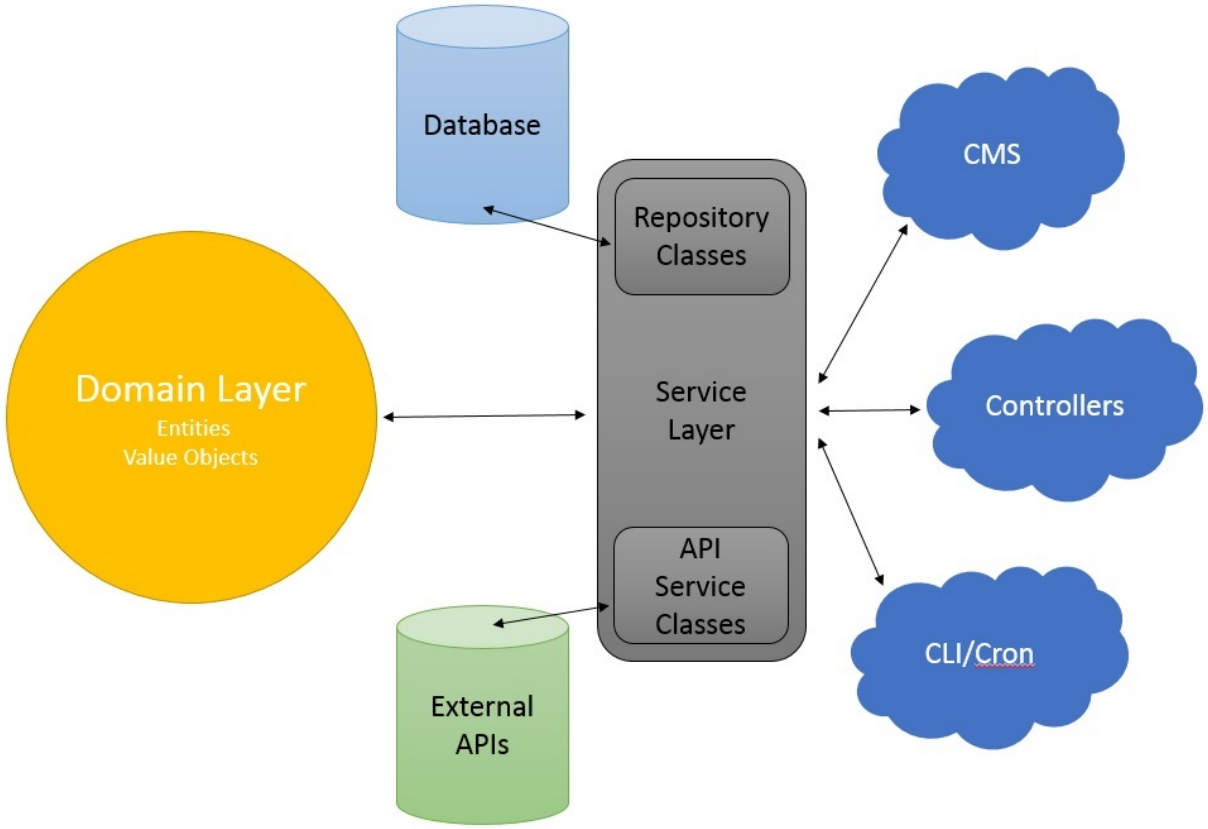
The DMS aims to alleviate these issues by encouraging a clear application structure, inspired by concepts from DDD and Hexagonal Architecture, to allow complexity to be expressed through robust and maintainable code. While also providing an integrated CMS framework to build powerful backends quickly and easily.

Terminology

Building on concepts of DDD, this documentation uses similar language and jargon. See glossary.

Architecture

All applications must define their business logic somewhere in the application. Every developer has their own opinion on how to structure an application, the DMS takes an approach where the application is separated into layers. The functionality of an application should be contained within service classes which expose an API to be used by each application entry point, whether it be the web UI, CLI or CMS.



See blog package for example code

Getting Started

Requirements

- PHP 7.0+
- Laravel 5.4+

Installation

It is recommended to install the DMS via Composer in a fresh Laravel project.

1. Create a new Laravel project More details

```
composer create-project --prefer-dist laravel/laravel your-project-name
cd your-project-name
```

2. Edit the `.env` file and enter the correct database credentials.

```
# Enter the correct database credentails
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
```

```
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

3. Install the latest version of DMS via Composer

```
composer require dms-org/web.laravel
```

4. Add the service provider your `config/app.php`

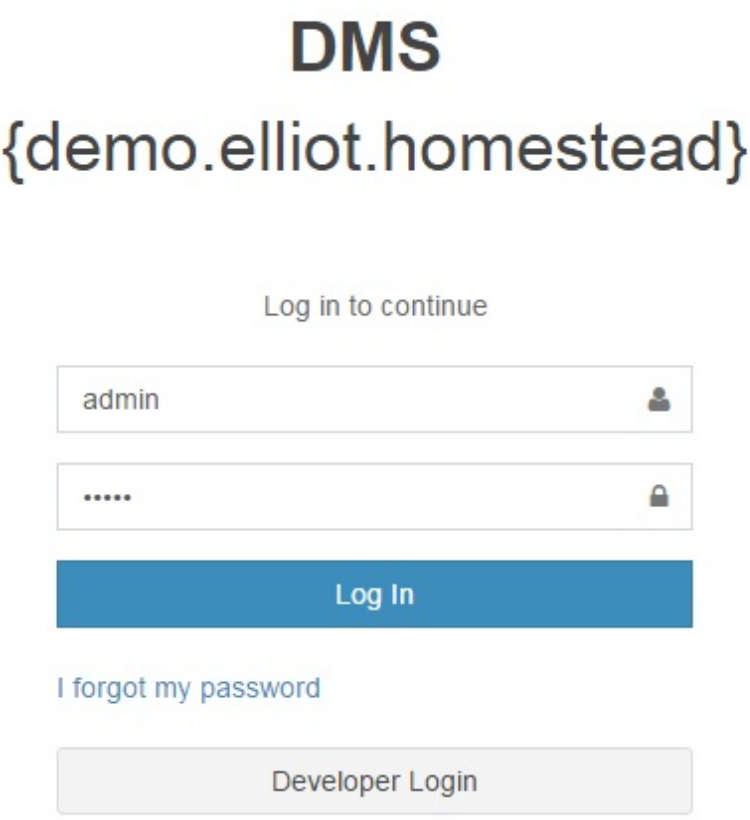
```
/*
 * Package Service Providers...
 */
Dms\Web\Laravel\DmsServiceProvider::class,
```

5. Run the DMS installation command

```
php artisan dms:install
```

6. Visit `http://your-app-domain/dms` to view the backend of your new project

   You can login with the default user account. **U:** admin **P:** admin.



Your First App

To illustrate how to build apps using the DMS, let's go through building a simple TODO list app. *See installation guide if necessary*

Defining your domain model

Firstly lets build our entities. Lets create a file `app/Domain/Entities/TodoItem.php`.

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

/**
 * Your first entity.
```

```
 *
 * An item on the TODO list.
 */
class TodoItem extends Entity
{
    // Define constants for property names
    const DESCRIPTION = 'description';
    const COMPLETED = 'completed';

    /**
     * @var string
     */
    public $description;

    /**
     * @var bool
     */
    public $completed;

    /**
     * Initialises a new TODO item.
     *
     * @param string $description
     */
    public function __construct(string $description)
    {
        parent::__construct();
        $this->description = $description;
        $this->completed   = false;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        // Enables strong typing for this entity
        $class->property($this->description)->asString();

        $class->property($this->completed)->asBool();
    }
}
```

Building your persistence layer

So now we have our entities, we need a way to persist them to a database.

Defining the mapper

Create the entity mapper under `app/Infrastructure/Persistence/TodoItemMapper.php`. Configure how to map the entity to the database table.

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\TodoItem;

/**
 * Your first entity mapper.
 *
 * The App\Domain\Entities\TodoItem entity mapper.
 */
class TodoItemMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(TodoItem::class);
        $map->toTable('todo_items');

        $map->idToPrimaryKey('id');

        $map->property(TodoItem::DESCRIPTION)->to('description')->asVarchar(255);

        $map->property(TodoItem::COMPLETED)->to('completed')->asBool();
    }
}
```

Register the mapper in `app/AppOrm.php`:

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

namespace App;

use App\Domain\Entities\TodoItem;
use App\Infrastructure\Persistence\TodoItemMapper;
use Dms\Core\Persistence\Db\Mapping\Definition\Orm\OrmDefinition;
use Dms\Core\Persistence\Db\Mapping\Orm;
use Dms\Web\Laravel\Persistence\Db\DmsOrm;

/**
 * The application's orm.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppOrm extends Orm
{
    /**
     * Defines the object mappers registered in the orm.
     *
     * @param OrmDefinition $orm
```

```
     *
     * @return void
     */
    protected function define(OrmDefinition $orm)
    {
        $orm->enableLazyLoading();

        $orm->encompass(DmsOrm::inDefaultNamespace());

        // You can register your mappers here
        $orm->entities([
            TodoItem::class => TodoItemMapper::class
        ]);
    }
}
```

Defining the repository interface

Create the repository interface `app/Domain/Services/ITodoItemRepository.php`.

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

namespace App\Domain\Services\Persistence;

use Dms\Core\Model\ICriteria;
use Dms\Core\Model\ISpecification;
use Dms\Core\Persistence\IRepository;
use App\Domain\Entities\TodoItem;

/**
 * The repository for the App\Domain\Entities\TodoItem entity.
 */
interface ITodoItemRepository extends IRepository
{
    /**
     * {@inheritDoc}
     *
     * @return TodoItem[]
     */
    public function getAll() : array;

    /**
     * {@inheritDoc}
     *
     * @return TodoItem
     */
    public function get($id);

    /**
     * {@inheritDoc}
     *
     * @return TodoItem[]
     */
    public function getAllById(array $ids) : array;

    /**
     * {@inheritDoc}
```

```
     *
     * @return TodoItem|null
     */
    public function tryGet($id);

    /**
     * {@inheritDoc}
     *
     * @return TodoItem[]
     */
    public function tryGetAll(array $ids) : array;

    /**
     * {@inheritDoc}
     *
     * @return TodoItem[]
     */
    public function matching(ICriteria $criteria) : array;

    /**
     * {@inheritDoc}
     *
     * @return TodoItem[]
     */
    public function satisfying(ISpecification $specification) : array;
}
```

Defining the repository implementation

Create the repository implementation `app/Infrastructure/Persistence/DbTodoItemRepository.php`.

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Connection\IConnection;
use Dms\Core\Persistence\Db\Mapping\IOrm;
use Dms\Core\Persistence\DbRepository;
use App\Domain\Services\Persistence\ITodoItemRepository;
use App\Domain\Entities\TodoItem;

/**
 * The database repository implementation for the App\Domain\Entities\TodoItem entity.
 */
class DbTodoItemRepository extends DbRepository implements ITodoItemRepository
{
    public function __construct(IConnection $connection, IOrm $orm)
    {
        parent::__construct($connection, $orm->getEntityMapper(TodoItem::class));
    }

    // Custom queries can go here...
}
```

Bind your repository implementation to the interface in `app/Providers/AppServiceProvider.php`

```php
<?php

namespace App\Providers;

use App\AppCms;
use App\AppOrm;
use App\Domain\Services\Persistence\ITodoItemRepository;
use App\Infrastructure\Persistence\DbTodoItemRepository;
use Dms\Core\ICms;
use Dms\Core\Persistence\Db\Mapping\IOrm;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        //
    }

    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        $this->app->singleton(IOrm::class, AppOrm::class);
        $this->app->singleton(ICms::class, AppCms::class);

        // Bind your repositories here
        $this->app->singleton(ITodoItemRepository::class,
↪DbTodoItemRepository::class);
    }
}
```

Migrating the database

To sync the database we must generate a migration and run it:

```
# Auto-generate a migration to sync the database
php artisan dms:make:migration add_todo_items_table
# Run the migration
php artisan migrate
```

You can review the generated migration under the `database/migrations/` directory

Building the CMS

Defining your modules

To build the backend we must first configure the module `app/Cms/Modules/TodoItemModule.php`

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\ITodoItemRepository;
use App\Domain\Entities\TodoItem;
use Dms\Common\Structure\Field;

/**
 * The todo-item module.
 */
class TodoItemModule extends CrudModule
{
    public function __construct(ITodoItemRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('todo-item');

        $module->labelObjects()->fromProperty(TodoItem::DESCRIPTION);

        $module->metadata([
            'icon' => 'list', // Choose icon from http://fontawesome.io/icons/
        ]);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('description', 'Description')->string()->required()
                )->bindToProperty(TodoItem::DESCRIPTION),
                //
                $form->field(
                    Field::create('completed', 'Completed')->bool()
                )->bindToProperty(TodoItem::COMPLETED),
            ]);

        });

        $module->removeAction()->deleteFromDataSource();
```

```
        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(TodoItem::DESCRIPTION)->to(Field::create('description
↪', 'Description')->string()->required());
            $table->mapProperty(TodoItem::COMPLETED)->to(Field::create('completed',
↪'Completed')->bool());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Defining your package

Create the package (a group of modules) `app/Cms/TodoAppPackage.php`

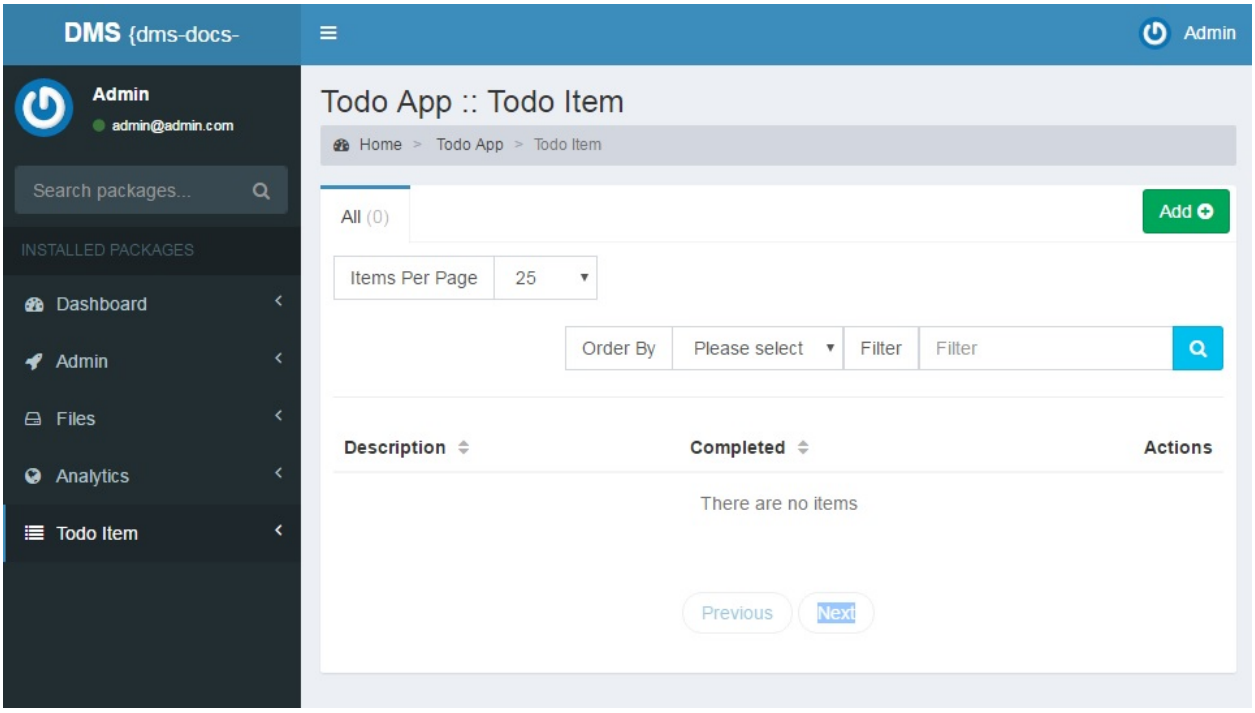
```
<?php declare(strict_types = 1);

namespace App\Cms;

use Dms\Core\Package\Definition\PackageDefinition;
use Dms\Core\Package\Package;
use App\Cms\Modules\TodoItemModule;

/**
 * The todo-app package.
 */
class TodoAppPackage extends Package
{
    /**
     * Defines the structure of this cms package.
     *
     * @param PackageDefinition $package
     *
     * @return void
     */
    protected function define(PackageDefinition $package)
    {
        $package->name('todo-app');

        $package->metadata([
            'icon' => 'check-square', // Choose icon from http://fontawesome.io/icons/
        ]);

        $package->modules([
            'todo-item' => TodoItemModule::class,
        ]);
    }
}
```

Register the package in `app/AppCms.php`

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

namespace App;
```

```php
use App\Cms\TodoAppPackage;
use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Package\Analytics\AnalyticsPackage;
use Dms\Web\Laravel\Auth\AdminPackage;
use Dms\Web\Laravel\Document\PublicFilePackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            // Default packages installed out of the box
            'admin'     => AdminPackage::class,
            'documents' => PublicFilePackage::class,
            'analytics' => AnalyticsPackage::class,

            // Register your packages here...
            'todo-app' => TodoAppPackage::class,
        ]);
    }
}
```

Visit `http://your-app-domain/dms` and login to see your first backend:



Scaffolding

For each and every project, setting up your entities, mappers, modules can feel repetitive and painful. The DMS provides powerful scaffolding commands that allow you to generate mappers for the ORM and modules for the CMS out of the box.

Scaffolding the ORM

```
php artisan dms:scaffold:persistence
```

Running this command will look for entities in the `App\Domain\Entities` namespace and auto-generate your persistence layer.

- Mapper and repository classes will be put under `app/Infrastructure/Pesistence`
- Repository interfaces will be put under `app/Domain/Services/Pesistence`

All that is left to do is register your mappers in `app/AppOrm.php`

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

namespace App;

use App\Domain\Entities\TodoItem;
use App\Infrastructure\Persistence\TodoItemMapper;
use Dms\Core\Persistence\Db\Mapping\Definition\Orm\OrmDefinition;
use Dms\Core\Persistence\Db\Mapping\Orm;
use Dms\Web\Laravel\Persistence\Db\DmsOrm;

/**
```

```
 * The application's orm.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppOrm extends Orm
{
    /**
     * Defines the object mappers registered in the orm.
     *
     * @param OrmDefinition $orm
     *
     * @return void
     */
    protected function define(OrmDefinition $orm)
    {
        $orm->enableLazyLoading();

        $orm->encompass(DmsOrm::inDefaultNamespace());

        // You can register your mappers here
        $orm->entities([
            TodoItem::class => TodoItemMapper::class
        ]);
    }
}
```

And to bind your repository interfaces to the implementations

```
<?php

namespace App\Providers;

use App\AppCms;
use App\AppOrm;
use App\Domain\Services\Persistence\ITodoItemRepository;
use App\Infrastructure\Persistence\DbTodoItemRepository;
use Dms\Core\ICms;
use Dms\Core\Persistence\Db\Mapping\IOrm;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        //
    }

    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
```

```
    {
        $this->app->singleton(IOrm::class, AppOrm::class);
        $this->app->singleton(ICms::class, AppCms::class);

        // Bind your repositories here
        $this->app->singleton(ITodoItemRepository::class,␣
↪DbTodoItemRepository::class);
    }
}
```

Generating Migrations

```
php artisan dms:make:migration your_migration_name
```

Running this command will look at the current structure of your mappers and the current database structure and generate a migration class to sync your database with the current entity structure.

- A migration will be generated under `database/migrations`

An example auto-generated migration:

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class AddTodoItemsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('todo_items', function (Blueprint $table) {
            $table->integer('id')->autoIncrement()->unsigned();
            $table->string('description', 255);
            $table->boolean('completed');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('todo_items');
    }
}
```

This command only generates the migration, you will have to run `php artisan migrate` to actually run the migration

It is recommended to review the migrations to ensure the database is synced expected

Scaffolding the CMS

```
php artisan dms:scaffold:cms your-app-name
```

Running this command will look for entities in the `App\Domain\Entities` namespace and auto-generate your CMS layer.

- Module classes will be put under `app/Cms/Modules`
- A package class will be put under `app/Cms/YouAppNamePackage.php`

All that is left to do is register your package in `app/AppCms.php`

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

namespace App;

use App\Cms\TodoAppPackage;
use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Package\Analytics\AnalyticsPackage;
use Dms\Web\Laravel\Auth\AdminPackage;
use Dms\Web\Laravel\Document\PublicFilePackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            // Default packages installed out of the box
            'admin'     => AdminPackage::class,
            'documents' => PublicFilePackage::class,
            'analytics' => AnalyticsPackage::class,

            // Register your packages here...
            'todo-app' => TodoAppPackage::class,
        ]);
    }
}
```

Repositories

Persisting entities

You can save (create or update) entities to the database using the repository

```php
<?php

function saveItem(ITodoItemRepository $repo)
{
    $repo->save(new TodoItem('some item'));

    $repo->saveAll([
        new TodoItem('an item'),
        new TodoItem('another item'),
    ]);
}
```

Loading entities

You can load entities from the database

```php
<?php

function getItem(ITodoItemRepository $repo)
{
    // Get the entity with the id = 1, throw if not found
    $entity = $repo->get(1);

    // Get the entity with the id = 1, null if not found
    $entity = $repo->tryGet(1);

    // Get the entities with the ids 1, 2 or 3, throw if any not found
    $entities = $repo->getAll([1, 2, 3]);

    // Get the entity with the ids 1, 2 or 3, ignore if any not found
    $entities = $repo->tryGetAll([1, 2, 3]);
}
```

Deleting entities

You can load entities from the database

```php
<?php

function deleteItems(ITodoItemRepository $repo)
{
    // Delete the entity
    $repo->remove($someEntity);

    // Delete the entity with the ids 1, 2 or 3
    $repo->removeById(1);

    // Delete the entity
```

```
    $repo->removeAll([$someEntity, $anotherEntity]);

    // Delete the entity with the ids 1, 2 or 3
    $repo->removeAllById([1, 2, 3]);

    // Delete all entities
    $repo->clear();
}
```

Criteria

Simple queries can be expressed through criteria, so you don't need to resort to SQL for most of your database operations.

```
<?php

function getEntitiesWithCriteria(ITodoItemRepository $repo)
{
    $items = $repo->matching(
        $repo->criteria()
            ->where(TodoItem::COMPLETED, '=', true)
    );

    $items = $repo->matching(
        $repo->criteria()
            ->whereStringContainsCaseInsensitive(TodoItem::DESCRIPTION, 'some text')
            ->orderByDesc(TodoItem::ID)
            ->skip(5)
            ->limit(10)
    );
}

function removeEntitiesWithCriteria(ITodoItemRepository $repo)
{
    $repo->removeMatching(
        $repo->criteria()
            ->whereIn(TodoItem::ID, [1, 2, 3])
    );
}
```

Custom Queries

Add methods to the repository for custom database queries.

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Connection\IConnection;
use Dms\Core\Persistence\Db\Mapping\IOrm;
use Dms\Core\Persistence\DbRepository;
use App\Domain\Services\Persistence\ITodoItemRepository;
use App\Domain\Entities\TodoItem;
```

```php
/**
 * The database repository implementation for the App\Domain\Entities\TodoItem entity.
 */
class DbTodoItemRepository extends DbRepository implements ITodoItemRepository
{
    public function __construct(IConnection $connection, IOrm $orm)
    {
        parent::__construct($connection, $orm->getEntityMapper(TodoItem::class));
    }

    /**
     * Gets five random todo items
     *
     * @return TodoItem[]
     */
    public function loadRandomItems() : array
    {
        return $this->loadQuery('
            SELECT * FROM todo_items
            ORDER BY RAND()
            LIMIT :limit
        ', ['limit' => 5]);
    }

    /**
     * Marks all todo items as complete
     *
     * @return void
     */
    public function markAllItemsAsComplete()
    {
        $this->connection->prepare('
            UPDATE todo_items
            SET completed = TRUE
        ')->execute();
    }
}
```

Field Types

There are many types of fields which you can use throughout your application, see below for examples.

Scalars

Native data types (strings, integers, floats, booleans) can be expressed as follows

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
```

```php
use Dms\Core\Model\Object\Entity;

class Person extends Entity
{
    const NAME = 'name';
    const AGE = 'age';
    const WEIGHT = 'weight';
    const HAPPY = 'happy';

    /**
     * @var string
     */
    public $name;

    /**
     * @var int
     */
    public $age;

    /**
     * @var float
     */
    public $weight;

    /**
     * @var bool
     */
    public $happy;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->age)->asInt();

        $class->property($this->weight)->asFloat();

        $class->property($this->happy)->asBool();
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Person;
```

```php
/**
 * The App\Domain\Entities\Person entity mapper.
 */
class PersonMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Person::class);
        $map->toTable('people');

        $map->idToPrimaryKey('id');

        $map->property(Person::NAME)->to('name')->asVarchar(255);

        $map->property(Person::AGE)->to('age')->asInt();

        $map->property(Person::WEIGHT)->to('weight')->asDecimal(16, 8);

        $map->property(Person::HAPPY)->to('happy')->asBool();
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IPersonRepository;
use App\Domain\Entities\Person;
use Dms\Common\Structure\Field;

/**
 * The person module.
 */
class PersonModule extends CrudModule
{
    public function __construct(IPersonRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }
```

```
    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('person');

        $module->labelObjects()->fromProperty(Person::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(Person::NAME),
                //
                $form->field(
                    Field::create('age', 'Age')->int()->required()
                )->bindToProperty(Person::AGE),
                //
                $form->field(
                    Field::create('weight', 'Weight')->decimal()->required()
                )->bindToProperty(Person::WEIGHT),
                //
                $form->field(
                    Field::create('happy', 'Happy')->bool()
                )->bindToProperty(Person::HAPPY),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Person::NAME)->to(Field::create('name', 'Name')->
↪string()->required());
            $table->mapProperty(Person::AGE)->to(Field::create('age', 'Age')->int()->
↪required());
            $table->mapProperty(Person::WEIGHT)->to(Field::create('weight', 'Weight')-
↪>decimal()->required());
            $table->mapProperty(Person::HAPPY)->to(Field::create('happy', 'Happy')->
↪bool());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Nullable

Fields can be defined as nullable which means they can possible contain a `null` value. This is useful for optional fields.

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class User extends Entity
{
    const NAME = 'name';

    /**
     * @var string|null
     */
    public $name;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->nullable()->asString();
    }

    /**
     * @return bool
     */
    public function isAnonymous() : bool
    {
        return $this->name === null;
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\User;


/**
 * The App\Domain\Entities\User entity mapper.
 */
class UserMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
```

```
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(User::class);
        $map->toTable('users');

        $map->idToPrimaryKey('id');

        $map->property(User::NAME)->to('name')->nullable()->asVarchar(255);
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IUserRepository;
use App\Domain\Entities\User;
use Dms\Common\Structure\Field;

/**
 * The user module.
 */
class UserModule extends CrudModule
{
    public function __construct(IUserRepository $dataSource, IAuthSystem $authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('user');

        $module->labelObjects()->fromProperty(User::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    // Omit ->required() to make the field optional
                    Field::create('name', 'Name')->string()
```

```
                )->bindToProperty(User::NAME),
                //
            ]);
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(User::NAME)->to(Field::create('name', 'Name')->
↪string());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Enums

Custom enums can be defined using class constants:

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\Enum;
use Dms\Core\Model\Object\PropertyTypeDefiner;

class Colour extends Enum
{
    const RED = 'red';
    const GREEN = 'green';
    const BLUE = 'blue';

    /**
     * Defines the type of the options contained within the enum.
     *
     * @param PropertyTypeDefiner $values
     *
     * @return void
     */
    protected function defineEnumValues(PropertyTypeDefiner $values)
    {
        $values->asString();
    }

    // Static factory methods...

    public static function red() : self
    {
        return new self(self::RED);
    }
```

```
    public static function green() : self
    {
        return new self(self::GREEN);
    }

    public static function blue() : self
    {
        return new self(self::BLUE);
    }
}
```

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Car extends Entity
{
    const COLOUR = 'colour';

    /**
     * @var Colour
     */
    public $colour;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->colour)->asObject(Colour::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Car;

/**
 * The App\Domain\Entities\Car entity mapper.
 */
class CarMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
```

```
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Car::class);
        $map->toTable('cars');

        $map->idToPrimaryKey('id');

        $map->enum(Car::COLOUR)->to('colour')->usingValuesFromConstants();
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\ICarRepository;
use App\Domain\Entities\Car;
use Dms\Common\Structure\Field;
use App\Domain\Entities\Colour;

/**
 * The car module.
 */
class CarModule extends CrudModule
{
    public function __construct(ICarRepository $dataSource, IAuthSystem $authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('car');

        $module->labelObjects()->fromProperty(Car::ID);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
```

```
                    Field::create('colour', 'Colour')->enum(Colour::class, [
                        Colour::RED => 'Red',
                        Colour::GREEN => 'Green',
                        Colour::BLUE => 'Blue',
                    ])->required()
                )->bindToProperty(Car::COLOUR),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Car::COLOUR)->to(Field::create('colour', 'Colour')->
↪enum(Colour::class, [
                Colour::RED => 'Red',
                Colour::GREEN => 'Green',
                Colour::BLUE => 'Blue',
            ])->required());


            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Files

Files and images can be modelled using value objects.

- `Dms\Common\Structure\FileSystem\File`
- `Dms\Common\Structure\FileSystem\Image`

These classes wrap an absolute file path which can reference a file stored on disk or externally.

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\FileSystem\File;
use Dms\Common\Structure\FileSystem\Image;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class House extends Entity
{
    const FLOORPLAN = 'floorplan';
    const PHOTO = 'photo';

    /**
```

```
     * @var File
     */
    public $floorplan;

    /**
     * @var Image
     */
    public $photo;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->floorplan)->asObject(File::class);

        $class->property($this->photo)->asObject(Image::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\House;
use Dms\Common\Structure\FileSystem\Persistence\FileMapper;
use Dms\Common\Structure\FileSystem\Persistence\ImageMapper;

/**
 * The App\Domain\Entities\House entity mapper.
 */
class HouseMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(House::class);
        $map->toTable('houses');

        $map->idToPrimaryKey('id');

        $map->embedded(House::FLOORPLAN)
            ->using(new FileMapper('floorplan', 'floorplan_file_name', public_path(
↪'app/house-files')));
```

```
        $map->embedded(House::PHOTO)
            ->using(new ImageMapper('photo', 'photo_file_name', public_path('app/
↪house-files')));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IHouseRepository;
use App\Domain\Entities\House;
use Dms\Common\Structure\Field;

/**
 * The house module.
 */
class HouseModule extends CrudModule
{
    public function __construct(IHouseRepository $dataSource, IAuthSystem $authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('house');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('floorplan', 'Floorplan')
                        ->file()
                        ->required()
                        ->moveToPathWithRandomFileName(public_path('app/house-files'))
                )->bindToProperty(House::FLOORPLAN),
                //
                $form->field(
                    Field::create('photo', 'Photo')
                        ->image()
                        ->required()
                        ->moveToPathWithRandomFileName(public_path('app/house-files'))
                )->bindToProperty(House::PHOTO),
```

```
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Date and time

Date and time fields can be modelled using value objects.

- `Dms\Common\Structure\DateTime\Date` - A date value `eg 2000-10-12`
- `Dms\Common\Structure\DateTime\TimeOfDay` - A time value `eg 06:12:56`
- `Dms\Common\Structure\DateTime\DateTime` - A date and time value `eg 2000-10-12 06:12:56`
- `Dms\Common\Structure\DateTime\TimezonedDateTime` - A date and time in a particular timezone value `eg 2000-10-12 06:12:56 Australia/Melbourne`

These classes wrap a native `DateTime` instance and provide more descriptive API for each type of date/time class.

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\DateTime\Date;
use Dms\Common\Structure\DateTime\DateTime;
use Dms\Common\Structure\DateTime\TimeOfDay;
use Dms\Common\Structure\DateTime\TimezonedDateTime;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Event extends Entity
{
    const DATE = 'date';
    const TIME = 'time';
    const DATETIME = 'dateTime';
    const TIMEZONED_DATETIME = 'timezoneDateTime';

    /**
     * @var Date
     */
    public $date;
```

```
    /**
     * @var TimeOfDay
     */
    public $time;

    /**
     * @var DateTime
     */
    public $dateTime;

    /**
     * @var TimezonedDateTime
     */
    public $timezoneDateTime;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->date)->asObject(Date::class);

        $class->property($this->time)->asObject(TimeOfDay::class);

        $class->property($this->dateTime)->asObject(DateTime::class);

        $class->property($this->timezoneDateTime)->asObject(TimezonedDateTime::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Event;
use Dms\Common\Structure\DateTime\Persistence\DateMapper;
use Dms\Common\Structure\DateTime\Persistence\TimeOfDayMapper;
use Dms\Common\Structure\DateTime\Persistence\DateTimeMapper;
use Dms\Common\Structure\DateTime\Persistence\TimezonedDateTimeMapper;

/**
 * The App\Domain\Entities\Event entity mapper.
 */
class EventMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
```

```
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Event::class);
        $map->toTable('events');

        $map->idToPrimaryKey('id');

        $map->embedded(Event::DATE)
            ->using(new DateMapper('date'));

        $map->embedded(Event::TIME)
            ->using(new TimeOfDayMapper('time'));

        $map->embedded(Event::DATETIME)
            ->using(new DateTimeMapper('date_time'));

        $map->embedded(Event::TIMEZONED_DATETIME)
            ->using(new TimezonedDateTimeMapper('timezone_date_time_date_time',
↪'timezone_date_time_timezone'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IEventRepository;
use App\Domain\Entities\Event;
use Dms\Common\Structure\Field;

/**
 * The event module.
 */
class EventModule extends CrudModule
{
    public function __construct(IEventRepository $dataSource, IAuthSystem $authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
```

```
    {
        $module->name('event');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('date', 'Date')->date()->required()
                )->bindToProperty(Event::DATE),
                //
                $form->field(
                    Field::create('time', 'Time')->time()->required()
                )->bindToProperty(Event::TIME),
                //
                $form->field(
                    Field::create('date_time', 'Date Time')->dateTime()->required()
                )->bindToProperty(Event::DATETIME),
                //
                $form->field(
                    Field::create('timezone_date_time', 'Timezone Date Time')->
↪dateTimeWithTimezone()->required()
                )->bindToProperty(Event::TIMEZONED_DATETIME),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Event::DATE)->to(Field::create('date', 'Date')->
↪date()->required());
            $table->mapProperty(Event::TIME)->to(Field::create('time', 'Time')->
↪time()->required());
            $table->mapProperty(Event::DATETIME)->to(Field::create('date_time', 'Date␣
↪Time')->dateTime()->required());
            $table->mapProperty(Event::TIMEZONED_DATETIME)->to(Field::create(
↪'timezone_date_time', 'Timezone Date Time')->dateTimeWithTimezone()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Date and time ranges

Date and time range fields can be modelled using value objects.

- `Dms\Common\Structure\DateTime\DateRange` - A date range `eg 2000-10-12 until 2002-03-05`
- `Dms\Common\Structure\DateTime\TimeRage` - A time range value `eg 06:12:56 until 14:40:23`
- `Dms\Common\Structure\DateTime\DateTimeRange` - A date and time range value `eg 2000-10-12 06:12:56 until 2002-03-05 14:40:23`

- `Dms\Common\Structure\DateTime\TimezonedDateTimeRange` - A date and time in a particular timezone range value `eg 2000-10-12 06:12:56 Australia/Melbourne until 2002-03-05 14:40:23 UTC`

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\DateTime\DateRange;
use Dms\Common\Structure\DateTime\DateTimeRange;
use Dms\Common\Structure\DateTime\TimeRange;
use Dms\Common\Structure\DateTime\TimezonedDateTimeRange;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Holiday extends Entity
{
    const DATE_RANGE = 'dateRange';
    const TIME_RANGE = 'timeRange';
    const DATETIME_RANGE = 'dateTimeRange';
    const TIMEZONED_DATETIME_RANGE = 'timezoneDateTimeRange';

    /**
     * @var DateRange
     */
    public $dateRange;

    /**
     * @var TimeRange
     */
    public $timeRange;

    /**
     * @var DateTimeRange
     */
    public $dateTimeRange;

    /**
     * @var TimezonedDateTimeRange
     */
    public $timezoneDateTimeRange;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->dateRange)->asObject(DateRange::class);

        $class->property($this->timeRange)->asObject(TimeRange::class);

        $class->property($this->dateTimeRange)->asObject(DateTimeRange::class);
```

```
        $class->property($this->timezoneDateTimeRange)->
↪asObject(TimezonedDateTimeRange::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use App\Domain\Entities\Holiday;
use Dms\Common\Structure\DateTime\Persistence\DateRangeMapper;
use Dms\Common\Structure\DateTime\Persistence\DateTimeRangeMapper;
use Dms\Common\Structure\DateTime\Persistence\TimeRangeMapper;
use Dms\Common\Structure\DateTime\Persistence\TimezonedDateTimeRangeMapper;
use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;

/**
 * The App\Domain\Entities\Holiday entity mapper.
 */
class HolidayMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Holiday::class);
        $map->toTable('holidays');

        $map->idToPrimaryKey('id');

        $map->embedded(Holiday::DATE_RANGE)
            ->using(new DateRangeMapper('start_date', 'end_date'));

        $map->embedded(Holiday::TIME_RANGE)
            ->using(new TimeRangeMapper('time_time', 'end_time'));

        $map->embedded(Holiday::DATETIME_RANGE)
            ->using(new DateTimeRangeMapper('start_date_time', 'end_date_time'));

        $map->embedded(Holiday::TIMEZONED_DATETIME_RANGE)
            ->using(new TimezonedDateTimeRangeMapper('start_date_time', 'start_
↪timezone', 'end_date_time', 'end_timezone'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use App\Domain\Entities\Holiday;
use App\Domain\Services\Persistence\IHolidayRepository;
use Dms\Common\Structure\Field;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;

/**
 * The holiday module.
 */
class HolidayModule extends CrudModule
{
    public function __construct(IHolidayRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('holidays');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('date_range', 'Date Range')->dateRange()->required()
                )->bindToProperty(Holiday::DATE_RANGE),
                //
                $form->field(
                    Field::create('time_range', 'Time Range')->timeRange()->required()
                )->bindToProperty(Holiday::TIME_RANGE),
                //
                $form->field(
                    Field::create('date_time_range', 'Date Time Range')->
↪dateTimeRange()->required()
                )->bindToProperty(Holiday::DATETIME_RANGE),
                //
                $form->field(
                    Field::create('timezoned_date_time_range', 'Timezoned Date Time␣
↪Range')->dateTimeWithTimezoneRange()->required()
                )->bindToProperty(Holiday::TIMEZONED_DATETIME_RANGE),
                //
            ]);

        });
```

```
        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Holiday::DATE_RANGE)->to(Field::create('date_range',
↪'Date Range')->dateRange()->required());
            $table->mapProperty(Holiday::TIME_RANGE)->to(Field::create('time_range',
↪'Time Range')->timeRange()->required());
            $table->mapProperty(Holiday::DATETIME_RANGE)->to(Field::create('date_time_
↪range', 'Date Time Range')->dateTimeRange()->required());
            $table->mapProperty(Holiday::TIMEZONED_DATETIME_RANGE)->to(Field::create(
↪'timezoned_date_time_range', 'Timezone Date Time Range')->
↪dateTimeWithTimezoneRange()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Location

Geographical locations can be modelled using value objects:

- `Dms\Common\Structure\Geo\LatLng` - A latitude/longitude coordinate
- `Dms\Common\Structure\Geo\StreetAddress` - A street address string
- `Dms\Common\Structure\Geo\StreetAddressWithLatLng` - A street address string and its associated lat/lng coordinates

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\Geo\LatLng;
use Dms\Common\Structure\Geo\StreetAddress;
use Dms\Common\Structure\Geo\StreetAddressWithLatLng;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Building extends Entity
{
    const ADDRESS = 'address';
    const LAT_LNG = 'latLng';
    const ADDRESS_WITH_LAT_LNG = 'addressWithLatLng';

    /**
     * @var StreetAddress
     */
    public $address;

    /**
```

```
     * @var LatLng
     */
    public $latLng;

    /**
     * @var StreetAddressWithLatLng
     */
    public $addressWithLatLng;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->address)->asObject(StreetAddress::class);

        $class->property($this->latLng)->asObject(LatLng::class);

        $class->property($this->addressWithLatLng)->
↪asObject(StreetAddressWithLatLng::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Building;
use Dms\Common\Structure\Geo\Persistence\StreetAddressMapper;
use Dms\Common\Structure\Geo\Persistence\StreetAddressWithLatLngMapper;
use Dms\Common\Structure\Geo\Persistence\LatLngMapper;

/**
 * The App\Domain\Entities\Building entity mapper.
 */
class BuildingMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Building::class);
        $map->toTable('buildings');

        $map->idToPrimaryKey('id');
```

```
        $map->embedded(Building::ADDRESS)
            ->using(new StreetAddressMapper('address'));

        $map->embedded(Building::LAT_LNG)
            ->using(new LatLngMapper('lat', 'lng'));

        $map->embedded(Building::ADDRESS_WITH_LAT_LNG)
            ->using(new StreetAddressWithLatLngMapper('address1', 'lat1', 'lng1'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IBuildingRepository;
use App\Domain\Entities\Building;
use Dms\Common\Structure\Field;

/**
 * The building module.
 */
class BuildingModule extends CrudModule
{
    public function __construct(IBuildingRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('building');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('address', 'Address')->streetAddress()->required()
                )->bindToProperty(Building::ADDRESS),
                //
                $form->field(
                    Field::create('lat_lng', 'Lat Lng')->latLng()->required()
                )->bindToProperty(Building::LAT_LNG),
```

```
                //
                $form->field(
                    Field::create('address_with_lat_lng', 'Address With Lat Lng')->
↪streetAddressWithLatLng()->required()
                )->bindToProperty(Building::ADDRESS_WITH_LAT_LNG),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Building::ADDRESS)->to(Field::create('address',
↪'Address')->streetAddress()->required());
            $table->mapProperty(Building::LAT_LNG)->to(Field::create('lat_lng', 'Lat␣
↪Lng')->latLng()->required());
            $table->mapProperty(Building::ADDRESS_WITH_LAT_LNG)->to(Field::create(
↪'address_with_lat_lng', 'Address With Lat Lng')->streetAddressWithLatLng()->
↪required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Web

Miscellaneous web-related fields can be modelled using the value objects:

- `Dms\Common\Structure\Web\EmailAddress` - An email address `john@gmail.com`
- `Dms\Common\Structure\Web\IpAddress` - An IP address `123.123.123.123`
- `Dms\Common\Structure\Web\Url` - An absolute url `https://www.google.com`
- `Dms\Common\Structure\Web\Html` - A raw HTML string `<p>abc</p>`

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\Web\EmailAddress;
use Dms\Common\Structure\Web\Html;
use Dms\Common\Structure\Web\IpAddress;
use Dms\Common\Structure\Web\Url;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Business extends Entity
{
    const EMAIL_ADDRESS = 'emailAddress';
```

```
    const IP_ADDRESS = 'ipAddress';
    const WEBSITE = 'website';
    const DESCRIPTION = 'description';

    /**
     * @var EmailAddress
     */
    public $emailAddress;

    /**
     * @var IpAddress
     */
    public $ipAddress;

    /**
     * @var Url
     */
    public $website;

    /**
     * @var Html
     */
    public $description;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->emailAddress)->asObject(EmailAddress::class);

        $class->property($this->ipAddress)->asObject(IpAddress::class);

        $class->property($this->website)->asObject(Url::class);

        $class->property($this->description)->asObject(Html::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Business;
use Dms\Common\Structure\Web\Persistence\EmailAddressMapper;
use Dms\Common\Structure\Web\Persistence\IpAddressMapper;
use Dms\Common\Structure\Web\Persistence\UrlMapper;
use Dms\Common\Structure\Web\Persistence\HtmlMapper;

/**
```

```
 * The App\Domain\Entities\Business entity mapper.
 */
class BusinessMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Business::class);
        $map->toTable('businesses');

        $map->idToPrimaryKey('id');

        $map->embedded(Business::EMAIL_ADDRESS)
            ->using(new EmailAddressMapper('email_address'));

        $map->embedded(Business::IP_ADDRESS)
            ->using(new IpAddressMapper('ip_address'));

        $map->embedded(Business::WEBSITE)
            ->using(new UrlMapper('website'));

        $map->embedded(Business::DESCRIPTION)
            ->using(new HtmlMapper('description'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IBusinessRepository;
use App\Domain\Entities\Business;
use Dms\Common\Structure\Field;

/**
 * The business module.
 */
class BusinessModule extends CrudModule
{
    public function __construct(IBusinessRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
```

```
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('business');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('email_address', 'Email Address')->email()->
↪required()
                )->bindToProperty(Business::EMAIL_ADDRESS),
                //
                $form->field(
                    Field::create('ip_address', 'Ip Address')->ipAddress()->required()
                )->bindToProperty(Business::IP_ADDRESS),
                //
                $form->field(
                    Field::create('website', 'Website')->url()->required()
                )->bindToProperty(Business::WEBSITE),
                //
                $form->field(
                    Field::create('description', 'Description')->html()->required()
                )->bindToProperty(Business::DESCRIPTION),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Business::EMAIL_ADDRESS)->to(Field::create('email_
↪address', 'Email Address')->email()->required());
            $table->mapProperty(Business::IP_ADDRESS)->to(Field::create('ip_address',
↪'Ip Address')->ipAddress()->required());
            $table->mapProperty(Business::WEBSITE)->to(Field::create('website',
↪'Website')->url()->required());
            $table->mapProperty(Business::DESCRIPTION)->to(Field::create('description
↪', 'Description')->html()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Colours

Colours can be modelled using value objects:

- `Dms\Common\Structure\Colour\Colour` - A RGB colour value `rgb(123, 123, 123)`
- `Dms\Common\Structure\Colour\TransparentColour` - A RGBA colour value `rgba(123, 123, 123, 0.5)`

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\Colour\Colour;
use Dms\Common\Structure\Colour\TransparentColour;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Photograph extends Entity
{
    const BACKGROUND = 'background';
    const FOREGROUND = 'foreground';

    /**
     * @var Colour
     */
    public $background;

    /**
     * @var TransparentColour
     */
    public $foreground;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->background)->asObject(Colour::class);

        $class->property($this->foreground)->asObject(TransparentColour::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Photograph;
use Dms\Common\Structure\Colour\Mapper\ColourMapper;
use Dms\Common\Structure\Colour\Mapper\TransparentColourMapper;
```

```php
/**
 * The App\Domain\Entities\Photograph entity mapper.
 */
class PhotographMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Photograph::class);
        $map->toTable('photographs');

        $map->idToPrimaryKey('id');

        $map->embedded(Photograph::BACKGROUND)
            ->using(ColourMapper::asHexString('background'));

        $map->embedded(Photograph::FOREGROUND)
            ->using(TransparentColourMapper::asRgbaString('foreground'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IPhotographRepository;
use App\Domain\Entities\Photograph;
use Dms\Common\Structure\Field;

/**
 * The photograph module.
 */
class PhotographModule extends CrudModule
{
    public function __construct(IPhotographRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
```

```
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('photograph');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('background', 'Background')->colour()->required()
                )->bindToProperty(Photograph::BACKGROUND),
                //
                $form->field(
                    Field::create('foreground', 'Foreground')->
↪colourWithTransparency()->required()
                )->bindToProperty(Photograph::FOREGROUND),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Photograph::BACKGROUND)->to(Field::create('background
↪', 'Background')->colour()->required());
            $table->mapProperty(Photograph::FOREGROUND)->to(Field::create('foreground
↪', 'Foreground')->colourWithTransparency()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Money

Monetary values can be modelled using a value object:

- `Dms\Common\Structure\Money\Money` - Represents an amount of currency `$123.45 AUD`

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\Money\Money;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Product extends Entity
{
    const PRICE = 'price';
```

```
    /**
     * @var Money
     */
    public $price;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->price)->asObject(Money::class);
    }
}
```

Mapper Configuration

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Product;
use Dms\Common\Structure\Money\Persistence\MoneyMapper;

/**
 * The App\Domain\Entities\Product entity mapper.
 */
class ProductMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Product::class);
        $map->toTable('products');

        $map->idToPrimaryKey('id');

        $map->embedded(Product::PRICE)
            ->using(new MoneyMapper('price_amount', 'price_currency'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IProductRepository;
use App\Domain\Entities\Product;
use Dms\Common\Structure\Field;

/**
 * The product module.
 */
class ProductModule extends CrudModule
{
    public function __construct(IProductRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('product');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('price', 'Price')->money()->required()
                )->bindToProperty(Product::PRICE),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Product::PRICE)->to(Field::create('price', 'Price')->
↪money()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Arrays/Collections

It is often useful to represent an array of values.

For scalar values you can use a native `array`.

For a collection of value objects it is recommended you use `Dms\Core\Model\ValueObjectCollection`.

For a collection of entities *see relations*.

Entity Structure

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

namespace App\Domain\Entities;

use Dms\Common\Structure\Web\EmailAddress;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;
use Dms\Core\Model\Type\Builder\Type;
use Dms\Core\Model\ValueObjectCollection;

class Contact extends Entity
{
    const NAMES = 'NAMES';
    const EMAIL_ADDRESSES = 'emailAddresses';

    /**
     * @var string[]
     */
    public $names;

    /**
     * @var ValueObjectCollection|EmailAddress[]
     */
    public $emailAddresses;


    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->names)->asArrayOf(Type::string());

        $class->property($this->emailAddresses)->
↪asType(EmailAddress::collectionType());
    }
}
```

Mapper Configuration

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

```php
namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Contact;
use Dms\Common\Structure\Web\Persistence\EmailAddressMapper;

/**
 * The App\Domain\Entities\Contact entity mapper.
 */
class ContactMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Contact::class);
        $map->toTable('contacts');

        $map->idToPrimaryKey('id');

        // Map names array to JSON string in 'names' column
        $map->property(Contact::NAMES)
            ->mappedVia(
                function (array $names) {
                    return json_encode($names);
                },
                function (string $json) {
                    return (array)json_decode($json);
                }
            )
            ->to('names')
            ->asVarchar(8000);

        // Map email addresses to separate table 'contact_email_addresses'
        $map->embeddedCollection(Contact::EMAIL_ADDRESSES)
            ->toTable('contact_email_addresses')
            ->withPrimaryKey('id')
            ->withForeignKeyToParentAs('contact_id')
            ->using(new EmailAddressMapper('email_addresses'));
    }
}
```

Module Configuration

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
```

```php
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IContactRepository;
use App\Domain\Entities\Contact;
use Dms\Common\Structure\Field;

/**
 * The contact module.
 */
class ContactModule extends CrudModule
{
    public function __construct(IContactRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('contact');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('names', 'Names')->arrayOf(
                        Field::element()->string()->required()
                    )
                )->bindToProperty('names'),
                //
                $form->field(
                    Field::create('email_addresses', 'Email Addresses')->arrayOf(
                        Field::element()->email()->required()
                    )
                )->bindToProperty(Contact::EMAIL_ADDRESSES),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Custom Value Objects

You may find yourself repeating a bunch of fields throughout your application, in this case you can extract these fields into your own custom value objects which can then be reused throughout your app.

Entity Structure

app/Domain/Entities/Engine.php

Your custom value object class

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\ValueObject;

class Engine extends ValueObject
{
    const NAME = 'name';
    const HORSE_POWER = 'horsePower';

    /**
     * @var string
     */
    public $name;

    /**
     * @var int
     */
    public $horsePower;

    /**
     * Defines the structure of this class.
     *
     * @param ClassDefinition $class
     */
    protected function define(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->horsePower)->asInt();
    }
}
```

app/Domain/Entities/Vehicle.php

An entity which contains your value object as a field.

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
```

```php
use Dms\Core\Model\Object\Entity;

class Vehicle extends Entity
{
    const ENGINE = 'engine';

    /**
     * @var Engine
     */
    public $engine;

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->engine)->asObject(Engine::class);
    }
}
```

Mapper Configuration

app/Infrastructure/Persistence/EngineMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\IndependentValueObjectMapper;
use App\Domain\Entities\Engine;

/**
 * The App\Domain\Entities\Engine value object mapper.
 */
class EngineMapper extends IndependentValueObjectMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Engine::class);

        $map->property(Engine::NAME)->to('name')->asVarchar(255);

        $map->property(Engine::HORSE_POWER)->to('horse_power')->asInt();
    }
}
```

app/Infrastructure/Persistence/VehicleMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Vehicle;
use App\Infrastructure\Persistence\EngineMapper;

/**
 * The App\Domain\Entities\Vehicle entity mapper.
 */
class VehicleMapper extends EntityMapper
{
    /**
     * Defines the value object mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Vehicle::class);
        $map->toTable('vehicles');

        $map->idToPrimaryKey('id');

        // Embeds the columns of the value object in the 'vehicles' table
        // Each column will be prefixed so they will become 'engine_name' and 'engine_
↪horse_power'
        $map->embedded(Vehicle::ENGINE)
            ->withColumnsPrefixedBy('engine_')
            ->using(new EngineMapper());
    }
}
```

Module Configuration

app/Cms/Modules/Fields/EngineField.php

You can define custom fields for value objects

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

namespace App\Cms\Modules\Fields;

use Dms\Core\Common\Crud\Definition\Form\ValueObjectFieldDefinition;
use Dms\Core\Common\Crud\Form\ValueObjectField;
use App\Domain\Entities\Engine;
use Dms\Common\Structure\Field;

/**
 * The App\Domain\Entities\Engine value object field.
```

```
 */
class EngineField extends ValueObjectField
{
    public function __construct(string $name, string $label)
    {
        parent::__construct($name, $label);
    }

    /**
     * Defines the structure of this value object field.
     *
     * @param ValueObjectFieldDefinition $form
     *
     * @return void
     */
    protected function define(ValueObjectFieldDefinition $form)
    {
        $form->bindTo(Engine::class);

        $form->section('Details', [
            $form->field(
                Field::create('name', 'Name')->string()->required()
            )->bindToProperty(Engine::NAME),
            //
            $form->field(
                Field::create('horse_power', 'Horse Power')->int()->required()
            )->bindToProperty(Engine::HORSE_POWER),
            //
        ]);
    }
}
```

app/Cms/Modules/VehicleModule.php

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IVehicleRepository;
use App\Domain\Entities\Vehicle;
use App\Cms\Modules\Fields\EngineField;

/**
 * The vehicle module.
 */
class VehicleModule extends CrudModule
{
    public function __construct(IVehicleRepository $dataSource, IAuthSystem
↪$authSystem)
    {
        parent::__construct($dataSource, $authSystem);
```

```
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('vehicle');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    // Reference your custom value object field
                    (new EngineField('engine', 'Engine'))->required()
                )->bindToProperty(Vehicle::ENGINE),
                //
            ]);
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Vehicle::ENGINE)->to((new EngineField('engine',
↪'Engine'))->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Relations

You can define different types of relationships between entities, see examples below

One to One

An one to one bi-directional relationship between entities can be defined as follows.

Entity Structure

app/Domain/Entities/Country.php

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;
```

```php
class Country extends Entity
{
    const NAME = 'name';
    const CAPITAL_CITY = 'capitalCity';

    /**
     * @var string
     */
    public $name;

    /**
     * @var CapitalCity
     */
    public $capitalCity;

    /**
     * Country constructor.
     *
     * @param string      $name
     * @param CapitalCity $capitalCity
     */
    public function __construct(string $name, CapitalCity $capitalCity)
    {
        parent::__construct();
        $this->name           = $name;
        $this->capitalCity    = $capitalCity;
        $capitalCity->country = $this;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->capitalCity)->asObject(CapitalCity::class);
    }
}
```

app/Domain/Entities/CapitalCity.php

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class CapitalCity extends Entity
{
    const NAME = 'name';
    const COUNTRY = 'country';
```

```
    /**
     * @var string
     */
    public $name;

    /**
     * @var Country
     */
    public $country;

    /**
     * CapitalCity constructor.
     *
     * @param string  $name
     * @param Country $country
     */
    public function __construct(string $name, Country $country)
    {
        parent::__construct();
        $this->name           = $name;
        $this->country        = $country;
        $country->capitalCity = $this;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->country)->asObject(Country::class);
    }
}
```

Mapper Configuration

app/Infrastructure/Persistence/CountryMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Country;
use App\Domain\Entities\CapitalCity;

/**
 * The App\Domain\Entities\Country entity mapper.
 */
class CountryMapper extends EntityMapper
{
```

```
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Country::class);
        $map->toTable('countries');

        $map->idToPrimaryKey('id');

        $map->property(Country::NAME)->to('name')->asVarchar(255);

        $map->column('capital_city_id')->asUnsignedInt();
        $map->relation(Country::CAPITAL_CITY)
            ->to(CapitalCity::class)
            ->manyToOne()
            ->withBidirectionalRelation(CapitalCity::COUNTRY)
            ->withRelatedIdAs('capital_city_id');
    }
}
```

app/Infrastructure/Persistence/CapitalCityMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\CapitalCity;
use App\Domain\Entities\Country;

/**
 * The App\Domain\Entities\CapitalCity entity mapper.
 */
class CapitalCityMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(CapitalCity::class);
        $map->toTable('capital_cities');

        $map->idToPrimaryKey('id');

        $map->property(CapitalCity::NAME)->to('name')->asVarchar(255);
```

```
        $map->relation(CapitalCity::COUNTRY)
            ->to(Country::class)
            ->toOne()
            ->identifying()
            ->withBidirectionalRelation(Country::CAPITAL_CITY)
            ->withParentIdAs('capital_city_id');
    }
}
```

Module Configuration

app/Cms/Modules/CountryModule.php

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

namespace App\Cms\Modules;

use App\Domain\Entities\CapitalCity;
use App\Domain\Entities\Country;
use App\Domain\Services\Persistence\ICapitalCityRepository;
use App\Domain\Services\Persistence\ICountryRepository;
use Dms\Common\Structure\Field;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;

/**
 * The country module.
 */
class CountryModule extends CrudModule
{
    /**
     * @var ICapitalCityRepository
     */
    protected $capitalCityRepository;

    public function __construct(ICountryRepository $dataSource, IAuthSystem
↪$authSystem, ICapitalCityRepository $capitalCityRepository)
    {
        $this->capitalCityRepository = $capitalCityRepository;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('country');
```

```
        $module->labelObjects()->fromProperty(Country::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(Country::NAME),
                //
                //
            ]);

            if ($form->isCreateForm()) {
                $form->continueSection([
                    $form->field(
                        Field::create('capital_city', 'Capital City')->string()->
↪required()
                    )->bindToCallbacks(
                        function () {
                            // Unused
                        },
                        function (Country $country, string $cityName) {
                            $country->capitalCity = new CapitalCity($cityName,
↪$country);
                        }
                    ),
                ]);
            } else {
                $form->continueSection([
                    $form->field(
                        Field::create('capital_city', 'Capital City')
                            ->entityFrom($this->capitalCityRepository)
                            ->required()
                            ->labelledBy(CapitalCity::NAME)
                            // Add this line if you want to load the
                            // options asynchronously via autocomplete
                            ->searchableBy(Country::NAME)
                    )->bindToProperty(Country::CAPITAL_CITY),
                ]);
            }
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Country::NAME)->to(Field::create('name', 'Name')->
↪string()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

app/Cms/Modules/CapitalCityModule.php

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

namespace App\Cms\Modules;

use App\Domain\Entities\CapitalCity;
use App\Domain\Entities\Country;
use App\Domain\Services\Persistence\ICapitalCityRepository;
use App\Domain\Services\Persistence\ICountryRepository;
use Dms\Common\Structure\Field;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;

/**
 * The capital-city module.
 */
class CapitalCityModule extends CrudModule
{
    /**
     * @var ICountryRepository
     */
    protected $countryRepository;

    public function __construct(ICapitalCityRepository $dataSource, IAuthSystem
↪$authSystem, ICountryRepository $countryRepository)
    {
        $this->countryRepository = $countryRepository;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('capital-city');

        $module->labelObjects()->fromProperty(CapitalCity::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(CapitalCity::NAME),
            ]);

            if ($form->isCreateForm()) {
                $form->continueSection([
                    $form->field(
                        Field::create('country', 'Country')->string()->required()
                    )->bindToCallbacks(
                        function () {
```

```
                            // Unused
                        },
                        function (CapitalCity $capitalCity, string $countryName) {
                            $capitalCity->country = new Country($countryName,
↪$capitalCity);
                        }
                    )
                ]);
            } else {
                $form->continueSection([
                    $form->field(
                        Field::create('country', 'Country')
                            ->entityFrom($this->countryRepository)
                            ->required()
                            ->labelledBy(Country::NAME)
                            // Add this line if you want to load the
                            // options asynchronously via autocomplete
                            ->searchableBy(Country::NAME)
                    )->bindToProperty(CapitalCity::COUNTRY),
                ]);
            }
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(CapitalCity::NAME)->to(Field::create('name', 'Name')->
↪string()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

One to Many

An one to many bi-directional relationship between entities can be defined as follows.

Entity Structure

app/Domain/Entities/Mother.php

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

namespace App\Domain\Entities;

use Dms\Core\Model\EntityCollection;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Mother extends Entity
{
```

```
    const NAME = 'name';
    const CHILDREN = 'children';

    /**
     * @var string
     */
    public $name;

    /**
     * @var EntityCollection|Child[]
     */
    public $children;

    /**
     * Parent constructor.
     */
    public function __construct()
    {
        parent::__construct();
        $this->children = Child::collection();
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->children)->asType(Child::collectionType());
    }
}
```

app/Domain/Entities/Child.php

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

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Child extends Entity
{
    const NAME = 'name';
    const MOTHER = 'mother';

    /**
     * @var string
     */
    public $name;

    /**
     * @var Mother
```

```
     */
    public $mother;

    /**
     * Child constructor.
     *
     * @param string $name
     * @param Mother $mother
     */
    public function __construct($name, Mother $mother)
    {
        parent::__construct();
        $this->name   = $name;
        $this->mother = $mother;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->mother)->asObject(Mother::class);
    }
}
```

Mapper Configuration

app/Infrastructure/Persistence/MotherMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Mother;
use App\Domain\Entities\Child;

/**
 * The App\Domain\Entities\Mother entity mapper.
 */
class MotherMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
```

```
    {
        $map->type(Mother::class);
        $map->toTable('mothers');

        $map->idToPrimaryKey('id');

        $map->property(Mother::NAME)->to('name')->asVarchar(255);

        $map->relation(Mother::CHILDREN)
            ->to(Child::class)
            ->toMany()
            ->identifying()
            ->withBidirectionalRelation(Child::MOTHER)
            ->withParentIdAs('mother_id');
    }
}
```

app/Infrastructure/Persistence/ChildMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Child;
use App\Domain\Entities\Mother;

/**
 * The App\Domain\Entities\Child entity mapper.
 */
class ChildMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Child::class);
        $map->toTable('children');

        $map->idToPrimaryKey('id');

        $map->property(Child::NAME)->to('name')->asVarchar(255);

        $map->column('mother_id')->asUnsignedInt();
        $map->relation(Child::MOTHER)
            ->to(Mother::class)
            ->manyToOne()
            ->withBidirectionalRelation(Mother::CHILDREN)
            ->withRelatedIdAs('mother_id');
    }
```

```
}
```

Module Configuration

app/Cms/Modules/MotherModule.php

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IMotherRepository;
use App\Domain\Entities\Mother;
use Dms\Common\Structure\Field;
use App\Domain\Services\Persistence\IChildRepository;
use App\Domain\Entities\Child;

/**
 * The mother module.
 */
class MotherModule extends CrudModule
{
    /**
     * @var IChildRepository
     */
    protected $childRepository;

    public function __construct(IMotherRepository $dataSource, IAuthSystem
↪$authSystem, IChildRepository $childRepository)
    {
        $this->childRepository = $childRepository;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('mother');

        $module->labelObjects()->fromProperty(Mother::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(Mother::NAME),
                //
```

```
                $form->field(
                    Field::create('children', 'Children')
                        ->entitiesFrom($this->childRepository)
                        ->labelledBy(Child::NAME)
                        ->mapToCollection(Child::collectionType())
                        // Add this line if you want to load the
                        // options asynchronously via autocomplete
                        ->searchableBy(Child::NAME)
                )->bindToProperty(Mother::CHILDREN),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Mother::NAME)->to(Field::create('name', 'Name')->
↪string()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

app/Cms/Modules/ChildModule.php

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IChildRepository;
use App\Domain\Entities\Child;
use Dms\Common\Structure\Field;
use App\Domain\Services\Persistence\IMotherRepository;
use App\Domain\Entities\Mother;

/**
 * The child module.
 */
class ChildModule extends CrudModule
{
    /**
     * @var IMotherRepository
     */
    protected $motherRepository;


    public function __construct(IChildRepository $dataSource, IAuthSystem $authSystem,
↪ IMotherRepository $motherRepository)
```

```
    {
        $this->motherRepository = $motherRepository;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('child');

        $module->labelObjects()->fromProperty(Child::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(Child::NAME),
                //
                $form->field(
                    Field::create('mother', 'Mother')
                        ->entityFrom($this->motherRepository)
                        ->required()
                        ->labelledBy(Mother::NAME)
                )->bindToProperty(Child::MOTHER),
            ]);
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Child::NAME)->to(Field::create('name', 'Name')->
↪string()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Many to Many

A many to many bi-directional relationship between entities can be defined as follows.

Entity Structure

app/Domain/Entities/Article.php

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

```php
namespace App\Domain\Entities;

use Dms\Core\Model\EntityCollection;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Article extends Entity
{
    const TITLE = 'title';
    const TAGS = 'tags';

    /**
     * @var string
     */
    public $title;

    /**
     * @var EntityCollection|Tag[]
     */
    public $tags;

    /**
     * Article constructor.
     */
    public function __construct()
    {
        parent::__construct();
        $this->tags = Tag::collection();
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->title)->asString();

        $class->property($this->tags)->asType(Tag::collectionType());
    }
}
```

app/Domain/Entities/Tag.php

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

namespace App\Domain\Entities;

use Dms\Core\Model\EntityCollection;
use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

class Tag extends Entity
{
    const NAME = 'name';
```

```
    const ARTICLES = 'articles';

    /**
     * @var string
     */
    public $name;

    /**
     * @var EntityCollection|Article[]
     */
    public $articles;

    /**
     * Tag constructor.
     */
    public function __construct()
    {
        parent::__construct();
        $this->articles = Article::collection();
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();

        $class->property($this->articles)->asType(Article::collectionType());
    }
}
```

Mapper Configuration

app/Infrastructure/Persistence/ArticleMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Article;
use App\Domain\Entities\Tag;

/**
 * The App\Domain\Entities\Article entity mapper.
 */
class ArticleMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
```

```
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Article::class);
        $map->toTable('articles');

        $map->idToPrimaryKey('id');

        $map->property(Article::TITLE)->to('title')->asVarchar(255);

        $map->relation(Article::TAGS)
            ->to(Tag::class)
            ->toMany()
            ->withBidirectionalRelation(Tag::ARTICLES)
            ->throughJoinTable('article_tags')
            ->withParentIdAs('article_id')
            ->withRelatedIdAs('tag_id');
    }
}
```

app/Infrastructure/Persistence/TagMapper.php

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

namespace App\Infrastructure\Persistence;

use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;
use App\Domain\Entities\Tag;
use App\Domain\Entities\Article;

/**
 * The App\Domain\Entities\Tag entity mapper.
 */
class TagMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Tag::class);
        $map->toTable('tags');

        $map->idToPrimaryKey('id');

        $map->property(Tag::NAME)->to('name')->asVarchar(255);

        $map->relation(Tag::ARTICLES)
```

```
            ->to(Article::class)
            ->toMany()
            ->withBidirectionalRelation(Article::TAGS)
            ->throughJoinTable('article_tags')
            ->withParentIdAs('tag_id')
            ->withRelatedIdAs('article_id');
    }
}
```

Module Configuration

app/Cms/Modules/ArticleModule.php

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IArticleRepository;
use App\Domain\Entities\Article;
use Dms\Common\Structure\Field;
use App\Domain\Services\Persistence\ITagRepository;
use App\Domain\Entities\Tag;

/**
 * The article module.
 */
class ArticleModule extends CrudModule
{
    /**
     * @var ITagRepository
     */
    protected $tagRepository;

    public function __construct(IArticleRepository $dataSource, IAuthSystem
↪$authSystem, ITagRepository $tagRepository)
    {
        $this->tagRepository = $tagRepository;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('article');

        $module->labelObjects()->fromProperty(Article::TITLE);
```

```
        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('title', 'Title')->string()->required()
                )->bindToProperty(Article::TITLE),
                //
                $form->field(
                    Field::create('tags', 'Tags')
                        ->entitiesFrom($this->tagRepository)
                        ->labelledBy(Tag::NAME)
                        ->mapToCollection(Tag::collectionType())
                        // Add this line if you want to load the
                        // options asynchronously via autocomplete
                        ->searchableBy(Tag::NAME)
                )->bindToProperty(Article::TAGS),
            ]);
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Article::TITLE)->to(Field::create('title', 'Title')->
↪string()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

app/Cms/Modules/TagModule.php

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\ITagRepository;
use App\Domain\Entities\Tag;
use Dms\Common\Structure\Field;
use App\Domain\Services\Persistence\IArticleRepository;
use App\Domain\Entities\Article;

/**
 * The tag module.
 */
class TagModule extends CrudModule
{
    /**
     * @var IArticleRepository
```

```
     */
    protected $articleRepository;

    public function __construct(ITagRepository $dataSource, IAuthSystem $authSystem,␣
↪IArticleRepository $articleRepository)
    {
        $this->articleRepository = $articleRepository;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('tag');

        $module->labelObjects()->fromProperty(Tag::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(Tag::NAME),
                //
                $form->field(
                    Field::create('articles', 'Articles')
                        ->entitiesFrom($this->articleRepository)
                        ->labelledBy(Article::TITLE)
                        ->mapToCollection(Article::collectionType())
                        // Add this line if you want to load the
                        // options asynchronously via autocomplete
                        ->searchableBy(Article::TITLE)
                )->bindToProperty(Tag::ARTICLES),
                //
            ]);

        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Tag::NAME)->to(Field::create('name', 'Name')->
↪string()->required());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Advanced Topics

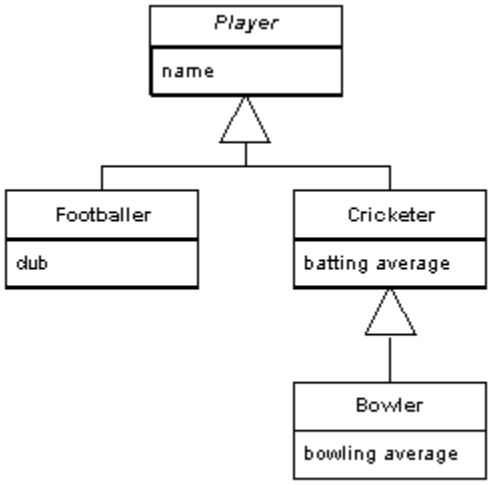
Advanced Topics

Subclasses

A powerful of feature of OOP to help remove duplication is subclassing. Subclassing allows you to create inheritance trees to better model your domain.

Entity Structure

We'll model this example from the following diagram:



app/Domain/Entities/Player.php

```
<?php

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;
use Dms\Core\Model\Object\Entity;

abstract class Player extends Entity
{
    const NAME = 'name';

    /**
     * @var string
     */
    public $name;

    public function __construct(string $name)
    {
```

```
        parent::__construct();
        $this->name = $name;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        $class->property($this->name)->asString();
    }
}
```

app/Domain/Entities/Footballer.php

```
<?php

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;

class Footballer extends Player
{
    const CLUB = 'club';

    /**
     * @var string
     */
    public $club;

    public function __construct(string $name, string $club)
    {
        parent::__construct($name);
        $this->club = $club;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        parent::defineEntity($class);

        $class->property($this->club)->asString();
    }
}
```

app/Domain/Entities/Cricketer.php

```php
<?php

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;

class Cricketer extends Player
{
    const BATTING_AVERAGE = 'battingAverage';

    /**
     * @var int
     */
    public $battingAverage;

    public function __construct(string $name, int $battingAverage)
    {
        parent::__construct($name);
        $this->battingAverage = $battingAverage;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        parent::defineEntity($class);

        $class->property($this->battingAverage)->asInt();
    }
}
```

app/Domain/Entities/Bowler.php

```php
<?php

namespace App\Domain\Entities;

use Dms\Core\Model\Object\ClassDefinition;

class Bowler extends Cricketer
{
    const BOWLING_AVERAIGE = 'bowlingAverage';

    /**
     * @var int
     */
    public $bowlingAverage;

    public function __construct(string $name, int $battingAverage, int
↪$bowlingAverage)
    {
```

```
        parent::__construct($name, $battingAverage);
        $this->bowlingAverage = $bowlingAverage;
    }

    /**
     * Defines the structure of this entity.
     *
     * @param ClassDefinition $class
     */
    protected function defineEntity(ClassDefinition $class)
    {
        parent::defineEntity($class);

        $class->property($this->bowlingAverage)->asInt();
    }
}
```

Mapper Configuration (Single Table Inheritance)

You can map all the subclasses to one database table using single table inheritance pattern with a type column.

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

namespace App\Infrastructure\Persistence;

use App\Domain\Entities\Bowler;
use App\Domain\Entities\Cricketer;
use App\Domain\Entities\Footballer;
use App\Domain\Entities\Player;
use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;

/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class PlayerMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Player::class);
        $map->toTable('players');

        $map->idToPrimaryKey('id');

        $map->column('type')->asEnum(['footballer', 'cricketer', 'bowler']);
        $map->property(Player::NAME)->to('name')->asVarchar(255);

        $map->subclass()->withTypeInColumn('type', 'footballer')->define(function
↪(MapperDefinition $map) {
```

```
            $map->type(Footballer::class);
            $map->property(Footballer::CLUB)->to('club')->asVarchar(255);
        });

        $map->subclass()->withTypeInColumn('type', 'cricketer')->define(function␣
↪(MapperDefinition $map) {
            $map->type(Cricketer::class);
            $map->property(Cricketer::BATTING_AVERAGE)->to('batting_average')->
↪asInt();

            $map->subclass()->withTypeInColumn('type', 'bowler')->define(function␣
↪(MapperDefinition $map) {
                $map->type(Bowler::class);
                $map->property(Bowler::BOWLING_AVERAGE)->to('bowling_average')->
↪asInt();
            });
        });
    }
}
```

Mapper Configuration (Class Table Inheritance)

Alternatively, you can map each the subclasses to a separate table.

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

namespace App\Infrastructure\Persistence;

use App\Domain\Entities\Bowler;
use App\Domain\Entities\Cricketer;
use App\Domain\Entities\Footballer;
use App\Domain\Entities\Player;
use Dms\Core\Persistence\Db\Mapping\Definition\MapperDefinition;
use Dms\Core\Persistence\Db\Mapping\EntityMapper;

/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class PlayerMapper extends EntityMapper
{
    /**
     * Defines the entity mapper
     *
     * @param MapperDefinition $map
     *
     * @return void
     */
    protected function define(MapperDefinition $map)
    {
        $map->type(Player::class);
        $map->toTable('players');

        $map->idToPrimaryKey('id');
        $map->property(Player::NAME)->to('name')->asVarchar(255);

        $map->subclass()->asSeparateTable('footballers')->define(function␣
↪(MapperDefinition $map) {
```

```
            $map->type(Footballer::class);
            $map->property(Footballer::CLUB)->to('club')->asVarchar(255);
        });

        $map->subclass()->asSeparateTable('cricketers')->define(function␣
↪(MapperDefinition $map) {
            $map->type(Cricketer::class);
            $map->property(Cricketer::BATTING_AVERAGE)->to('batting_average')->
↪asInt();

            $map->subclass()->asSeparateTable('bowlers')->define(function␣
↪(MapperDefinition $map) {
                $map->type(Bowler::class);
                $map->property(Bowler::BOWLING_AVERAGE)->to('bowling_average')->
↪asInt();
            });
        });
    }
}
```

Module Configuration

Modules support mapping to entity subclasses as shown in the following example

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

namespace App\Cms\Modules;

use App\Domain\Entities\Bowler;
use App\Domain\Entities\Cricketer;
use App\Domain\Entities\Footballer;
use App\Domain\Entities\Player;
use Dms\Common\Structure\Field;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;

/**
 * The player module.
 */
class PlayerModule extends CrudModule
{
    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('player');

        $module->labelObjects()->fromProperty(Player::NAME);

        $module->crudForm(function (CrudFormDefinition $form) {
```

```
            $typeField = Field::create('type', 'Type')->string()->oneOf([
                'footballer' => 'Footballer',
                'cricketer'  => 'Cricketer',
                'bowler'     => 'Bowler',
            ]);

            if (!$form->isCreateForm()) {
                // If the entity is already exist the type of the entity cannot be␣
↪changed
                $typeField->readonly();
            }

            $form->section('Details', [
                $form->field(
                    Field::create('name', 'Name')->string()->required()
                )->bindToProperty(Player::NAME),
                //
                $form->field(
                    $typeField
                )->bindToCallbacks(function (Player $player) {
                    return [
                            Footballer::class => 'footballer',
                            Cricketer::class  => 'cricketer',
                            Bowler::class     => 'bowler',
                       ][get_class($player)];
                }, function () {
                    // Unused
                }),
            ]);

            $form->dependentOn(['type'], function (CrudFormDefinition $form, array
↪$input) {
                if ($input['type'] === 'footballer') {
                    $form->mapToSubClass(Footballer::class);

                    $form->continueSection([
                        $form->field(
                            Field::create('club', 'Club')->string()->required()
                        )->bindToProperty(Footballer::CLUB)
                    ]);
                }

                if ($input['type'] === 'cricketer' || $input['type'] === 'bowler') {
                    if ($input['type'] === 'cricketer') {
                        $form->mapToSubClass(Cricketer::class);
                    }

                    $form->continueSection([
                        $form->field(
                            Field::create('batting_average', 'Batting Average')->
↪int()->required()
                        )->bindToProperty(Cricketer::BATTING_AVERAGE)
                    ]);

                    if ($input['type'] === 'bowler') {
                        $form->mapToSubClass(Bowler::class);

                        $form->continueSection([
```

```
                            $form->field(
                                Field::create('bowling_average', 'Bowling Average')->
↪int()->required()
                            )->bindToProperty(Bowler::BOWLING_AVERAGE)
                        ]);
                    }
                }
            });
        });

        $module->removeAction()->deleteFromDataSource();

        $module->summaryTable(function (SummaryTableDefinition $table) {
            $table->mapProperty(Player::NAME)->to(Field::create('name', 'Name')->
↪string());

            $table->view('all', 'All')
                ->loadAll()
                ->asDefault();
        });
    }
}
```

Advanced CMS Topics

Custom Actions

It is often necessary to support add functionality to your CMS that is not part of the standard CRUD operations. You can define custom actions in your modules which can be executed in the backend.

General Actions

For actions that operate on multiple entities you can define a general action as follows

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

namespace App\Cms\Modules;

use Dms\Common\Structure\FileSystem\File;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IArticleRepository;
use App\Domain\Entities\Article;
use App\Domain\Services\ArticleExportService;
use Dms\Common\Structure\Field;
use Dms\Core\Form\Builder\Form;

/**
 * The article module.
 */
class ArticleModule extends CrudModule
```

```
{
    /**
     * @var ArticleExportService
     */
    public $articleExportService;

    public function __construct(IArticleRepository $dataSource, IAuthSystem
↪$authSystem, ArticleExportService $articleExportService)
    {
        $this->articleExportService = $articleExportService;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('article');

        $module->labelObjects()->fromProperty(Article::TITLE);

        $module->action('export')
            ->form(Form::create()->section('Settings', [
                Field::create('date_range', 'Date Range')->dateRange()->required()
            ]))
            ->returns(File::class)
            ->handler(function (array $input) {
                // Generate your export file...
                return $this->articleExportService->generateExportForDateRange($input[
↪'date_range']);
            });

        // Omitted the rest of the module configuration
    }
}
```

Object Actions

If the action operates on one entity, you can define an object action

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

namespace App\Cms\Modules;

use Dms\Common\Structure\FileSystem\File;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Services\Persistence\IJobRepository;
use App\Domain\Entities\Job;
use Dms\Common\Structure\Field;
```

```
use Dms\Core\Form\Builder\Form;

/**
 * The job module.
 */
class JobModule extends CrudModule
{
    /**
     * @var JobCompletionService
     */
    protected $jobCompletionService;

    public function __construct(IJobRepository $dataSource, IAuthSystem $authSystem,␣
↪JobCompletionService $jobCompletionService)
    {
        $this->jobCompletionService = $jobCompletionService;
        parent::__construct($dataSource, $authSystem);
    }

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('job');

        $module->labelObjects()->fromProperty(Job::NAME);

        $module->objectAction('mark-as-complete')
            ->form(Form::create()->section('Details', [
                Field::create('comments', 'Comments')->string()
            ]))
            ->handler(function (Job $job, array $input) {
                $this->jobCompletionService->completeJob($job, $input['comments']);
            });

        // Omitted module configuration
    }
}
```

Custom Fields

If you need a field in your backend that is not provided out of the box you can implement a custom field renderer in your app which will override how a field is displayed.

app/Cms/Modules/YourModule.php

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

namespace App\Cms\Modules;

use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
```

```
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;
use App\Domain\Entities\Entity;
use Dms\Common\Structure\Field;

/**
 * The photograph module.
 */
class YourModule extends CrudModule
{

    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        $module->name('your-module');

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('custom_field', 'Custom Field')->string()->
↪required()
                )->bindToProperty(Entity::PROPERTY),
                //
            ]);
        });

    }
}
```

app/Cms/Custom/CustomFieldRenderer.php

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

namespace App\Cms\Custom;

use Dms\Core\Form\Field\Type\ArrayOfType;
use Dms\Core\Form\Field\Type\StringType;
use Dms\Core\Form\IField;
use Dms\Core\Form\IFieldType;
use Dms\Web\Laravel\Renderer\Form\Field\BladeFieldRenderer;
use Dms\Web\Laravel\Renderer\Form\FormRenderingContext;


/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class CustomFieldRenderer extends BladeFieldRenderer
{

    /**
     * Gets the expected class of the field type for the field.
```

```
     *
     * @return array
     */
    public function getFieldTypeClasses(): array
    {
        return [StringType::class];
    }

    /**
     * @param FormRenderingContext $renderingContext
     * @param IField               $field
     * @param IFieldType           $fieldType
     *
     * @return bool
     */
    protected function canRender(FormRenderingContext $renderingContext, IField
↪$field, IFieldType $fieldType): bool
    {
        return $renderingContext->getAction()->getPackageName() === 'your-package'
            && $renderingContext->getAction()->getModuleName() === 'your-module'
            && $field->getName() === 'custom_field';
    }

    /**
     * @param FormRenderingContext $renderingContext
     * @param IField               $field
     * @param IFieldType           $fieldType
     *
     * @return string
     */
    protected function renderField(FormRenderingContext $renderingContext, IField
↪$field, IFieldType $fieldType): string
    {
        return view('dms.custom-fields.text', [
            'field' => $field,
        ])->render();
    }

    /**
     * @param FormRenderingContext $renderingContext
     * @param IField               $field
     * @param mixed                $value
     * @param IFieldType           $fieldType
     *
     * @return string
     */
    protected function renderFieldValue(FormRenderingContext $renderingContext,␣
↪IField $field, $value, IFieldType $fieldType): string
    {
        return view('dms.custom-fields.readonly-text', [
            'field' => $field,
            'value' => $value,
        ])->render();
    }
}
```

config/dms.php

Register your custom form field renderer in `config/dms.php`

```
<?php
[
    // ...
    'services' => [
        // ...
        'renderers' => [
            // ...
            'form-fields' => [
                // Add your class here
                App\Cms\Custom\CustomFieldRenderer::class,
                // ...
            ],
            // ...
        ],
    ],
]
```

resources/views/dms/custom-fields/text.blade.php

```
Your custom field:
<input type="text" name="{{ $field->getName() }}" value="{{ $field->getInitialValue()␣
↪}}" />
```

resources/views/dms/custom-fields/readonly-text.blade.php

```
Your custom value: {{ $value }}
```

Dependent Fields

Often static form builders are quite limiting and dont provide much of the functionality required for even a semi-complex CMS.

Dependent fields offer a powerful way to build forms that contain complex conditionals, multiple steps or can even be (ab)used to provide live calculations or previews.

Multiple steps example

Here is an example of a form which will show different options for the second field based on the value of the first field.

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

namespace App\Cms\Modules;

use Dms\Common\Structure\Field;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
```

```
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;

class YourModule extends CrudModule
{
    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        // ...

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('type', 'Type')->string()->required()->oneOf([
                        'fruit'     => 'Fruit',
                        'vegetable' => 'Vegetable',
                    ])
                )->withoutBinding(),
            ]);

            $form->dependentOn(['type'], function (CrudFormDefinition $form, array
↪$input) {

                if ($input['type'] === 'fruit') {
                    $form->continueSection([
                        $form->field(
                            Field::create('type_of_fruit', 'Type Of Fruit')->string()-
↪>required()->oneOf([
                                'apple'  => 'Apple',
                                'orange' => 'Orange',
                            ])
                        )->withoutBinding(),
                    ]);
                }

                if ($input['type'] === 'vegetable') {
                    $form->continueSection([
                        $form->field(
                            Field::create('type_of_vegetable', 'Type Of Vegetable')->
↪string()->required()->oneOf([
                                'carrot' => 'Carrot',
                                'potato' => 'Potato',
                            ])
                        )->withoutBinding(),
                    ]);
                }

            });
        });

        // ...
    }
}
```

Live calculation example

Here is an example of a form which will sum two integers and display the result.

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

namespace App\Cms\Modules;

use Dms\Common\Structure\Field;
use Dms\Core\Auth\IAuthSystem;
use Dms\Core\Common\Crud\CrudModule;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Core\Common\Crud\Definition\Table\SummaryTableDefinition;

class YourModule extends CrudModule
{
    /**
     * Defines the structure of this module.
     *
     * @param CrudModuleDefinition $module
     */
    protected function defineCrudModule(CrudModuleDefinition $module)
    {
        // ...

        $module->crudForm(function (CrudFormDefinition $form) {
            $form->section('Details', [
                $form->field(
                    Field::create('a', 'A')->int()->required()
                )->withoutBinding(),
                //
                $form->field(
                    Field::create('b', 'B')->int()->required()
                )->withoutBinding(),
            ]);

            $form->dependentOn(['a', 'b'], function (CrudFormDefinition $form, array
↪$input) {

                $form->continueSection([
                    $form->field(
                        Field::create('result', 'Result')->int()->value($input['a'] +
↪$input['b'])->readonly()
                    )->withoutBinding(),
                ]);

            });
        });

        // ...
    }
}
```

Extending Modules

It may become necessary to extend the functionality of a module which is installed as a standalone package. To achieve this, you can hook into to the events emitted and add the desired functionality.

You can add these event listeners directly in your `app/AppCms.php` or extract them into their own classes if applicable.

Adding form fields

Here is an example of adding an additional colour field to the products module from the shop package.

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

namespace App;

use Dms\Common\Structure\Field;
use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Core\Common\Crud\Definition\Form\CrudFormDefinition;
use Dms\Package\Analytics\AnalyticsPackage;
use Dms\Package\Shop\Cms\ShopPackage;
use Dms\Package\Shop\Domain\Entities\Product\Product;
use Dms\Web\Laravel\Auth\AdminPackage;
use Dms\Web\Laravel\Document\PublicFilePackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            'admin'      => AdminPackage::class,
            'documents'  => PublicFilePackage::class,
            'analytics'  => AnalyticsPackage::class,
            'shop'       => ShopPackage::class,
        ]);

        \Event::listen('dms::shop.products.defined-form', function
↪(CrudFormDefinition $form) {

            // Add your form fields here
            $form->continueSection([
                $form->field(
                    Field::create('colour', 'Colour')->string()->required()
                )->bindToCallbacks(
```

```
                    function (Product $product) {
                        return $product->metadata['colour'] ?? null;
                    },
                    function (Product $product, string $value) {
                        $product->metadata['colour'] = $value;
                    }
                ),
            ]);

        });

    }
}
```

Adding actions

Here is an example of adding a custom action to the product module.

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

namespace App;

use Dms\Common\Structure\Field;
use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Core\Common\Crud\Definition\CrudModuleDefinition;
use Dms\Package\Analytics\AnalyticsPackage;
use Dms\Package\Shop\Cms\ShopPackage;
use Dms\Web\Laravel\Auth\AdminPackage;
use Dms\Web\Laravel\Document\PublicFilePackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            'admin'      => AdminPackage::class,
            'documents'  => PublicFilePackage::class,
            'analytics'  => AnalyticsPackage::class,
            'shop'       => ShopPackage::class,
        ]);

        \Event::listen('dms::shop.products.define', function (CrudModuleDefinition
↪$module) {
```

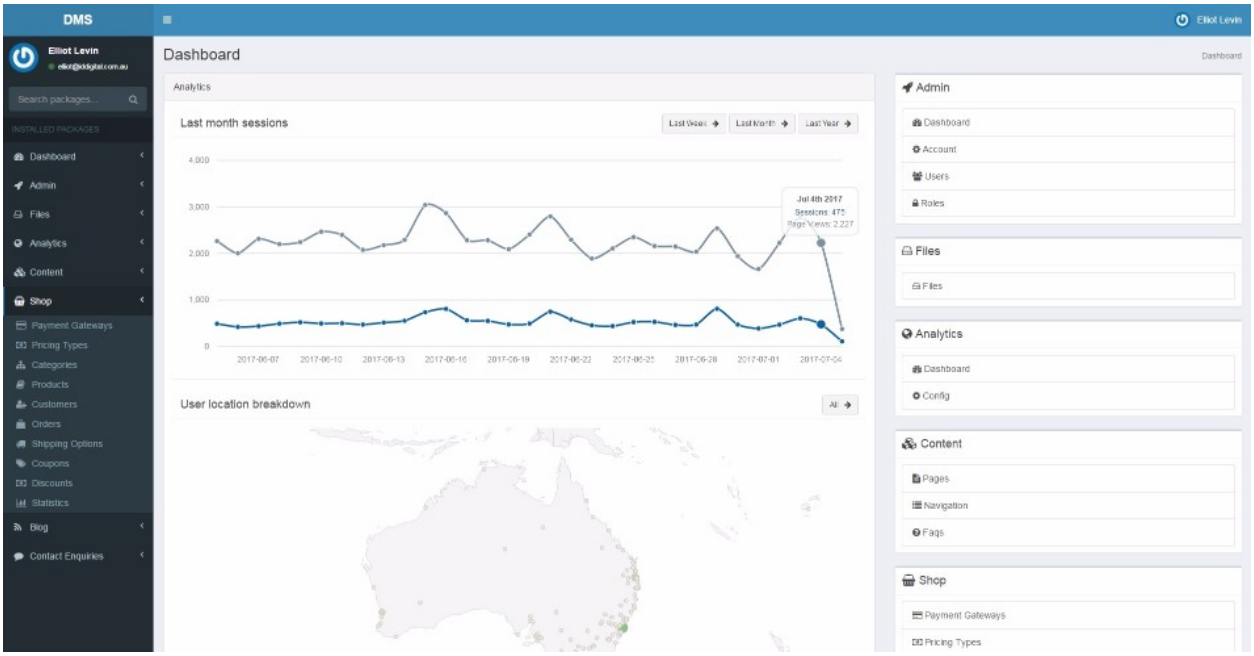
```
            // Add your custom action here
            $module->action('custom-action')
                ->handler(function (array $input) {
                    // ...
                });
        });

    }
}
```

Advanced ORM Topics

Packages

Analytics Package

The analytics package integrates with Google Analytics to provide basic analytics on the dashboard.



Installation

This package comes pre-installed with the DMS.

Usage

After configuring the Google Analytics settings in the backend you should ensure the analytics scripts are output to all your pages on the site.

Assuming you have a template blade view you can use this package to output the scripts:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        ...

        <!-- Output analytics scripts at the end of document head  -->
        {!! app(\Dms\Package\Analytics\AnalyticsEmbedCodeService::class)->
↪getEmbedCode() !!}
    </head>

    <body>
        ...
    </body>
</html>
```

Blog Package

The blog package provides the functionality for building a simple blogging platform.

Installation

Install the package via composer

```
composer require dms-org/package.blog
```

Register the package in `app/AppCms.php`

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

namespace App;

use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Package\Blog\Cms\BlogPackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            // Add this line to register the blog package...
```

```
            'blog' => BlogPackage::class,
        ]);
    }
}
```

Register the ORM in `app/AppOrm.php`

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

namespace App;

use Dms\Core\Persistence\Db\Mapping\Definition\Orm\OrmDefinition;
use Dms\Core\Persistence\Db\Mapping\Orm;
use Dms\Package\Blog\Infrastructure\Persistence\BlogOrm;

/**
 * The application's orm.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppOrm extends Orm
{
    /**
     * Defines the object mappers registered in the orm.
     *
     * @param OrmDefinition $orm
     *
     * @return void
     */
    protected function define(OrmDefinition $orm)
    {
        // Add this line to register the blog mappers
        $orm->encompass((new BlogOrm($this->iocContainer))->inNamespace('blog_'));
    }
}
```

Configure the blog package in `app/Providers/AppServiceProvider.php`

```
<?php

namespace App\Providers;

use App\AppCms;
use App\AppOrm;
use Dms\Core\ICms;
use Dms\Core\Persistence\Db\Mapping\IOrm;
use Dms\Package\Blog\Domain\Entities\BlogArticle;
use Dms\Package\Blog\Domain\Services\Config\BlogConfiguration;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
```

```
     *
     * @return void
     */
    public function register()
    {
        // ...

        // Register your blog configuration here...
        $this->app->bind(BlogConfiguration::class, function () {
            return BlogConfiguration::builder()
                ->setFeaturedImagePath(public_path('app/images/blog'))
                ->useDashedSlugGenerator()
                // Supply a preview callback to provide article previews
                // directly from the backend. This can be omitted to disable this␣
↪feature.
                ->setArticlePreviewCallback(function (BlogArticle $article) {
                    return view('blog.article', ['article' => $article])->render();
                })
                ->build();
        });
    }
}
```

Usage

Here is an example of how to utilise the default functionality of the blog package.

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

use Dms\Common\Structure\Web\EmailAddress;
use Dms\Package\Blog\Domain\Services\BlogKernel;

function demonstrateBlogFunctionality(BlogKernel $blog)
{
    $categories = $blog->categories()->getAll();

    $specificCategory = $blog->categories()->loadFromSlug('news');

    $articles = $blog->articles()->getAll();

    $articles = $blog->articles()->getPage(1, 15);

    $specificArticle = $blog->articles()->loadFromSlug('a-new-article');

    $authors = $blog->authors()->getAll();

    $specificAuthor = $blog->authors()->loadFromSlug('john-smith');

    $comment = $blog->comments()->postComment(
        $specificArticle->getId(),
        'John Smith',
        new EmailAddress('john-smith@gmail.com'),
        'This is a comment.'
    );
}
```

Content Package

Sometimes it is unnecessary to define fully-blown entities and mappers for structures in your app. The content package provides a generic structure that you can use to define the schema of the content of your app.

Installation

Install the package via composer

```
composer require dms-org/package.content
```

Register the ORM in `app/AppOrm.php`

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

namespace App;

use Dms\Core\Persistence\Db\Mapping\Definition\Orm\OrmDefinition;
use Dms\Core\Persistence\Db\Mapping\Orm;
use Dms\Package\Content\Persistence\ContentOrm;

/**
 * The application's orm.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppOrm extends Orm
{
    /**
     * Defines the object mappers registered in the orm.
     *
     * @param OrmDefinition $orm
     *
     * @return void
     */
    protected function define(OrmDefinition $orm)
    {
        // Add this line to register the content mappers
        $orm->encompass(new ContentOrm($this->iocContainer));
    }
}
```

Register the view composer

Register a view composer in the `boot` method of `app/Providers/AppServiceProvider.php` to resolve `Dms\Package\Content\Core\ContentLoaderService`.

```php
<?php

namespace App\Providers;

use Dms\Package\Content\Core\ContentLoaderService;
use Illuminate\Support\ServiceProvider;
```

```
class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        // Ensure the content can be loaded in all views
        // via $contentLoader
        \view()->composer('*', function ($view) {
            $view->with('contentLoader', app(ContentLoaderService::class));
        });
    }

    // Omitted...
}
```

Usage

Defining your content schema

Create a custom content package in `app/Cms/MyContentPackage.php`

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

namespace App\Cms;

use App\Http\Controllers\PageController;
use Dms\Core\ICms;
use Dms\Package\Content\Cms\ContentPackage;
use Dms\Package\Content\Cms\Definition\ContentConfigDefinition;
use Dms\Package\Content\Cms\Definition\ContentGroupDefiner;
use Dms\Package\Content\Cms\Definition\ContentModuleDefinition;
use Dms\Package\Content\Cms\Definition\ContentPackageDefinition;

/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class MyContentPackage extends ContentPackage
{
    protected static function defineConfig(ContentConfigDefinition $config)
    {
        $config
            ->storeImagesUnder(public_path('app/content/images'))
            ->mappedToUrl(url('app/content/images'))
            ->storeFilesUnder(public_path('app/content/files'));
    }

    /**
     * Defines the structure of the content.
     *
     * @param ContentPackageDefinition $content
     *
```

```
     * @return void
     */
    protected function defineContent(ContentPackageDefinition $content)
    {
        $content->module('pages', 'file-text', function (ContentModuleDefinition
↪$module) {

            $module->group('home', 'Home')
                ->withText('title', 'Title')
                ->withHtml('content', 'Content')
                ->withArrayOf('carousel-item', 'Hero Carousel', function␣
↪(ContentGroupDefiner $item) {
                    $item
                        ->withText('caption', 'Caption')
                        ->withImage('image', 'Image');
                })
                ->withMetadata('title', 'Meta - Title')
                ->withMetadata('description', 'Meta - Description')
                // Optionally define a preview callback to enable
                // live content previews in the CMS
                ->setPreviewCallback(function () {
                    return \app()->call(PageController::class . '@showHomePage')->
↪render();
                });

            // Add more pages here ...
        });

        // Define additional modules here...

        // Optionally, register your own custom modules if you want to include
        // them in your content package...
        $content->customModules([
            'custom' => CustomModule::class
        ]);
    }
}
```

Register the package in `app/AppCms.php`

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

namespace App;

use App\Cms\MyContentPackage;
use Dms\Core\Cms;
use Dms\Core\CmsDefinition;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
```

```
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            // Add this line to register your content package...
            'content' => MyContentPackage::class,
        ]);
    }
}
```

Load the content in your views

You can then use the `$contentLoader` in your blade files to retrieve the content from the database.

```
<?php $content = $contentLoader->load('pages.home') ?>
<html>
<head>
    <title>{{ $content->getMetadata('title') }}</title>
    <meta name="description" content="{{ $content->getMetadata('title') }}">
</head>
<body>
<header>
    <h1>{{ $content->getText('title') }}</h1>
</header>
<article>
    {!! $content->getHtml('content') !!}
</article>
<ul class="carousel">
    @foreach($content->getArrayOf('carousel-item') as $item)
        <li>
            <img src="{{ $item->hasImage('image') ? asset_file_url($item->getImage(
↪'image')) : 'http://placehold.it/500x500' }}" />
            <span>{{ $item->getText('caption') }}</span>
        </li>
    @endforeach
</ul>
</body>
</html>
```

FAQ's Package

It is often necessary to include a question and answer section in a website. The FAQ package provides the functionality for this.

Installation

Install the package via composer

```
composer require dms-org/package.faqs
```

Register the package in `app/AppCms.php`

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

namespace App;

use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Package\Faq\Cms\FaqPackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            // Add this line to register the package...
            'faqs' => FaqPackage::class,
        ]);
    }
}
```

Register the ORM in `app/AppOrm.php`

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

namespace App;

use Dms\Core\Persistence\Db\Mapping\Definition\Orm\OrmDefinition;
use Dms\Core\Persistence\Db\Mapping\Orm;
use Dms\Package\Blog\Infrastructure\Persistence\BlogOrm;

/**
 * The application's orm.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppOrm extends Orm
{
```

```
    /**
     * Defines the object mappers registered in the orm.
     *
     * @param OrmDefinition $orm
     *
     * @return void
     */
    protected function define(OrmDefinition $orm)
    {
        // Add this line to register the mappers
        $orm->encompass(new FaqOrm());
    }
}
```

Usage

Load the FAQ's in your controller

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

namespace App\Http\Controllers;

use Dms\Package\Faq\Core\FaqLoaderService;

/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class FaqsController extends Controller
{
    public function showFaqsPage(FaqLoaderService $faqLoaderService)
    {
        return view('faqs', [
            'faqs' => $faqLoaderService->loadFaqs(),
        ]);
    }
}
```

Display the FAQ's in your view

```
<article>
    <header>
        <h1>FAQ's</h1>
    </header>
    <section class="content">
        @foreach($faqs as $faq)
            <div>
                <strong>{{ $faq->question }}</strong>
                <div>{!! $faq->answer->asString() !!}</div>
            </div>
        @endforeach
    </section>
</article>
```

Contact Us Package

Most website's include a contact form. The contact us packages allows these enquires to be stored in the backend.

Installation

Install the package via composer

```
composer require dms-org/package.contact-us
```

Register the package in `app/AppCms.php`

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

namespace App;

use Dms\Core\Cms;
use Dms\Core\CmsDefinition;
use Dms\Package\ContactUs\Cms\ContactUsPackage;

/**
 * The application's cms.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppCms extends Cms
{
    /**
     * Defines the structure and installed packages of the cms.
     *
     * @param CmsDefinition $cms
     *
     * @return void
     */
    protected function define(CmsDefinition $cms)
    {
        $cms->packages([
            // Add this line to register the package...
            'contact-us' => ContactUsPackage::class,
        ]);
    }
}
```

Register the ORM in `app/AppOrm.php`

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

namespace App;

use Dms\Core\Persistence\Db\Mapping\Definition\Orm\OrmDefinition;
use Dms\Core\Persistence\Db\Mapping\Orm;
use Dms\Package\ContactUs\Persistence\ContactUsOrm;
```

```
/**
 * The application's orm.
 *
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class AppOrm extends Orm
{
    /**
     * Defines the object mappers registered in the orm.
     *
     * @param OrmDefinition $orm
     *
     * @return void
     */
    protected function define(OrmDefinition $orm)
    {
        // Add this line to register the mappers
        $orm->encompass(new ContactUsOrm());
    }
}
```

Usage

Sample contact form

```
<form action="..." method="post">
   {!! csrf_field() !!}

   <div class="form-group">
       <label>Your Name</label>
       <input type="text" class="form-control" required name="name">
   </div>
   <div class="form-group">
       <label>Email</label>
       <input type="email" class="form-control" required name="email">
   </div>
   <div class="form-group">
       <label>Subject</label>
       <input type="text" class="form-control" required name="subject">
   </div>
   <div class="form-group">
       <label>Your Message</label>
       <textarea name="message" class="form-control" required></textarea>
   </div>

   <button type="submit" class="btn btn-primary">Submit Enquiry</button>
</form>
```

Record contact enquiry and send notification

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

namespace App\Http\Controllers;
```

```
use Dms\Package\ContactUs\Core\ContactEnquiry;
use Dms\Package\ContactUs\Core\ContactEnquiryService;
use Illuminate\Http\Request;
use Illuminate\Mail\Message;

/**
 * @author Elliot Levin <elliotlevin@hotmail.com>
 */
class ContactController extends Controller
{
    public function submitEnquiry(Request $request, ContactEnquiryService
↪$contactEnquiryService)
    {
        $this->validate($request, [
            'name'    => 'required',
            'email'   => 'required|email',
            'subject' => 'required',
            'message' => 'required',
        ]);

        $contactEnquiryService->recordEnquiry(
            $request->input('email'),
            $request->input('name'),
            $request->input('subject'),
            $request->input('message'),
            function (ContactEnquiry $enquiry) {
                // Send the notification email...
            }
        );

        return redirect('contact/success');
    }
}
```

Building Your Own Package

Building your own DMS package is similar to building a normal application. You must define your entities, mappers and modules.

Then you must create a composer package so it can be installed in other projects.

For a complete example of a working package, including tests, see the blog package.