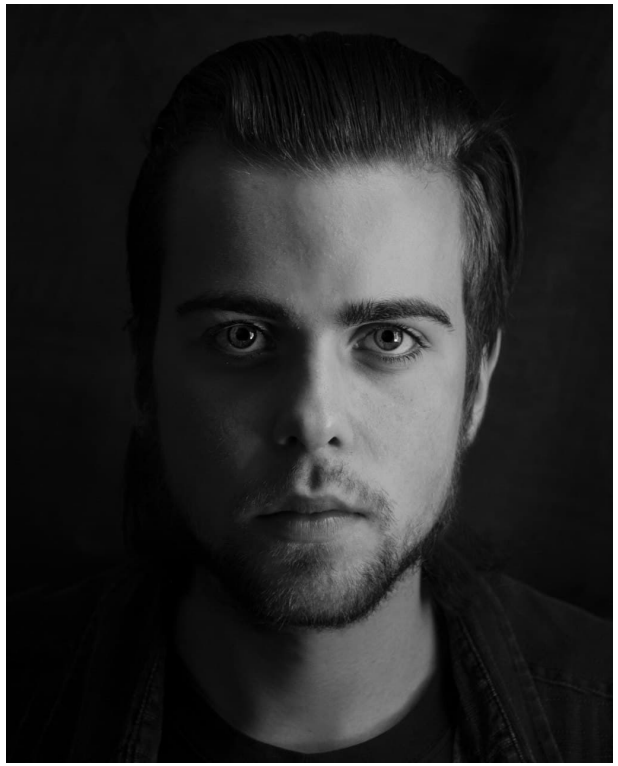
Веб разработка на PHP Symfony

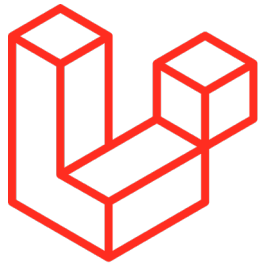
Doctrine (lifecycle callbacks, query builder, relations)

Introduction



Демьян Костельный
Middle PHP Developer

demian-kostelny-613b90151



Тема урока

Doctrine
(lifecycle callbacks, query builder, relations)

План урока

1. Doctrine Lifecycle Callbacks
2. Repository query builder
3. One-to-one, one-to-many, many-to-many doctrine relations

Doctrine Lifecycle Callbacks

Doctrine Lifecycle Callbacks

В случае, если нужно выполнить какое-то действие во время обновления/изменения либо удаления чего-то в записи Entity - нужно воспользоваться Doctrine Lifecycle Callbacks (обратным вызовом жизненного цикла).

Lifecycle callbacks представляют из себя публичные функции в классе Entity, которые позволяют обращаться к полям Entity в самые разные моменты выполнения какого-то действия (добавления/обновления/удаления).

Пример простого Lifecycle Callback - это когда вызывается публичный метод setCreatedAtValue() только тогда, когда новая запись в БД сохранится в таблице.

Doctrine Lifecycle Callbacks

Пример использования Lifecycle callback:

```php
/**
 * @ORM\PrePersist
 */
public function setNameValue(): void
{
    // Данная функция будет вызвана только тогда - перед тем как новая
    // запись появится в БД. Это делается благодаря обозначению @ORM\PrePersist
    // Как можно заметить - при сохранении новой записи в конце поля
    // name всегда будет добавляться текст "testing value"
    $this->name = $this->name . 'testing value';
}
```

Doctrine Lifecycle Callbacks

Некоторые Lifecycle Events которые можно использовать:

Название события	**Описание вызова события**
prePersist	Вызывает функцию перед тем, как сохранить запись в таблицу Entity.
postPersist	Вызывает функцию после того, как запись была сохранена в таблицу Entity.
preRemove	Вызывает функцию перед тем, как удалить запись в таблице Entity.
postRemove	Вызывает функцию после того, как запись была удалена из таблицы Entity.
preUpdate	Вызывает функцию перед тем, как обновить запись в таблице Entity.
postUpdate	Вызывает функцию после того, как запись была обновлена в таблице Entity.

Doctrine Lifecycle Callbacks

Полный список lifecycle callback событий можно посмотреть по ссылке:

https://www.doctrine-project.org/projects/doctrine-orm/en/latest/reference/events.html#lifecycle-events

Repository query builder

Repository query builder

В предыдущем уроке уже говорилось о репозиториях в Doctrine, сейчас мы поговорим о компоненте в Doctrine, который называется Query Builder. Суть его заключается в том, что можно строить SQL-запросы через функции данного класса, при этом не прописывая какого-то большого количества SQL кода запроса. Пример использования Query Builder в репозитории `src/Repository/ProductRepository`:

```php
public function findWhereName(string $name): array
{

    $queryBuilder = $this->createQueryBuilder('q') // Создаём новый queryBuilder чтобы создать запрос
            ->where('q.name = ":name"') // создаём условие что поле name должно быть равно переменной $name
            ->setParameter('name', $name) // выше мы создали ссылку на :name - теперь пора присвоить значение от $name
            ->orderBy('q.name', 'ASC'); // Делаем сортировку по полю name - по типу ASC

    $query = $queryBuilder->getQuery(); // Достём наш запрос из $queryBuilder

    return $query->execute(); // Запускаём наш запрос

}
```

Repository query builder

То есть, в общем получается, что Query Builder - это полноценный класс, который позволяет строить SQL запросы прямо через PHP код, и при этом делать это эффективно и легко.

One-to-one, one-to-many, many-to-many doctrine relations

One-to-one, one-to-many, many-to-many doctrine relations

Как известно - таблицы в MySQL таблицах можно связывать между собой. К примеру можно сделать так, чтобы одна таблица была привязана к полю ID второй таблицы и использовала это в качестве некого ключа.

Вообще - это называется реляционными таблицами. Чтобы создать поле, которые будет привязано к какой-то другой таблице, при создании Entity в типе поля нужно указать - relation. После этого нужно будет указать по отношению к какому классу новый Entity должен быть реляционным. После этого поле будет привязано к указанному Entity по принципу Many-to-one.

One-to-one, one-to-many, many-to-many doctrine relations

Это были не все типы реляционных отношений, полный список можно посмотреть в официальной документации Doctrine ORM:

https://www.doctrine-project.org/projects/doctrine-orm/en/2.8/reference/association-mapping.html

Проверка знаний

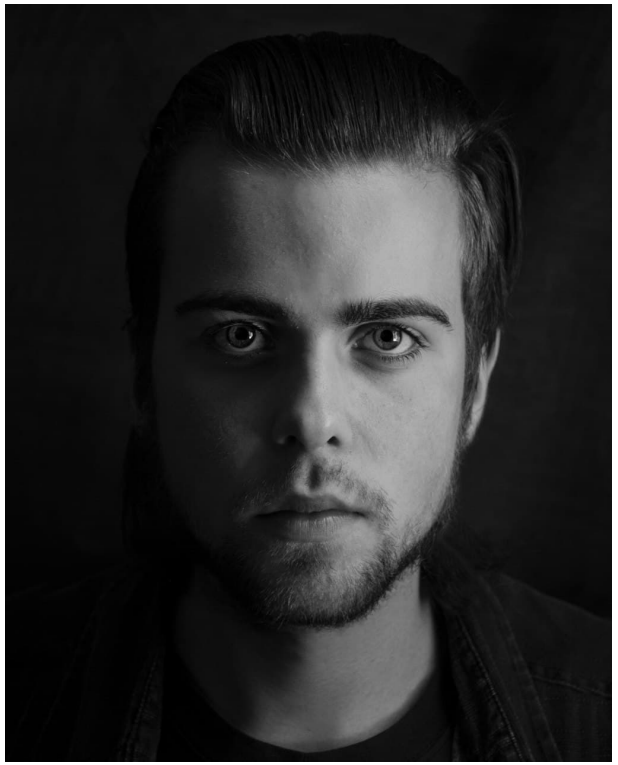
TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

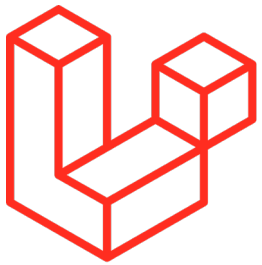
Проверьте как Вы усвоили данный материал на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!



Демьян Костельный
Middle PHP Developer



Информационный видеосервис для разработчиков программного обеспечения

TestProvider Pearson CyberBionic systematics Microsoft Partner Windows Azure EBA