

Работа с базой данных

Взаимодействие с базой данных происходит на базе [ORM](#).

Далее будет сильное упрощение, но оно отображает основной смысл работы

В данном случае требуется объяснение трех разных объектов в проекте

EntityManager - Менеджер сущностей

Это механизм для записи данных в базу. Он автоматически приводит все нужные форматы и даже устанавливает внешние ключи.

Самое простое что о нем можно сказать - он делает работу по записи в БД за вас.

Repository - Репозиторий

Это механизм для работы с базой данных через Entity. Механизм позволяет выбирать необходимые сущности из базы данных, сразу конвертируя все необходимые данные

Любой репозиторий предоставляет набор стандартных функций для работы с базой данных. Основные, которые требуются перечислены ниже

- `find(int $id)` - Получение сущности по PRIMARY KEY. Результат - объект сущности
- `findBy(array $criteria, ?array $orderBy=null, ?int $limit=null, ?int $offset=null)` - Поиск сущностей по запросу WHERE описанному как массив значений и в сортировке по массиву значений. Результат - Array of Entity
 - `criteria` - Массив параметров для WHERE например `['entryID'⇒'foo bar']`. Ключ - имя поля сущности, значение - необходимый фильтр
 - `orderBy` - Массив параметров для ORDER BY, например `['id'⇒'DESC','time'⇒'ASC']`. Ключ - имя поля сущности, значение : ASC или DESC
 - `limit` - Ограничение по количеству строк в ответе
 - `offset` - стартовый отступ в запросе Limit
- `findAll()` - Вывод всех записей в таблице. Результат - Array of Entity
- `findOneBy(array $criteria, ?array $orderBy=null)` - аналогичен FindBy но выводит только одну сущность. Внутри репозитория для конкретной сущности можно писать свои запросы к базе данных используя [queryBuilder](#) или [прямой SQL](#)

QueryBuilder предпочтительнее, так как он позволяет делать запросы к базе данных независимо от ее движка (Mysql, postgres и т.д.)

Entity - Сущность

Это класс, который является одновременно строкой результата запроса из БД и так же описанием таблицы

Пример

```
<?php

namespace App\Entity;

use App\Repository\FooRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: FooRepository::class)]
class Foo{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255, nullable: true)]
    private ?string $entryID = null;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getEntryID(): ?string
    {
        return $this->entryID;
    }

    public function setEntryID(?string $entryID): static
    {
        $this->entryID = $entryID;

        return $this;
    }
}
```

Конструктор данных механизмов есть в [консоли](#). Потому symfony напишет весь класс сама за Вас. Запросы о состоянии идут в интерактивном режиме.

Получаемая таблица

```
CREATE TABLE `foo` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `entry_id` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=63 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_unicode_ci;
```

Как видно из примера, таблица полностью повторяет все поля имеющие признак ORM\Column.

Можно дописать свои функции в класс если это требуется.

Строка результата

Когда мы совершили выборку из базы данных через репозиторий, мы получаем результат как объекты типа Entity. В данном примере - Foo

```
//some code  
$foos = $fooRepository->findAll();  
foreach($foos as $foo){  
    echo $foo->getEntryID();  
}  
//some code
```

В данном случае результат запроса к базе данных превращается в Array<Foo> и к каждому элементу можно обращаться как к классу

Запись данных в базу

Для записи данных используется EntityManager. Он позволяет интерпретировать объект Entity в строку запроса на вставку (при конвертации идет проверка по всем возможным параметрам MySQL - *realescapestring*, *preparedstatement*, *sqlinjection* и т.д.)

Для записи данных в БД требуется заполнить класс Entity и отправить его в базу (Все преобразования происходят автоматически)

```
//some code  
$foo = new Foo();  
$foo->setEntryID('foo bar');  
$entityManager->persist($foo);  
$entityManager->flush();  
//some code
```

В результате этого кода в базе данных в таблице Foo появится новая строка с `entryid = 'foo bar'`. Установка ID не нужна, так как это Autoincrement что указывается параметром `#[ORM\Id]` `#[ORM\GeneratedValue]`

Так же можно делать массовое добавление данных в базу путем множественного Persist и одного Flush

```
//some code
$array = [1,2,3,4,5,6,7,8,9,10];
foreach($array as $a){
    $foo = new Foo();
    $foo->setEntryID($a);
    $entityManager->persist($foo);
}
$entityManager->flush();
//some code
```

В данном случае произойдет одно обращение к базе данных для записи всех 10 строк.

From:

<http://mgdemo.ru:5555/> - МИС Mgerm

Permanent link:

http://mgdemo.ru:5555/doku.php?id=mgerm_external:symfony:database

Last update: **14-11-2024 12:31**

