Developer & Code

12. Адаптер для подключения к базе данных. Блог на PHP

Липень 6, 2021

В этом уроке, мы создадим адаптер подключения к базе данных. Также, перенесем все настройки в файл окружения `.env`. Для этого, мы воспользуемся библиотекой `devcoder-xyz/php-dotenv`, которую можно установить с помощью `composer`.

Какие темы мы разберем

В 12. PDO Провайдер. Блог на PHP уроке мы рассмотрим следующие темы:

- Добавим библиотеку для загрузки настроек с файла `.env`
- Создадим адаптер подключения к базе данных
- Усовершенствуем логику подключения к базе данных
- Используем Контейнер Зависимостей для загрузки адаптера подключения

Приятного просмотра:

В рамках урока, мы напишем следующий код.

Давайте добавим библиотеку для работы с .env файлами.

```
composer require devcoder-xyz/php-dotenv
```

Создаем файл `.env`, в котором будет находиться настройки подключения к базе данных. Мы не будем использовать файл `config/database.php`, так как настройки, которые могут меняться в зависимости от окружения, принято держать в файлах типа `.env`.

```
DATABASE_DSN=mysql:host=127.0.0.1;dbname=blog_php
DATABASE_USERNAME=root
DATABASE_PASSWORD=
```

Не забываем перенести настройки с файла `database.php` в новый файл `.env`.

После этого, спокойно удаляем файл `config/database.php`.

В файле `index.php` нужно указать путь к файлу `.env` классу `DevCoder\DotEnv`.

```
use DevCoder\DotEnv;

(new DotEnv(__DIR__ . '/.env'))->load();


/// Удаляем ненужное подключение к базе данных
```

Далее, создаем класс `Blog\Database` в директории `src/`. Этот класс будет служить адаптером для предоставления подключения к базе данных с использованием PDO класса.

```
<?php

declare(strict_types=1);

namespace Blog;

use http\Exception\InvalidArgumentException;
use PDO;
use PDOException;

class Database
{
    /**
     * @var PDO
     */
    private PDO $connection;

    /**
     * Database constructor.
     * @param string $dsn
     * @param string|null $username
     * @param string|null $password
     */
    public function __construct(string $dsn, string $username = null,
    {
        try {
            $this->connection = new PDO($dsn, $username, $password);
            $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ER
```

```
                $this->connection->setAttribute(PDO::ATTR_DEFAULT_FETCH_MO
            } catch (PDOException $exception) {
                throw new InvalidArgumentException($exception->getMessage(
            }
        }

        /**
         * @return PDO
         */
        public function getConnection(): PDO
        {
            return $this->connection;
        }
    }
```

В файле `config/di.php`, с помощью функции `getenv()`, загрузим настройки и передадим в класс `Database`.

```
use Blog\Database;

return [
    Database::class => autowire()
        ->constructorParameter('dsn', getenv('DATABASE_DSN'))
        ->constructorParameter('username', getenv('DATABASE_USERNAME')
        ->constructorParameter('password', getenv('DATABASE_PASSWORD')
];
```

С помощью контейнера, создаем и возвращаем объект типа `Database`, который использует настройки из `.env` файла.

```
$connection = $container->get(Database::class)->getConnection();
```

Скачать код данного урока можно по ссылке.

Підписуйтеся на канал “Спільнота програмістів – Developer & Code” в телеграмі

Макс Пронько - Програміст, CEO компанії Pronko Consulting, розбираємо Веб технології. Автор YouTube каналу Макс Пронько. Телеграм група сайту.

← 11. Контейнер зависимостей. Блог на PHP

Як додати кнопку до форми в Adobe Commerce →