در ابتدا فایل های function.php,routes.php را داخل پوشه core قرار داده و به فایل index.php داخل پوشه public رفته و مسیر این دو فایل را بروزرسانی میکنیم.

به فایل Database.php رفته و کد زیر را برای تعریف namespace در ابتدای فایل قبل تعریف کلاس مینویسیم:

```
namespace core;
```

حالا اگر وارد صفحه یادداشت ها شویم خطا میدهد که کلاس Database پیدا نمیشه.

به کلاس Database رفته و در ابتدا کد استفاده از namespace را مینویسیم بعد در تعریف یک شی از کلاس Database تغییراتی را اعمال میکنیم:

```
use core\Database;
```

و در هنگام استفاده :

```
$notes = new Database($config['database'], "root", "");
```

فقط نام پوشه ای که فایل های کلاس داخلش هست با نام فضای نام باید یکی باشد مثلا اگر نام فضای نام cores باشد خطا میدهد. دلیلش اینه که از نظر phpفرقی نمیکنه که نام پوشه و فضای نام متفاوت یا یکی باشد ولی اگه از سیستم autoloud استفاده میشه نام ها باید یکی باشد ما هم درپروژه laracast از سیستم autoload استفاده کردیم پس باید یکی باشد.

در آدرس دهی فضای نام ها از \ به جای / استفاده میشه که ما به مشکل میخوریم اگر در ابتدای spl_autoload_register در فایل indexکد ;var_dump($class) رو بزنیم خروجی cores\Database هست که باید cores/Database باشه پس به فایل index.php داخل پوشه public رفته و در بخش spl_autoload_register کد زیر را مینویسیم:

```
spl_autoload_register(function ($class) {
    $class = str_replace("\\", "/", $class);
    // بساز کلاس نام اساس بر را کلاس فایل مسیر
    $path = base_path("{$class}.php");

    if (file_exists($path)) {
        require_once $path;
    }
});
```

نکته مهم

اگر با خطای زیر مواجه شدید:

Fatal error: Uncaught Error: Class "core\PDO" not found in C:\xampp\htdocs\laracast-php\core\Database.php:12 Stack trace: #0 C:\xampp\htdocs\laracast-php\controller\note\index.php(5): core\Database->__construct(Array, 'root', '') #1 C:\xampp\htdocs\laracast-php\public\index.php(20): require('C:\\xampp\\htdocs...') #2 {main} thrown in **C:\xampp\htdocs\laracast-php\core\Database.php** on line **12**

چون داخل فضای نام core هستیم و میخواهیم از کلاس داخلی PDO استفاده کنیم php خطا میده چون دنبال کلاس PDO در فضای نام core میگرده پس

◆راه‌حل

برای استفاده از کلاس‌های **داخلی PHP** مثل PDO، باید از **global namespace**استفاده کنی و قبل از اسمش یک \ بذاری:

```
$this->connection = new \PDO($dsn, $username, $password);
```

یا کامل با:use

```
use PDO;

$this->connection = new PDO($dsn, $username, $password);
```

دلیل مشکل

وقتی توی یک namespace هستی، هر اسم کلاسی که بدون \ یا بدون useبنویسی، PHP اول در همون namespace دنبالش می‌گرده.
پس اگر coreهستی و نوشتی PDO، میره دنبال .core\PDO

با گذاشتن \ قبلش (\PDO) به PHP میگی:

برو توی global namespace و کلاس PDO پیش‌فرض PHP رو پیدا کن.

به کلاسهای validatoreوresponse رفته و ابتدای کدش ;namespace core نوشته و در هر فایلی که از این کلاس ها استفاده شده از use core\validatore استفاده میکنی. یا در php storm بر روی کلاس رفته alt+shift+enter یا import classes رو میزنی که خودش دستور use را بالا اضافه میکنه.

1. What — چیست؟

Namespaceدر PHP مثل یک پوشه برای کدها عمل می‌کند.
مثلاً اگر دو کتابخانه مختلف هر دو یک کلاس به اسم Userداشته باشند، بدون namespace با خطای "**Cannot redeclare class**"مواجه می‌شویم. ولی وقتی هرکدام در namespace خودش باشد، PHPمی‌تواند آن‌ها را از هم جدا کند.

```
<?php
namespace LibraryA;
class User {
    public function getName() {
        return "User from LibraryA";
    }
}

namespace LibraryB;
class User {
    public function getName() {
        return "User from LibraryB";
    }
}
```

---

2. Why — چرا؟

دلایل اصلی استفاده از namespace در:PHP

۱. **جلوگیری از تداخل نام‌ها** بین کلاس‌ها و توابع مختلف.
۲. **سازمان‌دهی کدها** به شکلی منطقی (مثل پوشه‌بندی فایل‌ها).
۳. **کار با کتابخانه‌ها و فریم‌ورک‌ها** بدون نگرانی از هم‌نام بودن کلاس‌ها.
۴. **افزایش خوانایی کد**، مخصوصاً در پروژه‌های بزرگ.

---

3. How — چطور استفاده می‌کنیم؟

الف) تعریف namespace

با دستور `namespace` در بالای فایل (قبل از هر کد دیگر) تعریف می‌شود:

```
<?php
namespace App\Controllers;

class HomeController {
    public function index() {
        echo "This is HomeController in
App\\Controllers";
    }
}
```

ب) استفاده از namespace

برای دسترسی به کلاس‌ها یا توابعی که در namespace دیگر هستند، دو راه داریم:

1. استفاده با مسیر کامل (Fully Qualified Name)

```
<?php
require 'HomeController.php';

$controller = new \App\Controllers\HomeController();
$controller->index();
```

توجه: در ابتدای مسیر `\` می‌گذاریم تا از global namespace شروع شود.

---

2. استفاده با `use`

```
<?php
require 'HomeController.php';

use App\Controllers\HomeController;

$controller = new HomeController();
$controller->index();
```

ج) alias (نام مستعار)

یا کلاس یک اسم کوتاه‌تر تعریف کنیم namespace می‌توانیم برای:

```
use App\Controllers\HomeController as HC;

$controller = new HC();
$controller->index();
```

✅خلاصه:

- **Namespace = یک پوشه مجازی برای کلاس‌ها/توابع/ثابت‌ها.**
- **دلیل استفاده: جلوگیری از تداخل اسم‌ها و سازماندهی کد.**
- **روش استفاده: تعریف با `namespace` و استفاده با `use` یا مسیر کامل.**

جلسه ۳۱ Namespaces — و Autoload در پروژه تو

با استناد به توضیحاتی که دادی، این جلسه عملاً سه کار می‌کنه:
۱ (منظم‌سازی ساختار پوشه‌ها، ۲) اضافه‌کردن namespace به کلاس‌ها، ۳) اصلاح autoload تا با namespace کار کنه.

(1ساختار پوشه‌ها بعد از جابه‌جایی

```
laracast-php/
├─ core/
│   ├─ Database.php
│   ├─ Validator.php
│   ├─ Response.php
│   ├─ functions.php
│   └─ routes.php
├─ controller/
│   └─ note/
│       ├─ index.php
│       ├─ show.php
```

```
│      └─ create.php
├─ controller/views/
│   └─ …
└─ public/
    └─ index.php
```

نکته **autoload**: **فقط کلاس‌ها** رو لود می‌کنه؛ فایل‌های فانکشن (مثل functions.php) رو هنوز باید require کنی (مگر با Composer و بخش files).

---

namespace (2زدن به کلاس‌ها

core/Database.php

```
<?php
namespace core;

class Database {
    public $connection;
    public $statement;

    public function __construct($config, $username =
"root", $password = "") {
        // از گلوبال‌نیم‌اسپیس \PDO حواست باشه:
        $dsn = "mysql:" . http_build_query($config, '',
';');
        $this->connection = new \PDO($dsn, $username,
$password, [
            \PDO::ATTR_DEFAULT_FETCH_MODE =>
\PDO::FETCH_ASSOC
        ]);
    }

    public function query($query, $params = []) {
        $this->statement = $this->connection-
>prepare($query);
        $this->statement->execute($params);
        return $this; // fluent API
    }
```

```
    public function fetch() {
        return $this->statement->fetch();
    }

    public function get() {
        return $this->statement->fetchAll();
    }

    public function findOrFail() {
        $result = $this->statement->fetch();
        if (!$result) {
            http_response_code(404);
            require
base_path('controller/views/404.view.php');
            exit;
        }
        return $result;
    }
}
```

core/Validator.php و core/Response.php

ابتدای هر کد:

```
<?php
namespace core;
// ... مابقی کلاس
```

چرا `\PDO`**؟ چون وقتی داخل** `namespace core;` **هستی، اگر بنویسی** `PDO`**، PHP دنبال** `core\PDO` **می‌گرده. با** `\PDO` **می‌گی برو از فضای نام گلوبال.**

---

3) استفاده از `use` **در کنترلرها**

مثلاً در `controller/note/index.php`**:**

```
<?php
```

```
use core\Database;

$config = require base_path('core/routes.php'); // یا
config
$db = new Database($config['database'], 'root', '');

$notes = $db->query(
    "SELECT * FROM notes WHERE user_id = :id",
    ['id' => 3]
)->get();

view('note/index.view.php', ['notes' => $notes,
'heading' => 'Notes']);
```

- خط use core\Database; یعنی از کلاس Database داخل فضای نام core استفاده می‌کنی.
- بدون use باید بنویسی: new \core\Database(...)

---

4) Autoload هماهنگ با namespace

در public/index.php:

```
<?php
const BASE_PATH = __DIR__ . '/../';
require_once BASE_PATH . 'core/functions.php';

// اتولودر: \ را به / تبدیل کن و از ریشه پروژه فایل را بیار
spl_autoload_register(function ($class) {
    // PSR-4 ساده‌شده:
    $class = str_replace('\\', '/', $class);
    $path  = base_path("{$class}.php"); // مثلاً
core/Database.php

    if (file_exists($path)) {
        require_once $path;
    }
```

```
});

// routes:
$routes = require base_path('core/routes.php');

// Resolve URL:
$url = parse_url($_SERVER['REQUEST_URI'],
PHP_URL_PATH);

// ساده‌ترین Router:
if (array_key_exists($url, $routes)) {
    require base_path($routes[$url]);
} else {
    http_response_code(404);
    echo "404 - Page not found";
    exit;
}
```

چرا str_replace('\\','/',...)؟

چون کلاس‌ها به‌صورت core\Database **به اتولودر می‌رسن؛ برای رسیدن به مسیر فایل، باید به** core/Database.php **تبدیل بشن.**

اگر خواستی سیستم‌عامل بی‌تفاوت باشه، می‌تونی به‌جای / **از** DIRECTORY_SEPARATOR **استفاده کنی. (هرچند** / **در ویندوز هم کار می‌کنه(.**

(5انتقال فایل‌های غیرکلاسی به core/

- routes.php **و** functions.php **رو بردی داخل** core/.
- functions.php **را هنوز باید** **require** **کنی چون اتولودر فقط کلاس‌ها را لود می‌کند.**

core/functions.php

```
<?php
function base_path($path) {
    return BASE_PATH . ltrim($path, '/');
```

```
}

function view($path, $attributes = []) {
    extract($attributes);
    require base_path("controller/views/{$path}");
}
```

extract($attributes) کلیدهای آرایه را به متغیر تبدیل می‌کند (مثل $heading).

---

6) نکته‌های رایج و خطاهای احتمالی

- **Class "core\PDO" not found**:
  به‌جای PDO از \PDO استفاده کن یا use PDO; بالای فایل.
- **Mismatch بین نام پوشه و namespace:**
  چون اتولودر تو این پروژه خیلی ساده است و مستقیماً \ را / می‌کند، باید core\Database دقیقاً به core/Database.php بخورد. پس ریشه‌ی namespace و نام پوشه باید هم‌خوان باشند.
- **اتولودر فقط کلاس‌ها را لود می‌کند:**
  فایل‌های فانکشن را با require بیار (مثل functions.php).
- **اسلش‌ها در ویندوز:**
  در PHP ، / در مسیرها روی ویندوز هم قابل قبول است. اما اگر دوست داری کاملاً بی‌تفاوت باشی از DIRECTORY_SEPARATOR استفاده کن.

---

7) یک مثال از اول تا آخر (کنترلر + ویو)

controller/note/show.php

```
<?php
use core\Database;

$db    = new Database($config['database'], 'root', '');
```

```
$id    = $_GET['id'] ?? null;

$note = $db->query(
    "SELECT * FROM notes WHERE id = :id AND user_id =
:uid",
    ['id' => $id, 'uid' => 3]
)->findOrFail();

view('note/show.view.php', [
    'heading' => 'Note',
    'note'    => $note
]);
```

controller/views/note/show.view.php

```
<?php require
base_path('controller/views/partials/nav.php'); ?>
<?php require
base_path('controller/views/partials/header.php'); ?>

<p><?= htmlspecialchars($note['body']) ?></p>

<?php require
base_path('controller/views/partials/footer.php'); ?>
```

(8جمع‌بندی «چی شد و چرا؟»

- با **namespace**ها هر کلاس به فضای نام خودش رفت تا تداخل نام نداشته باشیم و کدها منظم‌تر شوند.
- با **autoload**، دیگر نیازی نیست هر کلاس را دستی `require`کنیم؛ فقط کافی است نام کامل کلاس (`core\Database`) را استفاده کنیم.
- چون در namespace هستیم، برای کلاس‌های داخلی PHP مثل `PDO`باید از **گلوبال** استفاده کنیم.`\PDO` :
- مسیرها را بر اساس نام کلاس می‌سازیم (PSR-4 ساده‌شده).

۱۰ سؤال سخت از جلسه ۳۱ همراه پاسخ

سؤال ۱:
وقتی در فایل `Database.php` **فضای نام** `namespace core;` **تعریف شده، چرا** `new Database()` **باعث خطای** "class not found" **می‌شود؟**
جواب :چون بدون `use core\Database;` **یا** `\core\Database`**، PHP به دنبال کلاس** `Database` **در فضای نام گلوبال می‌گردد، در حالی‌که کلاس در فضای نام** `core` **قرار دارد.**

سؤال ۲:
چه تفاوتی بین `\PDO` **و** `PDO` **وجود دارد داخل فایل تحت** namespace **؟**
جواب `PDO` **:در فضای نام فعلی (**`core`**) جستجو می‌شود، اما** `\PDO` **به طور مستقیم کلاس** `PDO` **را از فضای نام جهانی (گلوبال) فرا می‌خواند.**

سؤال ۳:
چرا لازم است namespace **پروژه با ساختار پوشه‌ها یکسان باشد؟**
جواب :زیرا سیستم اتولود ساده‌ای که تعریف شده به نام‌ها نیاز دارد که مستقیم قابل تبدیل به مسیر فایل باشند (با `str_replace('\\', '/', ...)`**). اگر نام پوشه و** namespace **یکی نباشند، اتولودر فایل را پیدا نمی‌کند.**

سؤال ۴:
وظیفه خط زیر در Autoload **چیست؟**

```
$class = str_replace("\\", "/", $class);
```

جواب :علامت `\` **در** namespace **را به** `/` **تبدیل می‌کند تا بتوان مسیر فایل واقعی را ساخت (مثل** `core/Database.php`**).**

سؤال ۵:
اگر کلاس `Validator` **در فضای نام قرار دارد ولی در کنترلر از آن استفاده نشده است، چه خطایی رخ می‌دهد؟**
جواب :خطای `Class "Validator" not found` **می‌دهد، زیرا یا باید از** `use \core\Validator;` **استفاده شود یا نام کامل** `core\Validator` **.**

سؤال ۶:
اگر اتولوڈر نیامده باشد، ولی از کلاس `Database` **با** `use core\Database;` **استفاده شده باشد، چه اتفاقی می‌افتد؟**
جواب :خطای `file not found` **یا** `require_once` **رخ می‌دهد چرا که فایل کلاس autoload نشده و دستی** `require` **هم نشده است.**

سؤال ۷:
چرا برای فایل‌های فانکشن، تابعی از نوع autoload **کار نمی‌کند؟**
جواب Autoload **:فقط کلاس‌ها را بارگذاری می‌کند. برای فایل‌هایی که شامل توابع هستند (بدون کلاس)، باید از** `require` **دستی استفاده شود.**

سؤال ۸:
چه زمانی استفاده از `use` **در بالای فایل ضروری است؟ مثال بزن.**
جواب :وقتی از کلاس یا فضای نامی در فایل استفاده می‌کنیم، برای جلوگیری از نوشتن نام کامل. مثال:

```
use core\Database;
$db = new Database(...);
```

سؤال ۹:

چرا خطا "Class core\PDO not found" رخ داد و در لحظه چه باید می‌کردیم؟

جواب :کلاس `PDO` در فضای نام `core` نبوده اما بدون `\` یا `use PDO;` فراخوانی شده است. باید `new \PDO(...)` یا `use PDO;` استفاده شود.

سؤال ۱۰:

نحوه رفع خطا در اتولدینگ زمانی که کلاس `Response` داخل `core/Response.php` و داخل فضای نام `core` است به چه شکل است؟

جواب :در فایلی که می‌خواهیم از آن استفاده کنیم باید بنویسیم:

```
use core\Response;
new Response();
```

یا مستقیماً:

```
new \core\Response();
```