

1 تمرین — تشخیص مشکل در فضای نام

کد زیر را ببین و بگو چرا کار نمی‌کند؟

```
namespace Core;

class Database {
    public function connect() {
        echo "Connected!";
    }
}

$db = new Database();
$db->connect();
```

پاسخ:

خطا می‌دهد چون وقتی داخل یک فضای نام هستیم و کلاس را خارج از آن صدا می‌زنیم، باید یا مسیر کامل (Core\Database) را بدهیم یا از use استفاده کنیم.
نسخه درست:

```
use Core\Database;

$db = new Database();
$db->connect();
```

2 تمرین — ارجاع به کلاس‌های گلوبال

کد زیر را ببین:

```
namespace Core;

class MyPDO {
    public function __construct() {
        $pdo = new
PDO("mysql:host=localhost;dbname=test", "root", "");
    }
}
```

سؤال: خطای این کد چیست؟

پاسخ:

چون داخل namespace هستیم، PDO را PHP به شکل Core\PDO جستجو می‌کند که وجود ندارد. باید از گلوبال با PDO استفاده کنیم:

```
$pdo = new \PDO("mysql:host=localhost;dbname=test", "root", "");
```

3 تمرین — تطابق مسیر و namespace

فایل زیر در مسیر app/Models/User.php است:

```
namespace App;

class User {}
```

سؤال: آیا این ساختار درست است؟

پاسخ: خیر. اگر استاندارد PSR-4 را رعایت کنیم، namespace باید مسیر پوشه را نشان دهد، پس باید باشد:

```
namespace App\Models;
```

4 تمرین — استفاده از spl_autoload_register

کد زیر را کامل کن تا هر کلاس را با توجه به namespace لود کند:

```
spl_autoload_register(function ($class) {
    // اینجا کد رو کامل کن
});
```

پاسخ:

```
spl_autoload_register(function ($class) {
    $path = str_replace("\\", "/", $class) . ".php";
    if (file_exists($path)) {
        require $path;
    }
});
```

5 تمرین — ایمپورت چند کلاس

فرض کن دو کلاس داری:

```
namespace App\Models;
class User {}

namespace App\Controllers;
class User {}
```

سؤال: چطور هر دو کلاس را در یک فایل ایمپورت و استفاده کنیم؟
پاسخ:

```
use App\Models\User as ModelUser;
use App\Controllers\User as ControllerUser;

$modelUser = new ModelUser();
$controllerUser = new ControllerUser();
```

6 تمرین — مشکل مسیر در Autoload

اگر namespace این باشد:

```
namespace Core\Database;
```

و مسیر فایل `/core/Database.php` باشد، Autoload جواب می‌دهد؟
پاسخ: خیر. باید مسیر فایل `/core/Database/` باشد و داخلش `Database.php` باشد، چون هر `\` در namespace به یک پوشه ترجمه می‌شود.

7 تمرین namespace — تو در تو

کدی بنویس که کلاس زیر در فضای نام `App\Services\Payment` تعریف شود و پیغام `Payment started` را چاپ کند.

پاسخ:

```
namespace App\Services\Payment;

class Processor {
    public function start() {
        echo "Payment started";
    }
}

$processor = new Processor();
$processor->start();
```

8 تمرین — استفاده از namespace پیش فرض

اگر در یک فایل هیچ namespace ننویسیم، PHP آن را در چه فضای نامی قرار می‌دهد؟
پاسخ: در فضای نام پیش فرض. (Global Namespace)

9 تمرین — خطای دوبار تعریف کلاس

چرا استفاده از namespace باعث می‌شود خطای زیر کمتر پیش بیاید؟

```
Fatal error: Cannot redeclare class User
```

پاسخ: چون namespace نام کامل کلاس را شامل مسیرش می‌کند، مثل `App\Models\User` و `App\Controllers\User`، که دو کلاس متفاوت هستند.

10 تمرین — ترکیب Autoload و namespace

کدی بنویس که کلاس `App\Helpers\Str` در مسیر `app/Helpers/Str.php` باشد و توسط `autoload` لود شود.
پاسخ:

```
// autoload.php
spl_autoload_register(function ($class) {
    $path = str_replace("\\", "/", $class) . ".php";
    if (file_exists($path)) {
        require $path;
    }
});

// app/Helpers/Str.php
namespace App\Helpers;

class Str {
    public static function upper($string) {
        return strtoupper($string);
    }
}

// index.php
require 'autoload.php';

use App\Helpers\Str;
echo Str::upper("hello");
```

اگر بخوای، می‌تونم همین تمرین‌ها رو برات به سبک **امتحان واقعی** طراحی کنم، یعنی بدون جواب بدی و بعد با جواب مقایسه کنیم تا ببینی چند درصد جلسه ۳۱ رو بلدی. اینطوری دقیق محک می‌خوریم.

You said:

جواب ۷ رو اشتباه ندادی چون پایین باید use استفاده میکنیم

ChatGPT said:

درست می‌گی اگر تو فایل/فضای نام دیگه‌ای بسازی و استفاده کنی باید use بیاری (یا نام کامل بدی).

اما جواب تمرین ۷ من کفش داخل یک namespace واحد بود؛ وقتی در همون فایل و همون namespace هستیم، نیاز به use نیست.

دو حالت درست

(1) همه چیز در یک فایل و یک namespace بدون use)

```
<?php
namespace App\Services\Payment;

class Processor {
    public function start() { echo "Payment started"; }
}

// همین فایل، همین namespace
$processor = new Processor(); // use نیازی به
$processor->start();
```

(2) تعریف کلاس در یک فایل، استفاده در فایل دیگر (با use)

app/Services/Payment/Processor.php

```
<?php
namespace App\Services\Payment;

class Processor {
    public function start() { echo "Payment started"; }
}
```

index.php

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

use App\Services\Payment\Processor;
```

```
$processor = new Processor(); // اینجا چون فایل  
لازم است use فرق داره، namespace جداست  
$processor->start();
```

نکته: به جای use می‌تونیم نام کامل بدیم:

```
$processor = new \App\Services\Payment\Processor();
```

You said:

پس چرا تو تمرین یک گفتی خطاست باید از use استفاده کنی

ChatGPT said:

Thought for 12s

حق با توئه—من اون جا قاطی کردم، ببخش 🙏

قاعدهٔ دقیق:

- همان فایل و همان namespace: نیاز به use نیست.
- namespace App\Services;
-
- class Mailer {}
- \$m = new Mailer(); // بدون use
- فایل namespace/متفاوت: باید یا use بیاری یا نام کامل (FQN) بنویسی.

```
// File A: App/Services/Mailer.php  
namespace App\Services;  
class Mailer {}
```

```
// File B: index.php  
use App\Services\Mailer;  
$m = new Mailer(); // اوکی با use  
// یا:  
$m = new \App\Services\Mailer(); // FQN اوکی با
```

- کلاس‌های داخلی (PHP مثل PDO) در فضای نام دار:

اگر داخل `namespace Core;` هستی و بنویسی `new PDO, PHP` در فایل `new` دنبال `Core\PDO` می‌گردد و خطا می‌گیری. باید:

```
namespace Core;
use PDO;
$pdo = new PDO($dsn, $u, $p);
```

پس اگر در تمرین ۱ همه چیز تو همان `namespace` و همان فایل بود، گفتن «باید `use` بیاری» اشتباه از طرف من بود. اون توصیه فقط وقتی لازمه که از `namespace` دیگه/فایل دیگه اون کلاس رو مصرف کنی، یا وقتی داری از کلاس‌های داخلی `PHP` داخل یک `namespace` استفاده می‌کنی.

You said:

استاندارد PSR-4 چیه

ChatGPT said:

استاندارد PSR-4 در واقع یک قرارداد (Convention) هست که توسط `PHP-FIG` معرفی شده تا مشخص کنه وقتی داری یک کلاس `PHP` رو `autoload` می‌کنی، پوشه‌ها و `namespace`‌ها چطور باید به هم نگاشت بشن.

یعنی چی؟ یعنی با این استاندارد، `PHP` و `Autoloader` می‌فهمه وقتی میگی:

```
use App\Controllers\HomeController;
```

باید بره کدوم مسیر روی هارد رو بگرده و فایل رو لود کنه.

📌 تعریف رسمی PSR-4

→ `Namespace prefix + Relative Class Name` مسیر پوشه + نام فایل

- Namespace ها باید با ساختار پوشه‌ها هم‌خوان باشند.
- نام فایل باید دقیقاً با نام کلاس (Case-Sensitive) یکسان باشد.
- همه کلاس‌ها باید داخل یک namespace باشند (نه global namespace).

◆ مثال ساده

فرض کن در `composer.json` این رو تعریف کردی:

```
"autoload": {
    "psr-4": {
        "App\\": "src/"
    }
}
```

📁 ساختار پوشه‌ها:

```
project/
  src/
    Controllers/
      HomeController.php
```

📄 فایل: `HomeController.php`

```
<?php
namespace App\Controllers;

class HomeController {
    public function index() {
        echo "Home Page";
    }
}
```

📄 فایل: `index.php`

```
<?php
require __DIR__ . '/vendor/autoload.php';
```

```
use App\Controllers\HomeController;

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

نتیجه: وقتی Composer می‌بیند `App\Controllers\HomeController`، بر اساس PSR-4 می‌ره دنبال مسیر:

```
src/Controllers/HomeController.php
```

◆ قوانین مهم PSR-4

۱. هر namespace باید با مسیر پوشه‌ها هماهنگ باشد.
مثلاً `App\Models\User` باید در `src/Models/User.php` باشد.
 ۲. نام فایل باید دقیقاً مثل نام کلاس باشد.
 ۳. از `backslash (\)` برای جدا کردن namespace ها استفاده می‌کنیم، ولی توی پوشه واقعی `/` (یا `\` بسته به سیستم عامل).
 ۴. باید از autoloader مثلاً (Composer) استفاده کنی تا این ساختار اجرا بشه.
-

🔗 مزایای PSR-4

- نظم و هماهنگی در پروژه
- قابلیت autoload کلاس‌ها بدون `include` یا `require`
- سازگار با همهٔ پروژه‌ها و فریم‌ورک‌های مدرن (Laravel, Symfony, ...)