Deepseek

سوال ۱ : هدف اصلی استفاده از Middleware در این جلسه چه بود؟

جواب :برای کنترل دسترسی به صفحات مختلف بر اساس وضعیت کاربر (لاگین کرده یا نکرده). یعنی جلوگیری از دسترسی کاربران لاگین‌کرده به صفحه ثبت‌نام و جلوگیری از دسترسی کاربران لاگین‌نکرده به صفحات داخلی مثل یادداشت‌ها.

سوال ۲: متد `only()` چه کاری انجام می‌دهد؟

جواب :این متد یک کلید (مثل `'guest'` یا `'auth'`) می‌گیرد و آن را در آرایهٔ مسیر مربوطه، در کلید `middleware` ذخیره می‌کند تا بعداً در هنگام درخواست، بررسی شود.

سوال ۳: برای اینکه بتوانیم متدها را به صورت زنجیره‌ای (مثل `->get()->only()`) صدا بزنیم، باید چه تغییری در متد `add()` بدهیم؟

جواب :باید در انتهای متد `add()`، عبارت `return $this;` را قرار دهیم. این کار باعث می‌شود متد، شیء فعلی (`Router`) را برگرداند و بتوانیم متد دیگری را بلافاصله روی آن فراخوانی کنیم.

سوال ۴: تفاوت `auth` و `guest` در Middleware چیست؟

جواب:

- `auth`: برای صفحاتی است که **فقط کاربران لاگین‌کرده** می‌توانند به آنها دسترسی داشته باشند (مثل صفحه یادداشت‌ها).
- `guest`: برای صفحاتی است که **فقط کاربران لاگین‌نکرده** (مهمان) می‌توانند به آنها دسترسی داشته باشند (مثل صفحه ثبت‌نام).

سوال ۵: در کدام قسمت از کد، Middleware بررسی و اجرا می‌شود؟

جواب: در متد `route()` از کلاس `Router`، دقیقاً قبل از خطی که کنترلر بارگذاری می‌شود.(`require_once`)

سوال ۶: اگر کاربری که لاگین کرده است بخواهد به صفحه ثبت‌نام (`/register`) دسترسی پیدا کند، چه اتفاقی می‌افتد؟

جواب: به دلیل وجود `guest` Middleware ، شرط `if ($_SESSION['user'] ?? false)` برقرار شده و کاربر به صفحه اصلی (`/`) هدایت (Redirect) می‌شود.

سوال ۷: ثابت `MAP` (const) در کلاس `Middleware` چه کاربردی دارد؟

جواب: این ثابت یک آرایه است که نقشهٔ یا Lookup Table را نگهداری می‌کند. این نقشه، کلیدهای Middleware مثل (`'auth'`) را به نام کلاس‌های مربوطه (مثل `Auth::class`) ارتباط می‌دهد.

سوال ۸: اگر در مسیری یک کلید Middleware تعریف کنیم که در `MAP`وجود نداشته باشد (مثلاً `->only('admin')`)، چه اتفاقی می‌افتد؟

جواب :متد `resolve()`کلاس `Middleware`با خطای `Exception`متوقف می‌شود و پیغام "No matching middleware found for key 'admin'" را نمایش می‌دهد.

سوال ۹: دلیل اصلی خارج کردن منطق چک کردن سشن از داخل `Router`و قرار دادن آن در کلاس‌های جداگانه مثل `Auth`و `Guest`چیست؟

جواب :برای رعایت اصل **Separation of Concerns**(جداسازی نگرانی‌ها). این کار باعث تمیزتر شدن کد `Router`، قابلیت استفادهٔ مجاز از منطق و امکان اضافه کردن Middlewareهای جدید بدون دستکاری `Router`می‌شود.

سوال ۱۰: دستور `(new $middleware)->handle();`چه کاری انجام می‌دهد؟

جواب :این دستور به صورت داینامیک:

۱. یک شیء از کلاس Middleware مورد نظر (مثلاً `new Auth()`)می‌سازد.
۲. متد `handle()`آن شیء را فراخوانی می‌کند تا منطق بررسی اجرا شود (مثلاً چک کردن وجود سشن).

Claude

سوال ۱: مشکل اصلی که در ابتدای جلسه ۳۹ مطرح شد چه بود؟

جواب :مشکل این بود که وقتی کاربری ثبت نام کرده و وارد سیستم شده، هنوز هم می‌توانست به صفحه ثبت نام (register) دسترسی داشته باشد. این باعث می‌شد که یک

کاربر وارد شده بتواند حساب کاربری جدیدی ایجاد کند که منطقی نیست و باعث ایجاد رکوردهای اضافی در دیتابیس می‌شود.

سوال ۲ Middleware :چیست و چه نقشی در وب اپلیکیشن دارد؟

جواب Middleware **:**مانند یک "پل" یا "واسطه" است که بین درخواست کاربر و اصل برنامه قرار می‌گیرد. وظایف اصلی آن:

- **بررسی مجوز دسترسی** :آیا کاربر اجازه دسترسی به این صفحه را دارد؟
- **تأیید هویت** :آیا کاربر وارد سیستم شده است؟
- **اعمال محدودیت** :مثلاً تأیید ایمیل، پرداخت حق عضویت و...
- **هدایت کاربر** :در صورت عدم مجوز، کاربر را به صفحه مناسب هدایت کند

سوال ۳: چرا باید در متد `add` کلاس Router مقدار `return $this;` را اضافه کرد؟

جواب :برای فعال کردن **Method Chaining** یا زنجیره‌سازی متدها. با برگرداندن `$this`، می‌توانیم چندین متد را پشت سر هم فراخوانی کنیم:

php

```
// بدون Method Chaining
$router->get('/register', 'controller.php');
$router->only('guest'); // این کار نمی‌کند


// با Method Chaining
$router->get('/register', 'controller.php')->only('guest');
```

سوال ۴: کلید `middleware` که در آرایه routes اضافه شد چه نقشی دارد؟

جواب : کلید `middleware` برای ذخیره نوع کنترل دسترسی هر route استفاده می‌شود:

```php
$this->routes[] = [
    'uri' => $uri,
    'controller' => $controller,
    'method' => $method,
    'middleware' => null  // مقدار پیش‌فرض null است
];
```

وقتی متد `only()` فراخوانی می‌شود، این مقدار تغییر می‌کند:

- `'middleware' => 'guest'` → فقط مهمان‌ها
- `'middleware' => 'auth'` → فقط کاربران وارد شده

سوال ۵: تابع `array_key_last()` در متد `only` چه کاری انجام می‌دهد؟

جواب `array_key_last()` **:** شاخص آخرین المان آرایه `$this->routes` را برمی‌گرداند. چون وقتی `only()` فراخوانی می‌شود، می‌خواهیم middleware را به آخرین routeای که اضافه شده اعمال کنیم:

```php
public function only($key) {
    // آخرین route و middleware اضافه شده را پیدا کن آن را تنظیم کن
    $this->routes[array_key_last($this->routes)]['middleware'] = $key;
    return $this;
}
```

سوال ۶: منطق کنترل دسترسی برای middleware نوع 'guest' چگونه کار می‌کند؟

جواب :برای middleware نوع:'guest'

```
php
if ($route['middleware'] == 'guest') {
    if ($_SESSION['user'] ?? false) {  // اگر کاربر وارد شده باشد
        header('location: /');        // او را به خانه بفرست
        exit;                         // و اجرای کد را متوقف کن
    }
}
```

منطق :اگر کاربر قبلاً وارد شده، نباید به صفحات مخصوص مهمان‌ها) مثل login یا register) دسترسی داشته باشد.

سوال ۷: چرا کلاس‌های Auth و Guest در پوشه جداگانه ایجاد شدند؟

جواب :برای بهتر شدن ساختار کد و اعمال اصل :**Separation of Concerns**

۱. **تمیزی کد** :هر middleware مسئولیت مشخصی دارد
۲. **قابلیت نگهداری** :تغییر در یک middleware روی دیگری تأثیر نمی‌گذارد
۳. **قابلیت توسعه** :افزودن middleware جدید آسان‌تر می‌شود
۴. **تست‌پذیری** :هر کلاس را می‌توان جداگانه تست کرد

```
core/
middleware/
    → Auth.php      کنترل دسترسی کاربران وارد شده
    → Guest.php     کنترل دسترسی مهمان‌ها
    → Middleware.php  مدیریت کلی middleware ها
```

سوال ۸: کلاس Middleware و ثابت MAP چه نقشی دارند؟

جواب: کلاس `Middleware` مانند یک **مدیر** عمل می‌کند که middleware ها را سازماندهی می‌کند:

```php
class Middleware {
    public const MAP = [
        'guest' => Guest::class,    // کلید 'guest' به کلاس Guest اشاره می‌کند
        'auth' => Auth::class       // کلید 'auth' به کلاس Auth اشاره می‌کند
    ];
}
```

مزایا:

- **مدیریت متمرکز**: همه middleware ها در یک جا تعریف شده‌اند
- **انعطاف‌پذیری**: افزودن middleware جدید فقط نیاز به اضافه کردن یک خط دارد
- **جلوگیری از تکرار**: نیازی به if/else های متعدد نیست

سوال ۹: متد استاتیک `resolve` چگونه کار می‌کند؟

جواب: متد `resolve` مسئول تشخیص و اجرای middleware مناسب است:

```php
public static function resolve($key) {
    if (!$key) {
        return;  // اگر middleware نداریم، کاری نکن
    }
```

```
    $middleware = static::MAP[$key] ?? false;  // middleware مربوطه را پیدا کن

    if (!$middleware) {
        throw new \Exception("Middleware not found for key: " . $key);
    }

    (new $middleware)->handle();  // را فراخوانی کن handle نمونه‌ای از کلاس بساز و
}
```

مراحل:

۱. بررسی وجود کلید
۲. یافتن کلاس مربوطه از MAP
۳. کنترل وجود کلاس
۴. ایجاد نمونه و اجرای متد handle

سوال ۱۰: چه اتفاقی می‌افتد اگر middleware نامعلومی مثل 'foobar' استفاده کنیم؟

جواب : اگر middleware نامعلومی استفاده کنیم:

```php
$router->get('/test', 'controller.php')->only('foobar');
```

متد `resolve` Exception پرتاب می‌کند:

```php
if (!$middleware) {
    throw new \Exception("Middleware not found for key: foobar");
}
```

دلیل این رفتار:

- **امنیت** :جلوگیری از اجرای کدهای ناشناخته
- **دیباگ** :کمک به توسعه‌دهنده برای پیدا کردن خطاهای تایپی
- **پایداری** :اطمینان از اینکه فقط middleware های تعریف شده اجرا می‌شوند

راه حل :اضافه کردن middleware جدید به:MAP

```php
const MAP = [
    'guest' => Guest::class,
    'auth' => Auth::class,
    'foobar' => FooBar::class  // جدید middleware اضافه کردن
];
```

Chatgpt

سوال ۱

چرا در متد `add` **به آرایه** route **ها یک کلید** `middleware` **اضافه کردیم؟**

✅جواب:
چون هر route **ممکنه فقط برای یک نوع کاربر (مثلاً** **auth** **یا** **guest**) **قابل دسترسی باشه، پس نیاز داریم** middleware **مربوط به اون** route **رو ذخیره کنیم.**

? سوال ۲

دلیل استفاده از `array_key_last($this->routes)` **در متد** `only` **چیست؟**

✅جواب:

برای اینه که همیشه آخرین route اضافه شده توسط کاربر رو بگیریم و مقدار middleware اون رو تغییر بدیم.

؟ سوال ۳

متد `only`چه چیزی برمی‌گرداند و چرا؟

✅جواب:

متد `only`هم مثل `add`مقدار `$this`رو برمی‌گرداند تا بتونیم از chain method استفاده کنیم (مثلاً:

```
$router->get('login', 'controllers/login.php')-
>only('guest');
```)

؟ سوال ۴

در متد `route` ، چرا باید `middleware` ها رو قبل از `return` کردن `controller`بررسی کنیم؟

✅جواب :

چون اگه کاربر اجازه دسترسی نداشته باشه، نباید اصلاً `controller` مربوط به اون `route`اجرا بشه .

؟ سوال ۵

در شرط زیر چه اتفاقی می‌افتد؟

```
if ($route['middleware'] === 'guest' &&
isset($_SESSION['user'])) {
    header("Location: /");
    exit;
}
```

✅جواب:
اگه route فقط مخصوص مهمان باشه ولی کاربر login کرده باشه، او رو به صفحه اصلی هدایت می‌کنیم.

❓ سوال ۶

اگر برای route ای `->only('auth')`–تعریف کنیم، چه شرایطی برای دسترسی لازم است؟

✅جواب:
کاربر باید وارد شده باشد (`$_SESSION['user']` ست شده باشد.)

❓ سوال ۷

اگر هیچ middleware ای برای route تنظیم نشود چه می‌شود؟

✅جواب:
routeبه‌صورت آزاد در دسترس خواهد بود و همه (مهمان و کاربر لاگین‌شده) می‌توانند آن را ببینند.

❓ سوال ۸

چرا از `exit`بعد از `header("Location: ...")`استفاده کردیم؟

✅جواب:
چون باید اجرای اسکریپت را متوقف کنیم وگرنه controller همچنان اجرا می‌شود حتی بعد از ریدایرکت.

? سوال ۹

تفاوت بین middleware نوع **auth** و **guest**چیست؟

✅جواب:

- **auth**: فقط کاربر لاگین‌شده دسترسی دارد.
- **guest**: فقط کاربر مهمان) بدون (login دسترسی دارد.

? سوال ۱۰

چرا متد `only`داخل کلاس Router تعریف شد نه جای دیگر؟

✅جواب:
چون middleware باید مستقیماً به route متصل شود و فقط Router می‌تواند تشخیص دهد که کدام route به چه middleware ای نیاز دارد.

Preplexity

سؤال‌ها و پاسخ‌ها

سؤال: چرا نباید اجازه داد کاربر ثبت‌نام کرده دوباره صفحه ثبت‌نام را ببیند؟
پاسخ: زیرا ممکن است کاربر وارد شده مجدد ثبت‌نام کند و منطق سایت بهم بریزد یا کاربران غیرواقعی ایجاد شوند.

سؤال :عملکرد middleware چیست؟
پاسخ :مانند پلی بین درخواست ورودی و هسته اپلیکیشن عمل می‌کند و دسترسی را براساس شرایط بررسی می‌کند.

سؤال :راه اولیه حل محدود کردن دسترسی صفحات چه بود؟
پاسخ :قرار دادن شرط و انتقال (redirect) در هر کنترلر به طور مستقل بود.

سؤال :اشکال روش شرطگذاری در هر کنترلر چیست؟
پاسخ :پراکندگی و تکرار کد در بخش‌های مختلف برنامه.

سؤال :چگونه می‌توان این منطق را به سطح route منتقل کرد؟
پاسخ :با اضافه کردن متد only به هر route و تعیین نوع دسترسی.

سؤال :بعد از اضافه کردنonly ، چرا باید متد add و سایر متدهای مشابه مقدار this را returnکنند؟
پاسخ :تا chain کردن متدها و استفاده از only ممکن شود.

سؤال purpose :متد only در Router چیست؟
پاسخ :افزودن کلید middleware به آخرین route ثبت‌شده.

سؤال :چرا مقدار پیش‌فرض middleware برای هر route باید null باشد؟
پاسخ :چون همهroute ها نیاز به middleware ندارند.

سؤال: چه شرطی در متد route برای مهمان قرار داده می‌شود؟
پاسخ: اگر middleware مقدار guest داشت و session کاربر مقدار داشت، به صفحه اصلی منتقل شود.

سؤال: چه شرطی در متد route برای کاربر Auth گذاشته می‌شود؟
پاسخ: اگر middleware مقدار auth داشت و session کاربر مقدار نداشت، به صفحه اصلی منتقل شود.

سؤال: چرا باید کدهای شرط middleware را به فایل مستقل منتقل کنیم؟
پاسخ: برای تمیزتر شدن کد و رعایت اصل.Single Responsibility

سؤال: در پوشه middleware چه کلاس‌هایی باید ساخته شود؟
پاسخ: کلاس auth و guest که هر دو متد handle دارند.

سؤال: وظیفه متد handle در کلاس Guest چیست؟
پاسخ: چک می‌کند اگر session کاربر وجود داشت، درخواست را redirect می‌کند.

سؤال: وظیفه متد handle در کلاس Auth چیست؟
پاسخ: چک می‌کند اگر session کاربر وجود نداشت، درخواست را redirect می‌کند.

سؤال :ساختار کلاس مرکزی Middleware چیست؟
پاسخ :یک آرایه MAP که کلیدهای middleware به کلاس‌ها نگاشت داده می‌شوند و یک متد resolve برای اجرا دارد.

سؤال :متد resolve در Middleware چه کاری انجام می‌دهد؟
پاسخ :با گرفتن کلید، کلاس مورد نظر را Instantiate و متد handle را روی آن فراخوانی می‌کند.

سؤال :اگر کلید middleware اشتباه باشد چه اتفاقی می‌افتد؟
پاسخ :متد resolve یک Exception با پیام مناسب پرتاب می‌کند.

سؤال :اگر بخواهیم middleware جدید اضافه کنیم چه باید کرد؟
پاسخ :کلاس جدید می‌سازیم و کلیدش را به MAP اضافه و در routes آن را فراخوانی می‌کنیم.

سؤال :با این ساختار چگونه کد قابل نگهداری‌تر می‌شود؟
پاسخ :چون تمام لاجیک middleware در یک بخش مجتمع است و توسعه و رفع اشکال آسان می‌شود.

سؤال :کاربرد عملی این سیستم برای مدیریت صفحاتی مثل ثبت‌نام و یادداشت‌ها چیست؟
پاسخ :صفحه ثبت‌نام فقط برای مهمان و صفحه یادداشت‌ها فقط برای کاربران عضو نمایش داده می‌شود.

تمرین

Deepseek

تمرین ۱: پیاده‌سازی متد `delete` در Router

متد `delete` را به کلاس `Router` اضافه کنید تا بتوانید مسیرهای `DELETE` را تعریف کنید و از قابلیت chaining هم پشتیبانی کند.

php
```php
// پاسخ
public function delete($uri, $controller) {
    return $this->add('DELETE', $uri, $controller);
}
```

تمرین ۲: ایجاد Middleware برای بررسی ادمین

یک Middleware جدید به نام `Admin` ایجاد کنید که بررسی کند آیا کاربر در سشن دارای `is_admin = true` است یا نه. اگر نه، به صفحه اصلی redirect کند.

php
```php
// پاسخ در Core/Middleware/Admin.php
<?php

namespace Core\Middleware;

class Admin {
    public function handle() {
        if (!isset($_SESSION['user']) || $_SESSION['use
r']['is_admin'] !== true) {
            header('location: /');
            exit();
        }
    }
}
```

تمرین ۳: ثبت Middleware جدید در MAP

Middlewareای که در تمرین قبل ساختی را در آرایه `MAP` کلاس `Middleware` ثبت کن.

```php
// پاسخ
public const MAP = [
    'guest' => Guest::class,
    'auth' => Auth::class,
    'admin' => Admin::class // اضافه شد
];
```

تمرین ۴: ایجاد مسیر با Middleware admin

یک مسیر جدید ایجاد کن که فقط برای ادمین‌ها قابل دسترسی باشد.

```php
// پاسخ در routes.php
$router->get('/admin/dashboard', 'controllers/admin/dashboard.php')->only('admin');
```

تمرین ۵: تغییر رفتار redirect در Middleware auth

Middleware `Auth` را طوری تغییر بده که به جای صفحه اصلی، کاربر به صفحه لاگین (`/login`) redirect شود.

```php
// پاسخ در Core/Middleware/Auth.php
public function handle() {
    if (!isset($_SESSION['user'])) {
        header('location: /login'); // تغییر یافت
        exit();
    }
}
```

تمرین ۶: اضافه کردن متد patch به Router

متد `patch` را به Router اضافه کنید.

```php
// پاسخ
public function patch($uri, $controller) {
    return $this->add('PATCH', $uri, $controller);
}
```

تمرین ۷: ایجاد Middleware برای بررسی ایمیل تایید شده

یک Middleware جدید به نام `Verified` ایجاد کنید که بررسی کند آیا کاربر ایمیل خود را تایید کرده است یا نه.

```php
// پاسخ در Core/Middleware/Verified.php
<?php

namespace Core\Middleware;

class Verified {
    public function handle() {
        if (!isset($_SESSION['user']) || $_SESSION['user']['email_verified_at'] === null) {
            header('location: /verify-email');
            exit();
        }
    }
}
```

تمرین ۸: استفاده از چندین Middleware برای یک مسیر

سیستم را طوری تغییر دهید که بتوان برای یک مسیر بیش از یک Middleware تعریف کرد.

```php
// پاسخ (تغییر در متد only)
public function only($keys) {
    $lastRouteIndex = array_key_last($this->routes);
    $this->routes[$lastRouteIndex]['middleware'] = (array) $keys;
    return $this;
}

// و تغییر در متد resolve برای پردازش آرایه‌ای از middlewareها
```

تمرین ۹: ایجاد مسیر برای لاگاوت

یک مسیر `POST` برای لاگاوت ایجاد کنید که session را destroy کند.

```php
// پاسخ در routes.php
$router->post('/logout', 'controllers/auth/logout.php')->only('auth');
```

تمرین ۱۰: بهبود پیغام خطای Middleware

پیغام خطای Middleware را به فارسی تغییر دهید و شماره خطا هم به آن اضافه کنید.

```php
// پاسخ در متد resolve
public static function resolve($key) {
    if (!$key) {
        return;
    }

    $middleware = static::MAP[$key] ?? false;

    if (!$middleware) {
        throw new \Exception("میان‌افزار با کلید '{$key}' یافت نشد. (خطا در خط " . __LINE__ . ")");
```

```
        }

        (new $middleware)->handle();
}
```

Claude

تمرین ۱: پیاده‌سازی Method Chaining ساده

صورت سوال: یک کلاس `Calculator` بسازید که بتواند عملیات ریاضی را به صورت زنجیره‌ای انجام دهد.

کد مورد انتظار:

```php
<?php
class Calculator {
    private $result = 0;

    public function add($number) {
        $this->result += $number;
        return $this; // برای method chaining
    }

    public function multiply($number) {
        $this->result *= $number;
        return $this;
    }

    public function getResult() {
        return $this->result;
```

```
    }
}

// استفاده:
$calc = new Calculator();
$result = $calc->add(5)->multiply(2)->add(3)->getResult(); // نتیجه: ۱۳
echo $result;
```

تمرین ۲: ایجاد middleware برای بررسی سن

صورت سوال : یک middleware بنویسید که بررسی کند کاربر بالای ۱۸ سال است یا نه. اگر زیر ۱۸ سال باشد، او را به صفحه خطا هدایت کند.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class AgeRestriction {
    public function handle() {
        $userAge = $_SESSION['user_age'] ?? 0;

        if ($userAge < 18) {
            header('location: /age-restriction-error');
            exit;
        }
    }
}
```

```
// در کلاس Middleware:
const MAP = [
    'guest' => Guest::class,
    'auth' => Auth::class,
    'adult' => AgeRestriction::class
];

// در routes:
$router->get('/adult-content', 'controller/adult.php')->only('adult');
```

تمرین ۳ middleware :چندگانه

صورت سوال :روتر را طوری تغییر دهید که بتواند چندین middleware را همزمان اعمال کند.

کد مورد انتظار:

```
php
<?php
// برای پذیرش آرایه only تغییر متد
public function only($middlewares) {
    $middlewares = is_array($middlewares) ? $middlewares : [$middlewares];
    $this->routes[array_key_last($this->routes)]['middleware'] = $middlewares;
    return $this;
}

// تغییر متد resolve
public static function resolve($middlewares) {
```

```
    if (!$middlewares) return;

    $middlewares = is_array($middlewares) ? $middlewares : [$middlewares];

    foreach ($middlewares as $key) {
        $middleware = static::MAP[$key] ?? false;
        if (!$middleware) {
            throw new \Exception("Middleware not found for key: " . $key);
        }
        (new $middleware)->handle();
    }
}

// استفاده:
$router->get('/vip-section', 'controller/vip.php')->only(['auth', 'adult']);
```

تمرین ۴ middleware: برای بررسی نقش کاربر(Role-based)

صورت سوال: یک middleware بسازید که بر اساس نقش کاربر (admin, user, moderator) دسترسی را کنترل کند.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class RoleMiddleware {
    private $requiredRole;
```

```php
    public function __construct($role) {
        $this->requiredRole = $role;
    }

    public function handle() {
        $userRole = $_SESSION['user_role'] ?? 'guest';

        $roleHierarchy = [
            'guest' => 0,
            'user' => 1,
            'moderator' => 2,
            'admin' => 3
        ];

        $userLevel = $roleHierarchy[$userRole] ?? 0;
        $requiredLevel = $roleHierarchy[$this->requiredRole] ?? 999;

        if ($userLevel < $requiredLevel) {
            header('location: /access-denied');
            exit;
        }
    }
}

// استفاده:
class AdminOnly extends RoleMiddleware {
    public function __construct() {
        parent::__construct('admin');
    }
```

}

تمرین ۵: ایجاد middleware برای محدودیت تعداد درخواست(Rate Limiting)

صورت سوال: middleware ای بسازید که تعداد درخواست‌های کاربر را در هر دقیقه محدود کند.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class RateLimiter {
    private $maxRequests = 60; // حداکثر ۶۰ درخواست در دقیقه

    public function handle() {
        $userIp = $_SERVER['REMOTE_ADDR'];
        $currentTime = time();
        $windowStart = $currentTime - 60; // 1 دقیقه قبل

        // شمارش درخواست‌های اخیر (معمولاً از cache یا دیتابیس)
        if (!isset($_SESSION['requests_' . $userIp])) {
            $_SESSION['requests_' . $userIp] = [];
        }

        $requests = $_SESSION['requests_' . $userIp];

        // پاک کردن درخواست‌های قدیمی
```

```
        $requests = array_filter($requests, function($timestamp) use ($windowStart)
{
            return $timestamp > $windowStart;
        });

        if (count($requests) >= $this->maxRequests) {
            http_response_code(429); // Too Many Requests
            echo "Rate limit exceeded. Try again later.";
            exit;
        }

        // اضافه کردن درخواست فعلی
        $requests[] = $currentTime;
        $_SESSION['requests_' . $userIp] = $requests;
    }
}
```

تمرین ۶ middleware: برای لاگ کردن درخواست‌ها

صورت سوال: یک middleware بسازید که تمام درخواست‌ها را در فایل لاگ ثبت کند.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class RequestLogger {
    public function handle() {
        $logData = [
```

```
            'timestamp' => date('Y-m-d H:i:s'),
            'ip' => $_SERVER['REMOTE_ADDR'],
            'method' => $_SERVER['REQUEST_METHOD'],
            'uri' => $_SERVER['REQUEST_URI'],
            'user_agent' => $_SERVER['HTTP_USER_AGENT'] ?? 'Unknown'
        ];

        $logLine = sprintf(
            "[%s] %s %s %s - %s\n",
            $logData['timestamp'],
            $logData['ip'],
            $logData['method'],
            $logData['uri'],
            $logData['user_agent']
        );

        file_put_contents(
            __DIR__ . '/../../logs/requests.log',
            $logLine,
            FILE_APPEND | LOCK_EX
        );
    }
}

//این middleware باید در همه routes اجرا شود
//می‌توان آن را به صورت global در ابتدای متد route اضافه کرد
```

تمرین ۷ middleware برای: CSRF Protection

صورت سوال : یک middleware برای محافظت در برابر حملات CSRF بسازید.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class CSRFProtection {
    public function handle() {
        // فقط برای درخواست‌های POST, PUT, DELETE
        if (!in_array($_SERVER['REQUEST_METHOD'], ['POST', 'PUT', 'DELETE'])) {
            return;
        }

        $submittedToken = $_POST['csrf_token'] ?? $_SERVER['HTTP_X_CSRF_TOKEN'] ?? '';
        $sessionToken = $_SESSION['csrf_token'] ?? '';

        if (empty($sessionToken)) {
            throw new \Exception('CSRF token not found in session');
        }

        if (!hash_equals($sessionToken, $submittedToken)) {
            http_response_code(403);
            die('CSRF token mismatch');
        }
```

```
    }
}

// تابع کمکی برای تولید توکن
function csrf_token() {
    if (!isset($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf_token'];
}

// در فرم‌ها:
// <input type="hidden" name="csrf_token" value="<?= csrf_token() ?>">
```

تمرین ۸ middleware: شرطی بر اساس زمان

صورت سوال middleware ای بسازید که دسترسی به بخش خاصی را فقط در ساعات کاری (۹ صبح تا ۵ عصر) امکان‌پذیر کند.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class BusinessHours {
    private $startHour = 9;  // صبح 9
    private $endHour = 17;   // عصر 5
```

```
    public function handle() {
        $currentHour = (int)date('H');
        $currentDay = (int)date('N'); // 1=یکشنبه، ۷=دوشنبه

        // بررسی روزهای کاری (دوشنبه تا جمعه)
        if ($currentDay > 5) {
            $this->accessDenied('سرویس فقط در روزهای کاری فعال است');
        }

        // بررسی ساعات کاری
        if ($currentHour < $this->startHour || $currentHour >= $this->endHour) {
            $this->accessDenied('سرویس فقط از ساعت ۹ صبح تا ۵ عصر فعال است');
        }
    }

    private function accessDenied($message) {
        http_response_code(503); // Service Unavailable
        echo "دسترسی غیرمجاز: " . $message;
        exit;
    }
}

// استفاده:
$router->get('/business-panel', 'controller/business.php')->only('business_hours');
```

تمرین ۹ middleware: برای کش کردن صفحات

صورت سوال middleware **ای بسازید که پاسخ صفحات را برای مدت زمان مشخصی کش کند.**

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class CacheMiddleware {
    private $cacheTime = 300; // 5 دقیقه

    public function handle() {
        $cacheKey = $this->generateCacheKey();
        $cacheFile = __DIR__ . '/../../cache/' . $cacheKey . '.html';

        // اگر فایل کش وجود دارد و هنوز معتبر است
        if (file_exists($cacheFile) && (time() - filemtime($cacheFile)) < $this->cacheTime) {
            header('X-Cache: HIT');
            echo file_get_contents($cacheFile);
            exit;
        }

        // برای ذخیره خروجی output buffering شروع
        ob_start();
```

```
        // فراخوانی صفحه اصلی...
        // (اصلی پیاده‌سازی شود router این قسمت باید در)

        register_shutdown_function(function() use ($cacheFile) {
            $output = ob_get_contents();
            if ($output && !headers_sent()) {
                file_put_contents($cacheFile, $output);
                header('X-Cache: MISS');
            }
        });
    }

    private function generateCacheKey() {
        return md5($_SERVER['REQUEST_URI'] .
$_SERVER['QUERY_STRING']);
    }
}
```

تمرین ۱۰ middleware: برای بررسی وضعیت حساب کاربری

صورت سوال : یک middleware کامل بسازید که چندین بررسی روی حساب کاربری انجام دهد.

کد مورد انتظار:

```php
<?php
namespace core\middleware;

class AccountStatusChecker {
```

```php
public function handle() {
    if (!isset($_SESSION['user_id'])) {
        $this->redirectTo('/login', 'ابتدا وارد شوید');
    }

    $userId = $_SESSION['user_id'];

    // شبیه‌سازی دریافت اطلاعات کاربر از دیتابیس
    $user = $this->getUserFromDatabase($userId);

    if (!$user) {
        $this->redirectTo('/login', 'کاربر یافت نشد');
    }

    // بررسی فعال بودن حساب
    if (!$user['is_active']) {
        $this->redirectTo('/account-suspended', 'حساب شما غیرفعال است');
    }

    // بررسی تأیید ایمیل
    if (!$user['email_verified']) {
        $this->redirectTo('/verify-email', 'لطفاً ایمیل خود را تأیید کنید');
    }

    // بررسی انقضای حساب
    if ($user['expires_at'] && strtotime($user['expires_at']) < time()) {
        $this->redirectTo('/renew-account', 'حساب شما منقضی شده است');
```

```
        }

        // بررسی تغییر اجباری رمز عبور
        if ($user['must_change_password']) {
            $this->redirectTo('/change-password', 'باید رمز عبور خود را تغییر دهید');
        }

        // همه چیز OK است، ادامه بده
    }

    private function getUserFromDatabase($userId) {
        // دیتابیس query شبیه‌سازی
        return [
            'id' => $userId,
            'is_active' => true,
            'email_verified' => true,
            'expires_at' => null,
            'must_change_password' => false
        ];
    }

    private function redirectTo($url, $message) {
        $_SESSION['error_message'] = $message;
        header('location: ' . $url);
        exit;
    }
}
```

```
// در کلاس Middleware:
const MAP = [
    'guest' => Guest::class,
    'auth' => Auth::class,
    'account_check' => AccountStatusChecker::class
];

// استفاده:
$router->get('/dashboard', 'controller/dashboard.php')->only(['auth',
'account_check']);
```

نکات مهم برای تمرین‌ها:

۱. **تست کنید** :همه تمرین‌ها را با سناریوهای مختلف تست کنید
۲. **Error Handling**:همیشه حالت‌های خطا را در نظر بگیرید
۳. **Security**:در middleware های امنیتی دقت ویژه‌ای داشته باشید
۴. **Performance**: middlewareهایی که زیاد اجرا می‌شوند باید سریع باشند
۵. **Logging**:برای debug کردن، logگذاری مهم است

Preplexity

تمرین ۱ : ساخت Middleware مخصوص کاربران تاییدشده

صورت سؤال:

یک middleware جدید با نام EmailConfirmed بساز که فقط کاربرانی که ایمیل خود را تایید کرده‌اند بتوانند وارد صفحه پروفایل شوند.

پاسخ:

۱. ساخت کلاس:

```php
namespace core\middleware;

class EmailConfirmed {
    public function handle() {
        if (!($_SESSION['user']['email_confirmed'] ?? false)) {
            header('location: /laracast-php/public/verify-email');
            exit;
        }
    }
}
```

۲. افزودن به MAP در Middleware:

```php
public const MAP = [
    'guest' => Guest::class,
    'auth' => Auth::class,
    'confirmed' => EmailConfirmed::class
];
```

۳. استفاده در routes:

```php
$router->get('/profile', 'controller/profile/show.php')->only('confirmed');
```

تمرین ۲: نمایش پیغام خطا برای کلید middleware نامعتبر

صورت سؤال:

کدی بنویس که اگر کلید عبوری به middleware وجود نداشت، پیام خطا با نام کلید چاپ شود.

پاسخ:

```php
public static function resolve($key)
{
    if (! $key) {
        return;
    }
    $middleware = static::MAP[$key] ?? false;
    if (! $middleware) {
        throw new \Exception("Middleware not found for key: {$key}");
    }
    (new $middleware)->handle();
}
```

تمرین ۳: افزودن متد only به متد put وdelete

صورت سؤال:

کد متدهای put و delete را طوری بنویس که پس از ثبت route بتوان only را chain کرد.

پاسخ:

```php
public function put($uri, $controller) {
    return $this->add('PUT', $uri, $controller);
}

public function delete($uri, $controller) {
    return $this->add('DELETE', $uri, $controller);
}
```

استفاده:

```php
$router->put('/profile/update', 'controller/profile/update.php')->only('auth');
```

```
$router->delete('/note/remove', 'controller/note/remove.php')->only('auth');
```

تمرین ۴: جلوگیری از ورود کاربر به صفحه ثبت‌نام در صورت لاگین

صورت سؤال:

middlewareمناسب بنویس که اگر کاربر وارد شده بود، با مراجعه به صفحه ثبت‌نام به خانه برگردانده شود.

پاسخ:

```
php
namespace core\middleware;

class Guest {
    public function handle() {
        if ($_SESSION['user'] ?? false) {
            header('location: /laracast-php/public/');
            exit;
        }
    }
}
```

تمرین ۵: اجرای middleware فقط زمانی که مقدار آن ست شده

صورت سؤال:

در متد route از کلاسRouter ، طوری کد بزن که اگر مسیر middleware نداشته باشد، هیچ middleware اجرا نشود.

پاسخ:

```
php
public function route($uri, $method) {
    foreach ($this->routes as $route) {
```

```
        if ($route['uri'] == $uri && $route['method'] == strtoupper($method)) {
            if($route['middleware']) {
                Middleware::resolve($route['middleware']);
            }
            return require_once base_path($route['controller']);
        }
    }
    $this->abort();
}
```

Chatgpt

تمرین ۱:

یک متد در Router بساز به نام `only($key)` که اگر مقدار `'guest'` داده شد، کاربر لاگین کرده را به صفحه `/` ریدایرکت کند.

راهنمایی کد:

```
public function only($key)
{
    if ($key === 'guest' && $_SESSION['user'] ?? false)
{
        header("Location: /");
        exit();
    }
}
```

◆تمرین ۲:

برای متد `only('auth')` **کد بنویس که اگر کاربر لاگین نبود، او را به** `/login` **بفرستی.**

راهنمایی کد:

```
public function only($key)
{
    if ($key === 'auth' && !($_SESSION['user'] ??
false)) {
        header("Location: /login");
        exit();
    }
}
```

◆تمرین ۳:

یک Route **تعریف کن برای صفحه** `profile` **که فقط کاربر لاگین کرده بتواند ببیند.**

راهنمایی کد در فایل:routes

```
$router->get('/profile', 'controller/profile.php')-
>only('auth');
```

◆تمرین ۴:

یک Route **بساز برای** `register` **که فقط کاربران مهمان بتوانند آن را ببینند (کسی که لاگین کرده نباید بتواند وارد شود).**

راهنمایی کد در فایل:routes

```
$router->get('/register',
'controller/register/create.php')->only('guest');
```

◆تمرین ۵:

یک متد عمومی در Router اضافه کن که اگر مقدار role**در سشن** admin**بود، فقط در آن صورت کاربر بتواند وارد** /admin/dashboard **شود. در غیر این صورت او را بفرست به** /.

راهنمایی کد:

```
public function onlyAdmin()
{
    if (($_SESSION['role'] ?? '') !== 'admin') {
        header("Location: /");
        exit();
    }
}
```

و در:routes

```
$router->get('/admin/dashboard',
'controller/admin/dashboard.php')->onlyAdmin();
```

تمرین ۶: ساخت Middleware برای جلوگیری از دسترسی کاربران لاگین‌شده به صفحه ورود

✍سوال:
کدی بنویس که اگر کاربر وارد شده باشد (مثلاً $_SESSION['user'] **ست شده)، دیگر نتواند صفحه** /login **را ببیند و به صفحه** / **هدایت شود.**

✅پاسخ (ایده):

```
function guestOnly() {
    if ($_SESSION['user'] ?? false) {
        header("Location: /");
        exit;
    }
}
```

و در:router

```
$router->get('/login', 'controllers/login.php')-
>only('guest');
```

◆تمرین ۷ Middleware :برای کاربرانAdmin

✍سوال:

کدی بنویس که فقط اگر `$_SESSION['user']['role'] === 'admin'` **بود، اجازه دیدن صفحه** `/admin` **را بدهد.**

✅پاسخ (ایده):

```
function adminOnly() {
    if (($_SESSION['user']['role'] ?? '') !== 'admin')
{
        http_response_code(403);
        echo "Access Denied!";
        exit;
    }
}
```

◆تمرین ۸: ایجاد middleware عمومی

✍سوال:

کدی بنویس که بتوانی یک Middleware **عمومی تعریف کنی و آن را روی چند** Route **مختلف اعمال کنی.**

✅پاسخ (ایده):

```
$router->get('/dashboard',
'controllers/dashboard.php')->only('auth');
$router->get('/profile', 'controllers/profile.php')-
>only('auth');
```

و در:router.php

```
public function only($key) {
    if ($key === 'auth' && !($_SESSION['user'] ??
false)) {
        header("Location: /login");
        exit;
    }
}
```

◆تمرین ۹: ساخت Middleware چند شرطی

✍سوال:

می‌خواهیم صفحه `/settings` **فقط برای کاربر لاگین‌شده و با نقش admin نمایش داده شود.**

✅پاسخ (ایده):

```
function authAndAdmin() {
    if (!($_SESSION['user'] ?? false)) {
        header("Location: /login");
        exit;
    }
    if ($_SESSION['user']['role'] !== 'admin') {
        http_response_code(403);
        echo "Only admins can access settings.";
        exit;
    }
}
```

◆تمرین ۱۰ Middleware :برای جلوگیری از دسترسی مهمان‌ها به بخشAPI

✍سوال:

فرض کن یک API ساده `/api/users` **داری. فقط کاربر لاگین‌شده باید بتواند آن را ببیند** Middleware .**بنویس.**

✅پاسخ (ایده):

```
function apiAuth() {
    if (!($_SESSION['user'] ?? false)) {
        header("Content-Type: application/json");
        echo json_encode(["error" => "Unauthorized"]);
        http_response_code(401);
        exit;
    }
}
```