◆ ۳۰ سوال مفهومی با جواب

۱. **سوال :چرا متد** validate **در کلاس** LoginForm **به صورت** static **تعریف شد؟**
جواب :چون می‌خواهیم بدون ساختن شی، اعتبارسنجی را انجام دهیم و بتوانیم با LoginForm::validate(...) **آن را فراخوانی کنیم** $this **.در متد** static **وجود ندارد، بنابراین برای دسترسی به اجزای کلاس، یک شی از کلاس ساخته می‌شود.** ;($instance = new static($attributes) :

۲. **سوال ;** $instance = new static($attributes) **:چه کاری انجام می‌دهد؟**
جواب :یک شی از همان کلاس فعلی می‌سازد (LoginForm) **و مقدار** $attributes **را به سازنده می‌دهد. اگر کلاس فرزندی از** LoginForm **داشته باشیم، این شی از کلاس فرزند ساخته می‌شود.**

۳. **سوال :چرا بعد از** throw new ValidationException(...) **کدهای بعدی اجرا نمی‌شوند؟**
جواب :زیرا throw **جریان اجرای برنامه را متوقف می‌کند و دنبال نزدیک‌ترین** catch **می‌گردد.**

۴. **سوال :متد** failed() **چه کاری انجام می‌دهد؟**
جواب :تعداد خطاهای موجود در کلاس LoginForm **را می‌شمارد و اگر خطایی وجود داشته باشد، مقدار غیر صفر برمی‌گرداند.**

۵. **سوال** ValidationException **:چه ویژگی‌ای دارد که خطاها را مدیریت می‌کند؟**
جواب :دارای property error **است که شامل آرایه خطاهاست و می‌توان با متد** getError() **آن را گرفت.**

۶. **سوال :تفاوت** getMessage() **و** getError() **در** ValidationException **چیست؟**
جواب getMessage() **:پیام ثابت کلاس** Exception **را برمی‌گرداند، ولی** getError() **آرایه جزئیات خطاها را برمی‌گرداند.**

۷. **سوال :**چرا استفاده از try/catch در index.php بهتر از store.php است؟

جواب :چون می‌توان همه خطاهای پروژه را در یک نقطه مدیریت کرد و نیاز به تکرار try/catchدر همه کنترلرها نیست.

۸. **سوال :**اگر یک خطا پرتاب شود و try/catch مناسب وجود نداشته باشد، چه اتفاقی می‌افتد؟

جواب :برنامه متوقف می‌شود و PHP یک Fatal Error صادر می‌کند.

۹. **سوال :**چه تفاوتی بین مدیریت خطا با Exception و بدون Exception وجود دارد؟

جواب :با Exception → خطا پرتاب می‌شود و توسط catch مدیریت می‌شود. بدون Exception → خطا مستقیم در session ذخیره و کاربر ریدایرکت می‌شود.

۱۰. **سوال :**چرا آرایه `$attributes`در store.php ساخته می‌شود؟

جواب :برای ارسال اطلاعات فرم) مثل email و (password به متد validate تا اعتبارسنجی انجام شود.

۱۱. **سوال** `adderror` **:**چه کاری انجام می‌دهد؟

جواب :یک خطا برای یک فیلد مشخص ذخیره می‌کند.

۱۲. **سوال :**تغییر متد `adderror`برای زنجیره‌سازی چه مزیتی دارد؟

جواب :می‌توان متدها را پشت سر هم صدا زد، مثل `$form->addError(...)->throw();`.

۱۳. **سوال :**چرا `throw()`در LoginForm تعریف شد؟

جواب :تا شی LoginForm خودش بتواند خطاها را پرتاب کند و در catch مدیریت شود.

۱۴. **سوال** `$form = LoginForm::validate($attributes)` **:**چه چیزی برمی‌گرداند؟

جواب :یک شی LoginForm که شامل property های error و attributes است.

۱۵. **سوال :**چرا Validation باید به سطح بالاتر منتقل شود؟

جواب :برای جلوگیری از کد تکراری در کنترلرها و تمیز کردن کدها.

۱۶. **سوال :**ساختار کلاس Validator چه کاربردی دارد؟

جواب :شامل متدهای کمکی مثل `Is_Empty`و `Validate_email`و `Validate_password`برای اعتبارسنجی فیلدها است.

۱۷. **سوال :در متد**validate ، `$instance->failed()` **چه نقشی دارد؟**

جواب :بررسی می‌کند که آیا خطایی وجود دارد یا نه و در صورت وجود، Exception پرتاب می‌شود.

۱۸. **سوال :چرا از** `$_SERVER['HTTP_REFERER']` **استفاده می‌کنیم؟**

جواب :برای بازگشت کاربر به همان صفحه‌ای که فرم را پر کرده بود.

۱۹. **سوال :چه چیزی باعث می‌شود کد** store.php **کوتاه و تمیز شود؟**

جواب :حذف try/catch **از** store **و استفاده از** static validate **با** throw **در** LoginForm.

۲۰. **سوال :چرا در متد** validate **آرایه** `$attributes` **به** `$instance` **داده می‌شود؟**

جواب :تا متد سازنده کلاس LoginForm **بتواند اعتبارسنجی فیلدها را انجام دهد.**

۲۱. **سوال :تفاوت** `$this` **و** `$instance` **در متد** static **چیست؟**

جواب `$this` **:در متد** static **وجود ندارد،** `$instance` **شی ایجاد شده برای دسترسی به** property **ها و متدهاست.**

۲۲. **سوال :اگر کاربر ایمیل اشتباه وارد کند چه اتفاقی می‌افتد؟**

جواب :خطا توسط LoginForm **ذخیره شده و توسط** ValidationException **پرتاب می‌شود و در** catch **ذخیره می‌شود.**

۲۳. **سوال :تفاوت** `public $error` **و** `public readonly array $errors` **چیست؟**

جواب `readonly` **:فقط یکبار مقداردهی می‌شود و بعد تغییر نمی‌کند، اما** `public $error` **قابل تغییر است.**

۲۴. **سوال :چرا از** `new authenticator()` **استفاده می‌کنیم؟**

جواب :برای احراز هویت کاربران login) **یا** (register **و بررسی صحت** email **و** password.

۲۵. **سوال :اگر** `attempt()` false **باشد، چه کاری انجام می‌دهیم؟**

جواب :خطا در session **ذخیره و کاربر به صفحه** login **ریدایرکت می‌شود.**

۲۶. **سوال :مزیت** `static constructor` **در** ValidationException **چیست؟**

جواب :می‌توان یک instance **بسازیم، داده‌ها را اضافه کنیم و بلافاصله پرتاب کنیم.**

۲۷. **سوال :متد** `getErrors()` **چه کاری انجام می‌دهد؟**

جواب :آرایه خطاهای ذخیره شده در LoginForm یا ValidationException را برمی‌گرداند.

۲۸. **سوال :چه تفاوتی بین خطای اعتبارسنجی و خطای** login **وجود دارد؟**

جواب :اعتبارسنجی → مربوط به فیلدها و format آنهاست، login → مربوط به عدم تطابق با دیتابیس است.

۲۹. **سوال :چرا فایل** index.php **نقطه مرکزی مدیریت** Exception **است؟**

جواب :برای مدیریت جهانی خطاها و جلوگیری از تکرار try/catch در تمام کنترلرها.

۳۰. **سوال :چه مزیتی دارد که** ValidationException **شامل** `$attributes` **باشد؟**

جواب :داده‌های فرم قبلی را نگه می‌دارد تا در صفحه فرم دوباره نمایش داده شود.

◆ ۱۲ تمرین با جواب

تمرین ۱

سوال :یک آرایه `$attributes` **بسازید که شامل** email **و** password **از** $_POST **باشد.**

جواب:

```
$attributes = [
    'email' => $_POST['email'],
    'password' => $_POST['password']
];
```

تمرین ۲

سوال :اعتبارسنجی email **و** password **را با** LoginForm **انجام دهید.**

جواب:

```
$form = LoginForm::validate($attributes);
```

تمرین ۳

سوال :اگر اعتبارسنجی شکست خورد، خطاها را با Exception پرتاب کنید.
جواب:

```
$form->adderror('email', 'Invalid email or password')-
>throw();
```

تمرین ۴

سوال :بررسی کنید که login موفق بوده یا نه با.authenticator
جواب:

```
if ((new authenticator())-
>attempt($attributes['email'],
$attributes['password'])) {
    Redirect("/laracast-php/public");
}
```

تمرین ۵

سوال :اگر login موفق نبود، خطا ذخیره و ریدایرکت کنید.
جواب:

```
$form->adderror("email","کاربری با این ایمیل و پسوورد وجود ندارد")-
>throw();
```

تمرین ۶

سوال :در ValidationException متد ()getErrors را پیاده‌سازی کنید.
جواب:

```
public function getErrors() {
    return $this->errors;
                                                    }
```

تمرین ۷

سوال :در LoginForm متد ()failed را بنویسید که تعداد خطاها را بررسی کند.

جواب:

```
public function failed(){
    return count($this->error);
}
```

تمرین ۸

سوال :در index.php خطاهای ValidationException را catch کنید و به session منتقل کنید.

جواب:

```
catch (\core\ValidationException $exception){
    session::flash('error',$exception->error);
    session::flash('old', ['email' => $exception-
>attributes['email']]);
    return Redirect($_SERVER['HTTP_REFERER']);
}
```

تمرین ۹

سوال :در LoginForm متد ()throw را برای پرتاب Exception بنویسید.

جواب:

```
public function throw() {
    throw new ValidationException($this->error, $this-
>attributes);
}
```

تمرین ۱۰

سوال :تغییر متد adderror برای زنجیره‌سازی متدها.

جواب:

```
public function adderror($field, $message){
    $this->error[$field] = $message;
```

```
    return $this;
}
```

تمرین ۱۱

سوال :بررسی کنید که اگر فیلد email خالی باشد، خطا ثبت شود.

جواب:

```
if(Validator::Is_Empty($attributes['email'])){
    $form->adderror('email', 'Email is required');
}
```

تمرین ۱۲

سوال :در store.php بدون try/catch اعتبارسنجی انجام دهید و خطاها را پرتاب کنید.

جواب:

```
$form = LoginForm::validate([
    'email' => $_POST['email'],
    'password' => $_POST['password']
]);
if (!(new authenticator())-
>attempt($attributes['email'],
$attributes['password'])){
    $form->adderror("email","کاربری با این ایمیل و پسوورد وجود
ندارد")->throw();
```

X

```
}
```