Real World

Real World Examples

Anonymous · 09/09/2026

Real World Examples

Practical, framework-agnostic examples of using `derafu/translation`.

API Error Responses

```
use Derafu\Translation\Contract\TranslatableInterface;
use Symfony\Contracts\Translation\TranslatorInterface;
use Throwable;

final class ApiErrorHandler
{
    public function __construct(
        private readonly TranslatorInterface $translator,
    ) {
    }

    /**
     * @return array{error: array{message: string, code: int}}
     */
    public function handle(Throwable $e, ?string $locale = null): array
    {
        $message = $e instanceof TranslatableInterface
            ? $e->trans($this->translator, $locale)
            : $e->getMessage();

        return [
            'error' => [
                'message' => $message,
                'code' => $e->getCode(),
            ],
        ];
    }
}
```

Field-by-Field Validation Errors

A common pattern: collect one translatable exception per invalid field, then translate them all at once when building the response. Remember to register a `errors+intl-icu.*.*` resource (`ValidationException` uses the `errors` domain by default) — without one, `trans()` falls back to `symfony/translation`'s plain `%name%`-style substitution instead of ICU, which does not understand `{value}`-style placeholders. See ICU Formatting for the full explanation.

```
use Derafu\Translation\Exception\Core\TranslatableException;
use Symfony\Contracts\Translation\TranslatorInterface;

final class ValidationException extends TranslatableException
{
}

final class Validator
{
    /**
     * @return array<string, ValidationException> Keyed by field name.
     */
    public function validate(array $data): array
    {
        $errors = [];

        if (empty($data['email'])) {
            $errors['email'] = new ValidationException([
                'The field {field} is required.',
                'field' => 'email',
            ]);
        } elseif (!filter_var($data['email'], FILTER_VALIDATE_EMAIL)) {
            $errors['email'] = new ValidationException([
                'The value {value} is not a valid email.',
                'value' => $data['email'],
            ]);
        }

        return $errors;
    }
}

function formatErrors(array $errors, TranslatorInterface $translator, string $locale):
array
{
    $formatted = [];
    foreach ($errors as $field => $exception) {
        $formatted[$field] = $exception->trans($translator, $locale);
    }
    return $formatted;
}
```

```
// translations/errors+intl-icu.es.php
return [
    'The field {field} is required.' => 'El campo {field} es obligatorio.',
    'The value {value} is not a valid email.' => 'El valor {value} no es un correo
electrónico válido.',
];
```

Domain-Specific Exception Hierarchies

Group related exceptions under a shared base class with its own translation domain (see Exceptions for why `$defaultDomain` must be overridden this way):

```
use Derafu\Translation\Exception\Logic\TranslatableDomainException;

abstract class OrderException extends TranslatableDomainException
{
    protected string $defaultDomain = 'orders';
}

final class InsufficientStockException extends OrderException
{
    public static function forProduct(string $product, int $requested, int
$available): self
    {
        return new self([
            'Not enough stock for "{product}": requested {requested}, available
{available}.',
            'product' => $product,
            'requested' => $requested,
            'available' => $available,
        ]);
    }
}

final class InvalidStatusTransitionException extends OrderException
{
    public static function fromTo(string $from, string $to): self
    {
        return new self([
            'Cannot transition order from "{from}" to "{to}".',
            'from' => $from,
            'to' => $to,
        ]);
    }
}
```

```
// translations/orders+intl-icu.es.php
return [
    'Not enough stock for "{product}": requested {requested}, available {available}.'
=>
        'No hay stock suficiente de "{product}": se pidieron {requested}, hay
{available} disponibles.',
    'Cannot transition order from "{from}" to "{to}".' =>
        'No se puede pasar el pedido de "{from}" a "{to}".',
];
```

CLI Tools

Translatable exceptions work just as well outside of HTTP contexts — useful for CLI tools that need to report errors in the operator's language:

```
use Derafu\Translation\TranslatorFactory;
use Derafu\Translation\TranslationResourceRegistrar;

$locale = getenv('APP_LOCALE') ?: 'en';

$translator = TranslatorFactory::create($locale, ['en']);
(new TranslationResourceRegistrar($translator))->registerDirectory(__DIR__ .
'/translations');

try {
    runCommand();
} catch (Throwable $e) {
    $message = $e instanceof \Derafu\Translation\Contract\TranslatableInterface
        ? $e->trans($translator)
        : $e->getMessage();

    fwrite(STDERR, $message . PHP_EOL);
    exit(1);
}
```

Last updated on 09/09/2026