Testing

Testing Guide

Anonymous · 09/09/2026

Testing Guide

How to test code that uses `derafu/translation` — translatable exceptions, ICU formatting, and resource registration.

Testing Untranslated Behavior

Since `getMessage()` is fully formatted at construction time, most tests don't need a translator at all:

```
use PHPUnit\Framework\TestCase;

final class ValidationExceptionTest extends TestCase
{
    public function testPlainStringMessage(): void
    {
        $exception = new ValidationException('Email is invalid.');

        $this->assertSame('Email is invalid.', $exception->getMessage());
    }

    public function testIcuParametersAreSubstitutedEagerly(): void
    {
        $exception = new ValidationException([
            'The value {value} is not a valid email.',
            'value' => 'test@example',
        ]);

        $this->assertSame(
            'The value test@example is not a valid email.',
            $exception->getMessage(),
        );
    }
}
```

Testing with a Stub Translator

For unit tests, a `TranslatorInterface` stub configured with `willReturnCallback()` is usually enough — it doesn't need to verify how many times it was called, just what it returns:

```
use PHPUnit\Framework\TestCase;
use Symfony\Contracts\Translation\TranslatorInterface;

final class OrderExceptionTest extends TestCase
{
    public function testTranslation(): void
    {
        $translator = $this->createStub(TranslatorInterface::class);
        $translator->method('trans')->willReturnCallback(
            fn (string $id, array $parameters = []): string => match ($id) {
                'Not enough stock for "{product}".' => sprintf(
                    'No hay stock de "%s".',
                    $parameters['product'],
                ),
                default => $id,
            },
        );

        $exception = OrderException::insufficientStock('Widget');

        $this->assertSame(
            'No hay stock de "Widget".',
            $exception->trans($translator, 'es'),
        );
    }
}
```

Use `createMock()` instead of `createStub()` only when you actually need to assert on the number of calls (e.g. `expects($this->once())`) — PHPUnit flags `createMock()` used purely for return-value stubbing as unnecessary.

Testing with a Real Translator

For integration tests, use the real `TranslatorFactory` + `TranslationResourceRegistrar` against fixture translation files — verifying the actual behavior end to end, ICU formatting included.

Fixtures for the examples below (`tests/fixtures/translations/`):

```
// errors+intl-icu.es.php
return [
    'The value {value} is not a valid email.' => 'El valor {value} no es un correo
```

```
electrónico válido.',
];
```

```
// messages+intl-icu.en.php
return [
    'welcome' => 'Welcome!',
];
```

And an extra directory used only by the precedence test below, `tests/fixtures/translations-override/`:

```
// messages+intl-icu.en.php
return [
    'welcome' => 'Overridden message.',
];
```

```
use PHPUnit\Framework\TestCase;
use Derafu\Translation\TranslationResourceRegistrar;
use Derafu\Translation\TranslatorFactory;
use Symfony\Component\Translation\Translator;

final class TranslationIntegrationTest extends TestCase
{
    private Translator $translator;

    protected function setUp(): void
    {
        $this->translator = TranslatorFactory::create('en', ['en', 'es']);

        $registrar = new TranslationResourceRegistrar($this->translator);
        $registrar->registerDirectory(__DIR__ . '/../fixtures/translations');
    }

    public function testValidationErrorIsTranslated(): void
    {
        $exception = new ValidationException([
            'The value {value} is not a valid email.',
            'value' => 'test@example',
        ]);

        $this->assertSame(
            'El valor test@example no es un correo electrónico válido.',
            $exception->trans($this->translator, 'es'),
        );
    }
}
```

Testing Resource Registration

Test `TranslationResourceRegistrar` directly against a fixtures directory to verify discovery and precedence, without needing any specific exception class:

```
use PHPUnit\Framework\TestCase;
use Derafu\Translation\TranslationResourceRegistrar;
use Derafu\Translation\TranslatorFactory;

final class TranslationResourceRegistrarTest extends TestCase
{
    public function testDiscoversDomainsAndLocales(): void
    {
        $translator = TranslatorFactory::create('en');
        $registrar = new TranslationResourceRegistrar($translator);

        $registrar->registerDirectory(__DIR__ . '/../fixtures/translations');

        sort($domains = $registrar->getRegisteredDomains());
        sort($locales = $registrar->getRegisteredLocales());

        $this->assertSame(['errors+intl-icu', 'messages+intl-icu'], $domains);
        $this->assertSame(['en', 'es'], $locales);
    }

    public function testLaterDirectoriesOverridePreviousOnesForTheSameKey(): void
    {
        $translator = TranslatorFactory::create('en');
        $registrar = new TranslationResourceRegistrar($translator);

        $registrar->registerDirectories([
            __DIR__ . '/../fixtures/translations',
            __DIR__ . '/../fixtures/translations-override',
        ]);

        $this->assertSame(
            'Overridden message.',
            $translator->trans('welcome', [], 'messages+intl-icu'),
        );
    }
}
```

Testing Custom Resource Providers

```
use PHPUnit\Framework\TestCase;
use Derafu\Translation\Contract\TranslationResourceProviderInterface;
```

```
final class PackageTranslationResourceProviderTest extends TestCase
{
    public function testReturnsExpectedDirectories(): void
    {
        $provider = new PackageTranslationResourceProvider();

        $directories = iterator_to_array($provider->getDirectories());

        $this->assertNotEmpty($directories);
        foreach ($directories as $directory) {
            $this->assertDirectoryExists($directory);
        }
    }
}
```

Testing ICU Edge Cases

```
use PHPUnit\Framework\TestCase;
use Derafu\Translation\TranslatableMessage;

final class IcuFormattingTest extends TestCase
{
    public function testMissingOtherCategoryFallsBackToRawMessage(): void
    {
        $message = new TranslatableMessage(
            '{gender, select, male{He} female{She}}',
            ['gender' => 'unknown'],
        );

        $this->assertSame(
            '{gender, select, male{He} female{She}}',
            (string) $message,
        );
    }

    public function testUnmatchedBraceFallsBackToRawMessage(): void
    {
        $message = new TranslatableMessage('Hello {name', ['name' => 'John']);

        $this->assertSame('Hello {name', (string) $message);
    }

    public function testMissingParameterLeavesPlaceholderUnexpanded(): void
    {
        $message = new TranslatableMessage('{count} items');

        $this->assertSame('{count} items', (string) $message);
    }
}
```

Remember:

- Test both the untranslated (`getMessage()`) and translated (`trans()`) paths.
- Prefer `createStub()` over `createMock()` for a `TranslatorInterface` double unless you're actually asserting call counts.
- Use real fixture files and a real `Translator` for integration-level coverage of ICU formatting and resource discovery — it's fast and catches naming-convention mistakes that a stub translator can't.
- Always test with the `other` case missing and with missing parameters: both fail *silently*, falling back to the raw message, not by throwing.

Last updated on 09/09/2026