Welcome to knowledge-base

Pest

Kommandos

pest: composer test:pest-coverage

Siehe hier: pest.de

pest: Datei Testen

```
composer test:pest-coverage Tests\Pfad\Zur\Datei\DateiOhneEndung
```

php

Try catch finally

Beim Ausführen unserer Skripte treten Fehler und Ausnahmen auf, das gehört zur Natur der Sache. Jedoch ist es nötig, Fehler voneinander zu unterscheiden und entsprechend zu behandeln. Dafür können wir in PHP die Schlüsselwörter **try**, **catch** und **finally** verwenden.

1. Ein einfacher PHP try catch Block

Fehler und Ausnahmen können wir behandeln, wenn der Code, der diese produziert, in einem **try** Block geschrieben wird. Mit dem Schlüsselwort **catch** können wir diese dann abfangen.

```
<?php
try {
   # Tu was
} catch (Exception $e) {
   # Kümmere dich um Ausnahmen
}
```

1.1 Fehler und Ausnahmen

Ein Fehler tritt zum Beispiel auf, wenn du eine Funktion aufrufst, die nie definiert wurde.

```
<?php undefinierte_funktion(); ?>
```

Output:

```
Fatal error: Uncaught Error: Call to undefined function undefinierte\_funktion() in C:\\xampp\\htdocs\\codecitrus\
\try\_catch.php:1 Stack trace: #0 {main} thrown in C:\\xampp\\htdocs\\codecitrus\\try\_catch.php on line 1
```

Um Code zu definieren, der nur ausgeführt wird, wenn ein Fehler (**Error**) auftritt, kannst du **try** und **catch** anwenden.

```
<?php
try {
   undefinierte_funktion();
   echo 'Es ist kein Fehler aufgetreten';
} catch (Error $e) {
   echo 'Ein Fehler ist aufgetreten' . '<br>';
   echo $e->getMessage();
}
```

Output:

```
Ein Fehler ist aufgetreten
Call to undefined function undefinierte_funktion()
```

Wenn stattdessen eine Ausnahme (**Exception**) auftritt, musst du das im **catch** Block so angeben, um diese zu behandeln.

```php
<?php
function ausnahme() {
    throw new Exception('Eine Ausnahme ist aufgetreten');
}
try {
    ausnahme();
} catch (Exception $e) {
    echo $e->getMessage();
}
```

Output:

```
Eine Ausnahme ist aufgetreten
```

1.2 Beide abfangen mit Throwable

Die vorigen Erklärungen haben sowohl Exceptions als auch Errors behandelt. Um beide abzufangen, könntest du zwei **catch** Blöcke hintereinander definieren.

```php
<?php
try {
    undefinierte_funktion();
} catch (Exception $e) {
    echo 'Exception';
} catch (Error $e) {
    echo 'Error';
}
```

Output:

```
Error
```

Eine weitere Möglichkeit ist, **Throwable** mit **catch** abzufangen. Throwable ist das Interface, das jede PHP-Klasse implementieren muss, soll sie mit **throw** aufgerufen werden. Für das nächste Beispiel wird die Funktion *ausnahme*, die wir vorhin definiert haben, erneut genutzt.

```php
<?php
try {
    undefinierte_funktion();
} catch (Throwable $t) {
    echo 'Gotta catch \'em all!';
}
// Gotta catch 'em all!
```

```
try {
    ausnahme();
} catch (Throwable $t) {
    echo 'Schnapp sie dir alle!';
}
// Schnapp sie dir alle!
```

2. finally um den Block zu beenden

Mit dem Schlüsselwort **finally** kannst du deinem Block weiteren Code hinzufügen. **finally** ermöglicht es dir, Code zu schreiben, der ausgeführt wird egal ob ein Fehler auftritt oder nicht.

```
<?php
function finally_beispiel_1() {
    try {
        throw new Exception();
    } catch (Exception $e) {
        echo "Exception!" . '<br>';
    } finally {
        echo "Finally!";
    }
}
finally_beispiel_1();
```

Output:

```
Exception!
Finally!
```

2.1 Vorsicht mit return!

return kann bei Verwendung von **finally** zu unerwarteten Ergebnissen führen.

```
<?php
function finally_beispiel_2() {
    try {
        throw new Exception();
        return 1;
    } catch (Exception $e) {
        echo "Exception!" . '<br>';
        return 2;
    } finally {
        echo "Finally!" . '<br>';
        return 3;
    }
}
echo finally_beispiel_2();
```

Output:

```
Exception!
Finally!
3
```

Im vorigen Beispiel wurde der Wert von **return** aus dem **catch** Block einfach mit dem aus **finally** überschrieben. Dasselbe würde passieren, wenn **return** bereits einen Wert aus **try** empfangen hätte.

3. Fazit – PHP try & catch

Mit **try**, **catch** und **finally** stellt uns PHP mächtige Werkzeuge zur Verfügung, um Ausnahmefälle und Fehler zu behandeln. Damit können wir Logs erstellen, alternative Routinen ausführen oder einfach unser Skript stoppen. Vorsicht ist jedoch geboten, wenn wir unseren Code in solchen Blöcken mit **return** verwenden, da dies zu unerwarteten Ergebnissen führen kann.

- Original Artikel von Patrick

When does PHP call __destruct()?

In PHP,

`__construct()` is called while creating an object and `__destruct()` is called while the object is being removed from memory. Using this knowledge, we can create more fluent APIs as demonstrated by Freek Van der Herten in this short video.

Now let's see when PHP calls

`__destruct()` exactly.

An object is removed from memory if you explicitly remove it:

```php
<?php
$object = new Object();

unset($object); // __destruct will be called immediately.

$object = null; // __destruct will be called immediately.
```

It's also called when the scope where the object live is about to be terminated, for example at the end of a controller method:

```php
<?php
function store(Request $request)
{
  $object = new Object();

  User::create(...);
```

```
  // __destruct will be called here.

  return view('welcome');
}
```

Even if we're within a long running process, queued job for example, __destruct will be called before the end of the handle method:

```
<?php
function handle()
{
  $object = new Object();

  User::create(...);

  // __destruct will be called here.
}
```

It'll also be called when the script is being terminated:

```
<?php
function handle()
{
  $object = new Object();

  User::create(...);

  // __destruct will be called here.

  exit(1);
}
```

- Original article by Mohamed Said

phpstan

PHPStan ist ein Tool für statische Code-Analyse in PHP. Es überprüft den Code auf Typfehler, unbeabsichtigte Seiteneffekte und andere potenzielle Fehlerquellen. PHPStan bietet Entwicklern eine höhere Code-Qualität und hilft dabei, potenzielle Fehler im Code zu finden, bevor sie in der Produktion auftreten. Das Tool nutzt dabei eine Kombination aus Typ-Inferenz und statischer Analyse, um sicherzustellen, dass der Code typsicher ist und bestimmten Coding-Standards entspricht. Es ist auch eine Erweiterung von PHP und kann in der Entwicklungsumgebung oder als Teil von Continuous Integration/Continuous Deployment (CI/CD) Pipelines verwendet werden, um sicherzustellen, dass der Code in jeder Phase der Entwicklung korrekt funktioniert.

Kommandos

phpstan: Der tägliche Wahnsinn

```
vendor/bin/phpstan clear-result-cache --memory-limit=5G
vendor/bin/phpstan analyse --ansi --memory-limit=5G
vendor/bin/phpstan --generate-baseline --memory-limit=5G
```

Shortcuts

Visual Studio Code

Shortcut	Beschreibung
Cmd (⌘) + Shift (⇑) + c	Öffnet Consolen-IDE Iterm und springt direkt zum Projekt
Cmd (⌘) + Shift (⇑) + e	Öffnet den Workspace-Explorer
Cmd (⌘) + Rechts	Spring zum Ende der Zeile
Cmd (⌘) + Links	Spring zum Anfang der Zeile
Cmd (⌘) + Oben	Springt zur ersten Zeile
Cmd (⌘) + Unten	Springt zur letzten Zeile
Cmd (⌘) + Enter	Zeile nach unten einfügen
Cmd (⌘) + L	Komplette Zeile markieren
Cmd (⌘) + D	Wort auswählen - Wiederholen Sie die Auswahl anderer Vorkommen
Cmd (⌘) + F	Wort markieren und es wird im Suchfenster angezeigt, Suchfenster wird selektiert
Cmd (⌘) + E	Wort markieren und es wird im Suchfenster angezeigt, Markierung bleibt auf das Wor
Cmd (⌘) + A	Alles selektieren
Cmd (⌘) + C	Markierung kopieren
Cmd (⌘) + X	Markierung ausschneiden
Cmd (⌘) + V	Kopierten Zwischenspeicher einfügen

Shortcut	Beschreibung
Shift (⇑) + ESC	Suchfenster schließen
Option (⌥) + Oben	Zeile nach oben verschieben
Option (⌥) + Unten	Zeile nach unten verschieben
Option (⌥) + Shift (⇑) + oben	Zeile nach oben kopieren
Option (⌥) + Shift (⇑) + unten	Zeile nach unten kopieren
Cmd (⌘) + Option (⌥) + Shift (⇑) + Oben	Zeile nach oben markieren und cursor platzieren
Cmd (⌘) + Option (⌥) + Shift (⇑) + rechts	Zeile nach rechts markieren und cursor platzieren
Cmd (⌘) + Option (⌥) + Shift (⇑) + unten	Zeile nach unten markieren und cursor platzieren
Cmd (⌘) + Option (⌥) + Shift (⇑) + links	Zeile nach links markieren und cursor platzieren
Option (⌥) + Rechts	Cursor zum Wort-Ende
Option (⌥) + Shift (⇑) + Rechts	Markiert das nächste Wort nach rechts
Option (⌥) + Links	Cursor zum Wort+ Anfang
Option (⌥) + Shift (⇑) + Rechts	Markiert das nächste Wort nach links
Ctrl (^) + A	Zeilen Anfang
Ctrl (^) + E	Zeilen Ende
Ctrl (^) + Shift (⇑) + A	Vom Cursor alles bis zum Anfang markieren
Ctrl (^) + Shift (⇑) + E	Vom Cursor alles bis zum Ende markieren

Shortcut	Beschreibung
Ctrl (^) + Tab	Zwischen den offenen Dateien wechseln
Ctrl (^) + D	Löschen nach Rechts
Ctrl (^) + H	Löschen nach Links
Tab	Markierten Text ein Tab Schritt nach rechts verschieben
Cmd (⌘) + Tab	Markierten Text ein Tab Schritt nach lins verschieben
Option (⌥) + H	Github History anzeigen
Option (⌥) + Enter	Öffnet Workbench View
Cmd (⌘) + DEL	Vom Curosr alles nach links löschen
Cmd (⌘) + Backspace	Vom Curosr alles nach rechts löschen
Cmd (⌘) + Ctrl (^) + DEL	Vom Curosr das nächste WortPaar nach links löschen
Cmd (⌘) + Ctrl (^) + Backspace	Vom Curosr das nächste WortPaar nach rechts löschen
Cmd (⌘) + K + C	Zeile auskommentieren
Cmd (⌘) + K + U	Zeile einkommentieren
Cmd (⌘) + Shift (⇑) + A	Toggle Block-Kommentare / ein- und auskommentieren
Cmd (⌘)o + Shift (⇑) + /	Toggle Line-Kommentare / ein- und auskommentieren
Cmd (⌘) + Shift (⇑) + K	Zeile löschen
Cmd (⌘) + Option (⌥) + Ctrl (^) + /	Gehe zum Klammerende

Mac

Ausschneiden, Kopieren, Einsetzen und andere häufig verwendete Kurzbefehle

Shortcut	Beschreibung
Cmd (⌘) + X	Ausgewähltes Objekt aus dem Originaltext löschen und in die Zwischenablage kopieren.
Cmd (⌘) + C	Ausgewähltes Objekt in die Zwischenablage kopieren. Dies funktioniert auch mit Dateien im Finder.
Cmd (⌘) + V	Den Inhalt der Zwischenablage in das aktuelle Dokument oder die aktuelle App einfügen. Dies funktioniert auch mit Dateien im Finder.
Cmd (⌘) + Z	Den vorherigen Befehl widerrufen. Du kannst anschließend zum Wiederholen den Tastaturkurzbefehl Shift (⇧) + Cmd (⌘) +Z drücken, um den Widerrufen+ Befehl umzukehren. In manchen Apps kannst du mehrere Befehle widerrufen und wiederholen.
Cmd (⌘) + A	Alle Objekte auswählen.
Cmd (⌘) + F	Elemente in einem Dokument suchen oder ein Suchfenster öffnen.
Cmd (⌘) + G	Das nächste Auftreten des gefundenen Objekts suchen (vorwärts suchen). Um das vorherige Auftreten zu suchen (rückwärts suchen), drücke Shift (⇧) + Cmd (⌘) +G.
Cmd (⌘) + H	Die Fenster der vordersten App ausblenden. Um nur die vorderste App anzuzeigen und alle anderen Apps auszublenden, drücke Option (⌥) + Cmd (⌘) + H.
Cmd (⌘) + M	Das vorderste Fenster im Dock ablegen. Um alle Fenster der vordersten App im Dock abzulegen, drücke Option (⌥) + Cmd (⌘) +M.
Cmd (⌘) + O	Das ausgewählte Objekt oder ein Dialogfenster zur Auswahl einer Datei öffnen.
Cmd (⌘) + P	Das aktuelle Dokument drucken.
Cmd (⌘) + S	Das aktuelle Dokument sichern.

Shortcut	Beschreibung
Cmd (⌘) + T	Einen neuen Tab öffnen.
Cmd (⌘) + W	Das vorderste Fenster schließen. Um alle Fenster der App zu schließen, drücke Option (⌥) + Cmd (⌘) + W.
Option (⌥) + Cmd (⌘) + Esc	Beenden einer App erzwingen.
Cmd (⌘) + Leertaste	Das Spotlight-Suchfeld ein- oder ausblenden. Um eine Spotlight-Suche aus einem Finder+ Fenster durchzuführen, drücke Cmd (⌘) +Option (⌥) + Leertaste. (Wenn du mehrere Eingabequellen verwendest, um in anderen Sprachen zu schreiben, ändern diese Kurzbefehle die Eingabequelle, anstatt Spotlight anzuzeigen. Hier erfährst du, wie du einen nicht eindeutigen Tastaturkurzbefehl änderst
Ctrl (^) + Cmd (⌘) + Leertaste	Die Zeichenübersicht einblenden, in der du wählen kannst.
Ctrl (^) + Cmd (⌘) + F	App im Vollbildmodus verwenden, falls die App dies unterstützt.
Leertaste	verwenden, um eine Vorschau des ausgewählten Objekts zu sehen.
Cmd (⌘) + Tab	Unter den geöffneten Apps zur zuletzt verwendeten wechseln.
Cmd (⌘) + Abwärtsakzent (`)	Zwischen den Fenstern der App wechseln, die du gerade verwendest. (Das Zeichen auf der zweiten Taste ist je nach Tastatur unterschiedlich. Die Taste befindet sich in der Regel über der Tab und links von der Zahl 1.)
Shift (⇧) + Cmd (⌘) + 5	In macOS Mojave ein Bildschirmfoto oder eine Bildschirmaufnahme erstellen. Verwende bei älteren macOS-Versionen Shift (⇧) + Cmd (⌘) +3 oder Shift (⇧) + Cmd (⌘) +4, um Bildschirmfotos aufzunehmen.
Shift (⇧) + Cmd (⌘) + N	Einen neuen Ordner im Finder anlegen.
Cmd (⌘) + Komma (,)	Einstellungen für die vorderste App öffnen.

Kurzbefehle für Ruhezustand, Abmelden und Ausschalten

Möglicherweise musst du einige dieser Kurzbefehle etwas länger gedrückt halten als andere. So kannst du vermeiden, sie versehentlich zu verwenden.

Shortcut	**Beschreibung**
Ein-/Ausschalter	Drücken, um deinen Mac einzuschalten oder den Ruhezustand zu beenden. Halte den Schalter 1,5 Sekunden lang gedrückt, um deinen Mac in den Ruhezustand zu versetzen.* Halte den Schalter länger gedrückt, um das Ausschalten des Mac zu erzwingen.
Option (⌥) + Cmd (⌘) + Ein-/ Ausschalter* oder Option (⌥) + Cmd (⌘) +Medienauswurftaste	Ruhezustand des Mac aktivieren.
Ctrl (^) + Shift (⇧) + Ein-/ Ausschalter* oder Ctrl (^) + Shift (⇧) + Medienauswurftaste	Displays in Ruhezustand versetzen.
Ctrl (^) + Ein-/ Ausschalter* oder Ctrl (^) + Medienauswurftaste	Ein Dialogfenster anzeigen, das dir die Wahlmöglichkeiten "Neustart", "Ruhezustand" oder Ausschalten" anbietet.
Ctrl (^) + Cmd (⌘) + Ein-/ Ausschalter	* Neustart des Mac erzwingen, ohne zum Sichern von geöffneten oder nicht gesicherten Dokumenten aufzufordern.
Ctrl (^) + Cmd (⌘) +Medienauswurftaste	Alle Apps beenden und den Mac anschließend neu starten. Falls geöffnete Dokumente nicht gesicherte Änderungen enthalten, wirst du gefragt, ob du sie sichern möchtest.
Ctrl (^) + Option (⌥) + Cmd (⌘) + Ein-/Ausschalter* oder Ctrl (^) + Option (⌥) + Cmd (⌘) +Medienauswurftaste	Alle Apps beenden und den Mac anschließend ausschalten. Falls geöffnete Dokumente nicht gesicherte Änderungen enthalten, wirst du gefragt, ob du sie sichern möchtest.
Ctrl (^) + Cmd (⌘) +Q	Den Bildschirm sofort sperren.
Shift (⇧) + Cmd (⌘) +Q	Von deinem macOS+ Benutzeraccount abmelden. Du wirst zur Bestätigung des Vorgangs aufgefordert. Um

Shortcut	Beschreibung
	dich sofort ohne Bestätigung abzumelden, drücke Option (⌥) + Shift (⇧) + Cmd (⌘) +Q.

* Gilt nicht für den Touch ID-Sensor.

Finder- und Systemkurzbefehle

Shortcut	Beschreibung
Cmd (⌘) + D	Ausgewählte Dateien duplizieren.
Cmd (⌘) + E	Ausgewähltes Laufwerk oder Volume auswerfen.
Cmd (⌘) + F	Eine Spotlight-Suche im Finder+ Fenster starten.
Cmd (⌘) + I	Fenster "Informationen" für eine markierte Datei anzeigen.
Cmd (⌘) + R	(1) Wenn im Finder ein Alias ausgewählt ist ursprüngliche Datei für ausgewählten Alias anzeigen. (2) In einigen Apps wie Kalender oder Safari die Seite aktualisieren oder neu laden. (3) Unter Softwareupdate nochmals nach Softwareupdates suchen.
Shift (⇧) + Cmd (⌘) + C	Fenster "Computer" öffnen.
Shift (⇧) + Cmd (⌘) + D	Ordner "Schreibtisch" öffnen.
Shift (⇧) + Cmd (⌘) + F	Fenster "Zuletzt benutzt" öffnen, das alle kürzlich angesehenen oder geänderten Dateien auflistet.
Shift (⇧) + Cmd (⌘) + G	Ein Fenster "Gehe zum Ordner" öffnen.
Shift (⇧) + Cmd (⌘) + H	Benutzerordner des aktuellen macOS+ Benutzeraccounts öffnen.
Shift (⇧) + Cmd (⌘) + I	iCloud Drive öffnen.

Shortcut	Beschreibung
Shift (⇧) + Cmd (⌘) + K	Fenster "Netzwerk" öffnen.
Option (⌥) + Cmd (⌘) + L	Ordner "Downloads" öffnen.
Shift (⇧) + Cmd (⌘) + N	Neuen Ordner erstellen.
Shift (⇧) + Cmd (⌘) + O	Ordner "Dokumente" öffnen.
Shift (⇧) + Cmd (⌘) + P	Vorschaufenster in Finder+ Fenstern ein- oder ausblenden.
Shift (⇧) + Cmd (⌘) + R	Fenster "AirDrop" öffnen.
Shift (⇧) + Cmd (⌘) + T	Tableiste in Finder+ Fenstern ein- oder ausblenden.
Ctrl (^) + Shift (⇧) + Cmd (⌘) + T	Das ausgewählte Finder+ Objekt dem Dock hinzufügen (OS X Mavericks oder neuer)
Shift (⇧) + Cmd (⌘) + U	Ordner "Dienstprogramme" öffnen.
Option (⌥) + Cmd (⌘) + D	Dock ein- oder ausblenden.
Ctrl (^) + Cmd (⌘) + T	Das ausgewählte Objekt der Seitenleiste hinzufügen (OS X Mavericks oder neuer).
Option (⌥) + Cmd (⌘) + P	Pfadleiste in Finder+ Fenstern ein- oder ausblenden.
Option (⌥) + Cmd (⌘) + S	Seitenleiste in Finder+ Fenstern ein- oder ausblenden.
Cmd (⌘) + Schrägstrich (/)	Statusleiste in Finder+ Fenstern ein- oder ausblenden.
Cmd (⌘) + J	DarstellungsOption (⌥)en einblenden.
Cmd (⌘) + K	Das Fenster "Mit Server verbinden" öffnen.

Shortcut	Beschreibung
Ctrl (^) + Cmd (⌘) + A	Erstelle einen Alias für das ausgewählte Element.
Cmd (⌘) + N	Ein neues Finder+ Fenster öffnen.
Option (⌥) + Cmd (⌘) + N	Neuen intelligenten Ordner erstellen.
Cmd (⌘) + T	Tableiste ein- oder ausblenden, wenn nur ein Tab im aktuellen Finder geöffnet ist.
Option (⌥) + Cmd (⌘) + T	Symbolleiste ein- oder ausblenden, wenn nur ein Tab im aktuellen Finder+ Fenster geöffnet ist.
Option (⌥) + Cmd (⌘) + V	Dateien in der Zwischenablage von ihrem ursprünglichen zum aktuellen Speicherort bewegen.
Cmd (⌘) + Y	Die Funktion "Übersicht" zum Anzeigen einer Vorschau der ausgewählten Dateien nutzen.
Option (⌥) + Cmd (⌘) + Y	In der Übersicht eine Diashow der ausgewählten Dateien anzeigen.
Cmd (⌘) + 1	Die Objekte im Finder+ Fenster als Symbole anzeigen.
Cmd (⌘) + 2	Die Objekte im Finder+ Fenster als Liste anzeigen.
Cmd (⌘) + 3	Die Objekte im Finder+ Fenster in Spalten anzeigen.
Cmd (⌘) + 4	Die Objekte im Finder+ Fenster in einer Galerie anzeigen.
Cmd (⌘) + linke eckige Klammer ([)	Zum vorherigen Ordner wechseln.
Cmd (⌘) + rechte eckige Klammer (])	Zum nächsten Ordner wechseln.
Cmd (⌘) + Aufwärtspfeil	Ordner mit dem aktuellen Ordner öffnen.

Shortcut	Beschreibung
Cmd (⌘) + Ctrl (^) + Aufwärtspfeil	Ordner mit dem aktuellen Ordner in einem neuen Fenster öffnen.
Cmd (⌘) + Abwärtspfeil	Ausgewähltes Objekt öffnen.
Rechtspfeil	Ausgewählten Ordner öffnen. Diese Funktion ist nur in der Listendarstellung verfügbar.
Linkspfeil	Ausgewählten Ordner schließen. Diese Funktion ist nur in der Listendarstellung verfügbar.
Cmd (⌘) + Rückschritttaste	Das markierte Objekt in den Papierkorb verschieben.
Shift (⇧) + Cmd (⌘) + Rückschritttaste	Den Papierkorb leeren.
Option (⌥) + Shift (⇧) + Cmd (⌘) + Rückschritttaste	Den Papierkorb ohne Bestätigungsdialog leeren.
Cmd (⌘) + Helligkeit verringern	Bildschirmsynchronisation ein- bzw. ausschalten, wenn der Mac mit mehr als einem Monitor verbunden ist.
Option (⌥) + Helligkeit erhöhen	Systemeinstellung "Monitore" öffnen. Dies funktioniert mit beiden Helligkeitstasten.
Ctrl (^) + Helligkeit erhöhen oder Ctrl (^) + Helligkeit verringern	Helligkeit des externen Displays ändern (falls vom Display unterstützt).
Option (⌥) + Shift (⇧) + Helligkeit erhöhen oder Option (⌥) + Shift (⇧) + Helligkeit verringern	Display-Helligkeit in kleineren Schritten ändern. Füge die Ctrl (^) + Taste hinzu, um die Anpassung am externen Display vorzunehmen, sofern das Display dies unterstützt.
Option (⌥) + Mission Control	Systemeinstellung "Mission Control" öffnen.
Cmd (⌘) +Mission Control	Den Schreibtisch anzeigen.
Ctrl (^) + Abwärtspfeil	Alle Fenster der vordersten App einblenden.

Shortcut	Beschreibung
Option (⌥) + Lauter	Systemeinstellung "Ton" öffnen. Dies funktioniert mit allen Lautstärketasten.
Option (⌥) + Shift (⇧) + Lauter oder Option (⌥) + Shift (⇧) + Leiser	Lautstärke in kleineren Schritten regeln.
Option (⌥) + Tastaturhelligkeit erhöhen	Systemeinstellung "Tastatur" öffnen. Dies funktioniert mit beiden Tastaturhelligkeitstasten.
Option (⌥) + Shift (⇧) + Tastaturhelligkeit erhöhen oder Option (⌥) + Shift (⇧) + Tastaturhelligkeit verringern	Tastaturhelligkeit in kleineren Schritten anpassen.
Option (⌥) + Doppelklick	Einen Ordner in einem neuen Fenster öffnen und das aktuelle Fenster schließen.
Cmd (⌘) + Doppelklick	Einen Ordner in einem neuen Tab oder Fenster öffnen.
Cmd (⌘) beim Ziehen auf ein anderes Volume	Gezogenes Objekt auf ein anderes Volume verschieben, anstatt es zu kopieren.
Option (⌥) beim Ziehen	Gezogenes Objekt kopieren. Der Mauszeiger ändert sich, während du das Objekt ziehst.
Option (⌥) + Cmd (⌘) beim Ziehen	Alias des gezogenen Objekts erstellen. Der Mauszeiger ändert sich, während du das Objekt ziehst.
Option (⌥) + Klick auf ein Erweiterungsdreieck	Alle Ordner im ausgewählten Ordner öffnen. Diese Funktion ist nur in der Listendarstellung verfügbar.
Cmd (⌘) + Klick auf Fenstertitel	Ordner anzeigen, die den aktuellen Ordner enthalten.
Klicke in der Finder-Menüleiste auf das Menü "Gehe zu", um Kurzbefehle zum Öffnen vieler häufig benutzter Ordner zu	

Shortcut	Beschreibung
sehen, z. B. "Programme", "Dokumente", "Downloads", "Dienstprogramme" und "iCloud Drive".	

Kurzbefehle für Dokumente

Das Verhalten dieser Kurzbefehle kann je nach verwendeter App variieren.

Shortcut	Beschreibung
Cmd (⌘) + B	Den ausgewählten Text fett formatieren bzw. Fettschrift ein- oder ausschalten.
Cmd (⌘) + I	Den ausgewählten Text kursiv formatieren bzw. Kursivschrift ein- oder ausschalten.
Cmd (⌘) + K	Einen Weblink hinzufügen.
Cmd (⌘) + U	Den ausgewählten Text unterstrichen formatieren bzw. Unterstreichung ein- oder ausschalten.
Cmd (⌘) + T	Das Fenster "Schriften" ein- oder ausblenden.
Cmd (⌘) + D	Den Ordner "Schreibtisch" in den Dialogfenstern "Öffnen" oder "Sichern" auswählen.
Ctrl (^) + Cmd (⌘) + D	Die Definition eines ausgewählten Worts ein- oder ausblenden.
Shift (⇧) + Cmd (⌘) + Doppelpunkt (	)	Fenster "Rechtschreibung und Grammatik" einblenden.

Shortcut	Beschreibung	
Cmd (⌘) +Semikolon (;)	Falsch geschriebene Wörter im Dokument suchen.	
Option (⌥) + Rückschritttaste	Das Wort links von der Einfügemarke löschen.	
Ctrl (^) + H	Das Zeichen links von der Einfügemarke löschen. Verwende alternativ die Rückschritttaste.	
Ctrl (^) + D	Das Zeichen rechts von der Einfügemarke löschen. Verwende alternativ Fn + Rückschritttaste.	
Fn + Rückschritttaste	Vorwärts löschen auf Tastaturen, die keine Entf-Taste haben. Verwende alternativ Ctrl (^) + D.	
Ctrl (^) + K	Text zwischen der Einfügemarke und dem Ende der aktuellen Zeile oder des Absatzes löschen.	
Fn + Aufwärtspfeil	Seite nach oben	Eine Seite nach oben blättern.
Fn + Abwärtspfeil	Seite nach unten	Eine Seite nach unten blättern.
Fn + Linkspfeil	Pos1	Zum Anfang eines Dokuments blättern.
Fn + Rechtspfeil	Ende	Zum Ende eines Dokuments blättern.
Cmd (⌘) + Aufwärtspfeil	Einfügemarke an den Anfang des Dokuments bewegen.	
Cmd (⌘) + Abwärtspfeil	Einfügemarke an das Ende des Dokuments bewegen.	
Cmd (⌘) + Linkspfeil	Einfügemarke an den Anfang der aktuellen Zeile bewegen.	

Shortcut	Beschreibung
Cmd (⌘) + Rechtspfeil	Einfügemarke an das Ende der aktuellen Zeile bewegen.
Option (⌥) + Linkspfeil	Einfügemarke an den Anfang des vorhergehenden Worts bewegen.
Option (⌥) + Rechtspfeil	Einfügemarke an das Ende des nächsten Worts bewegen.
Shift (⇧) + Cmd (⌘) + Aufwärtspfeil	Text zwischen der Einfügemarke und dem Anfang des Dokuments markieren.
Shift (⇧) + Cmd (⌘) + Abwärtspfeil	Text zwischen der Einfügemarke und dem Ende des Dokuments markieren.
Shift (⇧) + Cmd (⌘) + Linkspfeil	Text zwischen der Einfügemarke und dem Anfang der aktuellen Zeile markieren.
Shift (⇧) + Cmd (⌘) + Rechtspfeil	Text zwischen der Einfügemarke und dem Ende der aktuellen Zeile markieren.
Shift (⇧) + Aufwärtspfeil	Textauswahl auf das der aktuellen Position am nächsten liegende Zeichen in der darüberliegenden Zeile erweitern.
Shift (⇧) + Abwärtspfeil	Textauswahl auf das der aktuellen Position am nächsten liegende Zeichen in der darunterliegenden Zeile erweitern.
Shift (⇧) + Linkspfeil	Textauswahl um ein Zeichen nach links erweitern.
Shift (⇧) + Rechtspfeil	Textauswahl um ein Zeichen nach rechts erweitern.

Shortcut	Beschreibung
Option (⌥) + Shift (⇧) + Aufwärtspfeil	Textauswahl bis zum Anfang des aktuellen Absatzes erweitern, bei nochmaligem Drücken bis zum Anfang des nächsten Absatzes.
Option (⌥) + Shift (⇧) + Abwärtspfeil	Textauswahl bis zum Ende des aktuellen Absatzes erweitern, bei nochmaligem Drücken bis zum Ende des nächsten Absatzes.
Option (⌥) + Shift (⇧) + Linkspfeil	Textauswahl bis zum Anfang des aktuellen Worts erweitern, bei nochmaligem Drücken bis zum Anfang des nächsten Worts.
Option (⌥) + Shift (⇧) + Rechtspfeil	Textauswahl bis zum Ende des aktuellen Worts erweitern, bei nochmaligem Drücken bis zum Ende des nächsten Worts.
Ctrl (^) + A	Zum Anfang der Zeile oder des Absatzes bewegen.
Ctrl (^) + E	Zum Ende einer Zeile oder eines Absatzes bewegen.
Ctrl (^) + F	Ein Zeichen vor bewegen.
Ctrl (^) + B	Ein Zeichen zurück bewegen.
Ctrl (^) + L	Cursor oder Auswahl im sichtbaren Bereich zentrieren.
Ctrl (^) + P	Eine Zeile nach oben bewegen
Ctrl (^) + N	Eine Zeile nach unten bewegen.
Ctrl (^) + O	Neue Zeile hinter der Einfügemarke einfügen.

Shortcut	Beschreibung	
Ctrl (^) + T	Das Zeichen hinter der Einfügemarke mit dem Zeichen davor austauschen.	
Cmd (⌘) + Linke geschweifte Klammer ({)	Linksbündig.	
Cmd (⌘) + Rechte geschweifte Klammer (})	Rechtsbündig.	
Shift (⇧) + Cmd (⌘) +Senkrechtstrich (	)	Zentrieren.
Option (⌥) + Cmd (⌘) + F	Zum Suchfeld wechseln.	
Option (⌥) + Cmd (⌘) + T	Symbolleiste in der App ein- oder ausblenden.	
Option (⌥) + Cmd (⌘) +C	Stil kopieren	Formatierungseinstellungen des ausgewählten Objekts in die Zwischenablage kopieren.
Option (⌥) + Cmd (⌘) +V	Stil einsetzen	Die kopierten Formatierungseinstellungen auf das ausgewählte Objekt anwenden.
Option (⌥) + Shift (⇧) + Cmd (⌘) +V	Einsetzen und Stil anpassen	Den Stil des umgebenden Inhalts auf das darin eingesetzte Objekt anwenden.
Option (⌥) + Cmd (⌘) + I	Das Fenster "Informationen" ein- oder ausblenden.	
Shift (⇧) + Cmd (⌘) + P	Papierformat	Ein Fenster zur Auswahl von Dokumenteinstellungen anzeigen.

Shortcut	Beschreibung
Shift (⇧) + Cmd (⌘) + S	Dialogfenster "Sichern unter" anzeigen oder aktuelles Dokument duplizieren.
Shift (⇧) + Cmd (⌘) + (–)	Ausgewähltes Objekt verkleinern.
Shift (⇧) + Cmd (⌘) + (+)	Ausgewähltes Objekt vergrößern. Cmd (⌘) +Gleichheitszeichen (=) führt dieselbe Funktion aus.
Shift (⇧) + Cmd (⌘) + (?)	Hilfe-Menü öffnen.

Valet

Kommandos

Valet: Erstellt V-Hosts Links: htttp://www.meine-lokale-webseite.test und htttps://www.meine-lokale-webseite.test

```
valet link --secure meine-lokale-webseite
```

Valet: Löscht die V-Hosts Links: htttp://www.meine-lokale-webseite.test und htttps://www.meine-lokale-webseite.test

```
valet unlink --secure meine-lokale-webseite
```

Valet: Alle registrierten Valet-Links anzeigen

```
valet links
```

Valet: Listet alle verfügbaren Valet Kommandos auf

```
valet -h
valet --help
```

Plugin

Auto Close Tag

Better Comments

Better Pest

DotENV

Dummy Text Generator

EditorConfig for VS Code

Filament Snippets

Git Blame

Git History

GitHub Pull Requests and Issues

GitHub Theme

gitignore

Image preview

Laravel Artisan

Laravel Blade formatter

Laravel Blade Snippets

Laravel Create View

Laravel Extension Pack

Laravel goto view

Plugin
laravel intellisense
Laravel Pint Formatter
Laravel Snippets
laravel-goto-components
Livewire Language Support
Markdown All in One
Markdown Preview Mermaid Support
Mermaid Markdown Syntax Highlighting
Mermaid Preview
MSSQL Snippets
Output Colorizer
Pest Snippets
PHP Intelephense
PHP IntelliSense
Rainbow Brackets 2
Remote - SSH
Remote - SSH: Editing Configuration Files
SVG
TODO Highlight

Plugin

vscode-icons

Sort lines

Conventional Commits

Zusammenfassung

Conventional Commits ist eine Methode zur Vereinheitlichung von Commit-Messages in Git-Repositories, die es erleichtert, Änderungen in der Codebasis zu verfolgen und nachzuvollziehen. Dabei handelt es sich um eine Reihe von Regeln und Konventionen, die festlegen, wie Commit-Messages aufgebaut sein sollten, um eine einheitliche Struktur und eine klare Bedeutung zu gewährleisten.

Die Struktur von Conventional Commits besteht aus einem Präfix und einem Nachrichtentext. Das Präfix besteht aus einem Typ, der die Art der Änderung angibt, und optional einer Bereichsangabe, die den betroffenen Codebereich kennzeichnet. Der Nachrichtentext sollte eine kurze Zusammenfassung der Änderung enthalten und optional eine längere Beschreibung oder Erklärung der Änderung.

Durch die Verwendung von Conventional Commits können Entwickler schnell herausfinden, welche Art von Änderungen in einem Commit vorgenommen wurden und welche Codebereiche betroffen sind. Dies erleichtert es, die Änderungen zu überprüfen und nachzuvollziehen, was insbesondere bei größeren Projekten oder bei der Zusammenarbeit mit anderen Entwicklern von Vorteil ist.

Conventional Commits ist ein Open-Source-Projekt und kann von Entwicklern auf der ganzen Welt verwendet werden. Es gibt auch eine Reihe von Tools und Plugins, die Conventional Commits unterstützen und die Integration in die Entwicklungs-Toolchain erleichtern.

Konventionelles Format für Commits

Präfix	**Art der Änderung**
feat	Eine neue Funktion
fix	Eine Fehlerbehebung
docs	Nur Änderungen an der Dokumentation
style	Änderungen, die die Bedeutung des Codes nicht beeinflussen (Leerzeichen, Formatierung, fehlende Semikolons usw.)
refactor	Eine Codeänderung, die weder einen Fehler behebt noch eine Funktion hinzufügt
perf	Eine Code-Änderung, die die Leistung verbessert

Präfix	Art der Änderung
test	Hinzufügen fehlender Tests oder Korrigieren vorhandener Tests
build	Änderungen, die das Build-System oder externe Abhängigkeiten betreffen (Beispielbereiche: gulp, broccoli, npm)
ci	Änderungen an unseren CI-Konfigurationsdateien und -Skripten (Beispielbereiche: Travis, Circle, BrowserStack, SauceLabs)
chore	Andere Änderungen, die keine src- oder Testdateien verändern

Klammern im Präfix für Bereichsangaben

Im Allgemeinen werden Klammern verwendet, um den Bereich des Commits anzugeben, z. B. das Modul oder das Tool, das von der Änderung betroffen ist. Wenn du Klammern verwendest, solltest du den Bereich in Klammern setzen, gefolgt von einem Doppelpunkt und einer Leerstelle, bevor du den eigentlichen Commit-Beschreibungstext schreibst.

Bereichsangabe	Beschreibung
(area)	Der Bereich, der von der Änderung betroffen ist (z.B. (login), (registration)

Beispiel

```
feat(login): add remember me checkbox
```

Die Verwendung von Klammern im Präfix bei Conventional Commits ist optional und hängt von der spezifischen Implementierung oder Konvention ab, die in deinem Projekt oder deiner Organisation verwendet wird.

Wenn du dich jedoch für die Verwendung von Klammern im Präfix entscheidest, solltest du sicherstellen, dass dies in der Dokumentation eures Projekts klar angegeben ist, damit alle Entwickler, die am Projekt arbeiten, sich an die gleiche Konvention halten können. Wenn du dich entscheidest, keine Klammern im Präfix zu verwenden, ist es jedoch wichtig, dass du eine klare und konsistente Commit-Beschreibung schreibst, die den Zweck und den Umfang der Änderung angibt, damit andere Entwickler die Änderung leicht verstehen und nachvollziehen können so wie oben beschrieben.

Quellen und Tools

Offizielle Dokumentation

Conventional Commits

Commitizen

"Commitizen", bietet eine vereinfachte Möglichkeit, Git-Commit-Nachrichten zu schreiben, die den Konventionen von "Conventional Commits" entsprechen.

Die Commitizen-Webanwendung stellt eine grafische Benutzeroberfläche zur Verfügung, die Entwicklern dabei hilft, durch einen interaktiven Prozess eine standardisierte Commit-Nachricht zu erstellen. Dadurch kann sichergestellt werden, dass Commit-Nachrichten konsistent und lesbar sind, was dazu beitragen kann, die Zusammenarbeit in einem Team zu erleichtern.

Commitizen

Pull request splitting

Pull Request Splitting | Pull Request Partitioning

In der Softwareentwicklung gibt es den Fachbegriff "Pull Request Splitting" oder "Pull Request Partitioning", um einen Pull Request in mehrere kleinere Pull Requests aufzuteilen.

Der Prozess des Pull Request Splitting ist eine Methode, um die Code-Review und die Integration von Änderungen in eine Codebasis zu vereinfachen. Bei komplexen oder umfangreichen Änderungen kann es schwierig sein, einen großen Pull Request zu überprüfen und zu integrieren. Durch das Aufteilen des Pull Requests in kleinere Einheiten wird es einfacher, Änderungen zu überprüfen und in die Codebasis zu integrieren.

Ein Beispiel für Pull Request Splitting könnte sein, wenn ein Entwickler eine neue Funktion in eine bestehende Anwendung integrieren möchte. Anstatt alle Änderungen in einem großen Pull Request zu bündeln, könnte der Entwickler den Pull Request in mehrere kleinere Einheiten aufteilen, wie z.B. eine Einheit für die neue Datenbank-Tabelle, eine Einheit für die Backend-Logik und eine Einheit für die Frontend-Integration. Auf diese Weise können die Reviewer die Änderungen leichter nachvollziehen und gezielter Feedback geben.

Das Pull Request Splitting ist also eine Methode, um große und komplexe Änderungen in kleinere und leichter verständliche Einheiten aufzuteilen, um die Code-Review und Integration von Änderungen zu erleichtern.

Workflow pull request splitting

Beispiel-Workflow

Hier ein Beispiel-Workflow für das Erstellen und Verwalten von Blog-Einträgen mit Pull-Request-Splitting:

```
graph TD
  A((master-branch)) -- erstelle --> B((blog-db-migration))
  A -- erstelle --> C((blog-list-page))
  A -- erstelle --> D((blog-create-entry))
  A -- erstelle --> E((blog-edit-entry))
  B -- merge --> A
  C -- merge --> A
  D -- merge --> A
  E -- merge --> A
  B -- merge --> C
  C -- merge --> D
  D -- merge --> E
```

1. Erstelle einen neuen Branch vom Master-Branch mit einem aussagekräftigen Namen wie z.B. "blog-db-migration" für die Migrationen.
    a. Führe alle notwendigen Schritte durch, um die Migrationen zu implementieren und zu testen.
2. Erstelle einen weiteren Branch vom aktuellen Master-Branch mit einem Namen wie "blog-list-page" für die Auflistung der Blog-Einträge.
    a. Führe den vorherigen Branch "blog-db-migration" in diesen neuen Branch "blog-list-page" ein und führe alle notwendigen Schritte durch, um die Auflistung der Blog-Einträge zu implementieren und zu testen.
3. Erstelle einen weiteren Branch vom aktuellen Master-Branch mit einem Namen wie "blog-create-entry" für das Hinzufügen neuer Blog-Einträge.
    a. Führe den vorherigen Branch "blog-list-page" in diesen neuen Branch "blog-create-entry" ein und führe alle notwendigen Schritte durch, um das Hinzufügen neuer Blog-Einträge zu implementieren und zu testen.
4. Erstelle einen weiteren Branch vom aktuellen Master-Branch mit einem Namen wie "blog-edit-entry" für das Bearbeiten von bestehenden Blog-Einträgen.
    a. Führe den vorherigen Branch "blog-create-entry" in diesen neuen Branch "blog-edit-entry" ein und führe alle notwendigen Schritte durch, um das Bearbeiten von bestehenden Blog-Einträgen zu implementieren und zu testen.

Warning

Es ist wichtig, dass jeder Pull-Request unabhängig voneinander funktioniert und nur die notwendigen Änderungen enthält. Daher sollte jeder Pull-Request nur dann bewertet werden, wenn der vorherige Pull-Request bereits gemerged wurde und der zu betrachtende Pull-Request daraufhin aktualisiert wurde. Wenn dies nicht beachtet wird, können zu viele Dateien im Pull-Request zum Review angezeigt werden, was das Review erschweren kann. Es ist auch wichtig, in jedem Pull-Request darauf hinzuweisen, auf welchen anderen Pull-Requests er aufbaut, um sicherzustellen, dass sie in der richtigen Reihenfolge geprüft und gemerged werden.

Tip

Stashes können als Workaround genutzt werden, um Änderungen in verschiedenen Branches zu organisieren

Falls man während der Arbeit bemerkt, dass man im aktuellen Branch Änderungen vorgenommen hat, die aber eigentlich einem anderen Branch zugeordnet werden müssten, kann man das Stashen als Workaround nutzen. Man kann die betreffenden Dateien in einem Stash zwischenspeichern, um sie später im richtigen Branch zuzuordnen. Auf diese Weise kann man vermeiden, dass Änderungen versehentlich im falschen Branch landen und so den Arbeitsablauf durcheinander bringen. Nachdem man den Stash angelegt hat, kann man ihn später auf den entsprechenden Branch verteilen und den Stash löschen.

Tip

Es ist sinnvoll, den Code regelmäßig mit dem Haupt-Branch zu synchronisieren, um sicherzustellen, dass es zu keinen Konflikten kommt, wenn die Pull-Requests gemerged werden sollen.

Workflow pull request splittingen

Sample workflow

Here's a sample workflow for creating and managing blog posts with pull request splitting:

```
graph TD
  A((master-branch)) -- create --> B((blog-db-migration))
  A -- create --> C((blog-list-page))
  A -- create --> D((blog-create-entry))
  A -- create --> E((blog-edit-entry))
  B -- merge --> A
  C -- merge --> A
  D -- merge --> A
  E -- merge --> A
  B -- merge --> C
  C -- merge --> D
  D -- merge --> E
```

1. Create a new branch from the master branch with a meaningful name such as "blog-db-migration" for the migrations.
    a. Perform all necessary steps to implement and test the migrations.
2. Create another branch from the current master branch with a name like "blog-list-page" for listing the blog entries.
    a. Add the previous branch "blog-db-migration" to this new branch "blog-list-page" and perform all the necessary steps to implement and test the listing of blog entries.
3. Create another branch from the current master branch with a name like "blog-create-entry" for adding new blog entries.
    a. Merge the previous branch "blog-list-page" into this new branch "blog-create-entry" and perform all necessary steps to implement and test adding new blog entries.
4. Create another branch from the current master branch with a name like "blog-edit-entry" for editing existing blog entries.
    a. Merge the previous branch "blog-create-entry" into this new branch "blog-edit-entry" and perform all necessary steps to implement and test editing of existing blog entries.

Warning

It is important that each pull request functions independently and contains only the necessary changes. Therefore, each pull request should only be evaluated if the previous pull request has already been merged and the pull request under review has been updated as a result. Failure to do so can result in too many files being displayed in the pull request for review, which can make the review more difficult. It is also important to indicate in each pull request what other pull requests it builds on to ensure that they are reviewed and merged in the correct order.

Tip

Stashes can be used as a workaround to organize changes in different branches

If you notice while working that you have made changes in the current branch, but they should actually be assigned to another branch, you can use stashing as a workaround. You can save the files in question in a stash, so that you can assign them to the correct branch later. This way you can avoid that changes accidentally end up in the wrong branch and thus disrupt the workflow. After creating the stash, you can later distribute it to the appropriate branch and delete the stash.

Tip

It is useful to periodically synchronize the code with the main branch to ensure that there are no conflicts when the pull requests are to be memoized.

Boy-Scouting-Principle - Pfadfinderregel

Zusammenfassung

Das Boy-Scout-Softwareentwicklungsprinzip und die Pfadfinderregel in der Softwareentwicklung sind wichtige Grundsätze, die Entwickler anwenden können, um den Code zu verbessern und sicherzustellen, dass er immer auf dem neuesten Stand bleibt.

Das Boy-Scout-Prinzip fordert Entwickler auf, den Code bei jeder Gelegenheit zu verbessern, indem sie ihn "aufräumen", um sicherzustellen, dass er lesbar, verständlich und wartbar bleibt. Dies kann bedeuten, dass Entwickler den Code durchsuchen, um potenzielle Fehler oder ineffiziente Codestrukturen zu finden und diese zu korrigieren. Es kann auch bedeuten, dass sie den Code umschreiben, um ihn verständlicher und leichter zu warten zu machen.

Die Pfadfinderregel besagt, dass Entwickler den Code so strukturieren sollten, dass er leicht lesbar, verständlich und erweiterbar ist. Dies kann bedeuten, dass sie den Code in kleine, gut strukturierte Module aufteilen, die klar definierte Aufgaben erfüllen. Entwickler sollten auch aussagekräftige Variablennamen, Funktionen und Klassen verwenden, um den Code verständlicher zu machen, sowie ausreichende Kommentare hinzufügen, um die Funktionsweise und die Absicht des Codes zu beschreiben.

Zusammenfassend zielen sowohl das Boy-Scout-Prinzip als auch die Pfadfinderregel darauf ab, den Code lesbarer, verständlicher und wartbarer zu machen, um sicherzustellen, dass die Software immer auf dem neuesten Stand bleibt und bereit ist, den Anforderungen der Benutzer und der Organisation gerecht zu werden.

Boy-Scouting-Principle - Pfadfinderregel

Zusammenfassung

Die SOLID-Design-Prinzipien sind eine Gruppe von fünf Prinzipien, die von Robert C. Martin vorgeschlagen wurden, um Software-Designs zu verbessern. Jedes dieser Prinzipien stellt einen grundlegenden Leitfaden dar, der dabei helfen soll, die Qualität, die Wartbarkeit und die Erweiterbarkeit von Software zu verbessern.

Die fünf SOLID-Prinzipien sind:

Single Responsibility Principle (SRP)

Single Responsibility Principle (SRP): Eine Klasse sollte nur eine Verantwortlichkeit haben. Das bedeutet, dass sie nur eine Aufgabe erfüllen und nur für eine Art von Änderungen anfällig sein sollte.

```php
<?php


// bad
class Customer {
    private $name;
    private $email;

    public function __construct($name, $email) {
        $this->name = $name;
        $this->email = $email;
    }

    public function getName() {
        return $this->name;
    }

    public function getEmail() {
        return $this->email;
    }

    public function sendEmail($subject, $body) {
        // Code zum Senden einer E-Mail
    }
}
/*
In diesem Beispiel ist die `Customer`-Klasse sowohl für das Speichern von Kundeninformationen und das Bereitstellen von Zugriffsmethoden als auch für das Senden von E-Mails verantwortlich. Die Klasse hat damit mehrere Verantwortlichkeiten und ist somit schwerer wartbar und erweiterbar.
*/

// good
class Customer {
```

```
    private $name;
    private $email;

    public function __construct($name, $email) {
        $this->name = $name;
        $this->email = $email;
    }

    public function getName() {
        return $this->name;
    }

    public function getEmail() {
        return $this->email;
    }
}

class EmailSender {
    public function sendEmail(Customer $customer) {
        $email = $customer->getEmail();
        // Code zum Senden einer E-Mail
    }
}
/*
In diesem Beispiel ist die `Customer`-Klasse nur für das Speichern von Kundeninformationen und das Bereitstellen von Zugriffsmethoden verantwortlich. Die `EmailSender`-Klasse ist nur für das Senden von E-Mails zuständig. Beide Klassen erfüllen jeweils nur eine Verantwortlichkeit.
*/
```

Open-Closed Principle (OCP)

Software-Entitäten (Klassen, Module, Funktionen usw.) sollten offen für Erweiterungen, aber geschlossen für Änderungen sein. Das bedeutet, dass die Funktionalität einer Software-Einheit durch Hinzufügen neuer Code-Module oder Klassen erweitert werden sollte, ohne den bestehenden Code ändern zu müssen.

```
// bad
class AreaCalculator {
    public function calculate($shapes) {
        $area = 0;

        foreach ($shapes as $shape) {
            if ($shape instanceof Circle) {
                $area += pi() * pow($shape->getRadius(), 2);
            } elseif ($shape instanceof Rectangle) {
                $area += $shape->getLength() * $shape->getWidth();
            }
            // Weitere Bedingungen für andere Formen
        }

        return $area;
    }
}
/*
```

In diesem Beispiel ist die `AreaCalculator`-Klasse offen für Modifikationen, da sie geändert werden muss, wenn eine neue Form hinzugefügt wird. Die Klassen `Circle` und `Rectangle` sind auch offen für Modifikationen, da sie neue Methoden benötigen, wenn neue Formen hinzugefügt werden.
*/

```
// good
 interface Shape {
    public function area();
}

class Circle implements Shape {
    private $radius;

    public function __construct($radius) {
        $this->radius = $radius;
    }

    public function area() {
        return pi() * pow($this->radius, 2);
    }
}

class Rectangle implements Shape {
    private $length;
    private $width;

    public function __construct($length, $width) {
        $this->length = $length;
        $this->width = $width;
    }

    public function area() {
        return $this->length * $this->width;
    }
}

class AreaCalculator {
    public function calculate(Shape $shape) {
        return $shape->area();
    }
}

/*
In diesem Beispiel sind die Klassen `Circle` und `Rectangle` geschlossen für Änderungen, da sie ihre Funktionalität nicht ändern, wenn neue Formen hinzugefügt werden. Die `AreaCalculator`-Klasse ist offen für Erweiterungen, da sie einfach erweitert werden kann, um die Berechnung der Fläche neuer Formen zu unterstützen.
*/
 ```

Liskov Substitution Principle (LSP)

Liskov Substitution Principle (LSP): Objekte einer abgeleiteten Klasse sollten durch Objekte ihrer Basisklasse ersetzbar sein, ohne das Programm zu beeinträchtigen. Das bedeutet, dass eine abgeleitete Klasse alle Verhaltensweisen ihrer Basisklasse übernehmen sollte, ohne dass das Verhalten des Programms sich ändert.

```php
<?php
```

```php
// bad
class Vehicle {
    protected $fuel;

    public function refuel($fuel) {
        // Code zum Betanken des Fahrzeugs
    }

    public function accelerate() {
        // Code zum Beschleunigen des Fahrzeugs
    }
}

class Car extends Vehicle {
    public function refuel($fuel) {
        if ($fuel > 50) {
            throw new Exception('Too much fuel');
        }

        parent::refuel($fuel);
    }

    public function accelerate() {
        // Code zum Beschleunigen des Autos
    }
}

/*
In diesem Beispiel verletzt die `Car`-Klasse das Liskov Substitution Principle, da sie die Vertragsbedingungen der `Vehicle`-Klasse verletzt. Die `refuel()`-Methode der `Car`-Klasse führt eine zusätzliche Überprüfung durch, um sicherzustellen, dass nicht zu viel Kraftstoff hinzugefügt wird. Wenn ein Code-Abschnitt den Aufruf der `refuel()`-Methode auf einer `Vehicle`-Instanz aufruft und stattdessen eine `Car`-Instanz erhält, führt die zusätzliche Überprüfung möglicherweise zu unerwarteten Fehlern oder Ausnahmen.
*/


// good
class Vehicle {
    public function startEngine() {
        // Code zum Starten des Motors
    }
}

class Car extends Vehicle {
    public function startEngine() {
        parent::startEngine();
        // Code speziell für Autos zum Starten des Motors
    }
}

class ElectricCar extends Car {
    public function startEngine() {
        // Code zum Starten des Elektromotors
    }
}
```

```
/*
In diesem Beispiel kann die `ElectricCar`-Klasse ohne Probleme anstelle der `Car`-Klasse verwendet werden, da sie alle Methoden und Verhaltensweisen der `Car`-Klasse erbt und sie auch überschreiben kann, wenn dies erforderlich ist.
*/
```

Interface Segregation Principle (ISP)

Interface Segregation Principle (ISP): Die Schnittstellen sollten auf die spezifischen Bedürfnisse der Clients zugeschnitten sein. Das bedeutet, dass eine Klasse nicht gezwungen werden sollte, Methoden oder Eigenschaften zu implementieren, die sie nicht benötigt.

```php
<?php

// bad
interface Animal {
    public function move();
    public function fly();
    public function swim();
}

class Bird implements Animal {
    public function move() {
        // Code zur Bewegung eines Vogels
    }

    public function fly() {
        // Code zum Fliegen eines Vogels
    }

    public function swim() {
        throw new Exception('Birds cannot swim');
    }
}

class Fish implements Animal {
    public function move() {
        // Code zur Bewegung eines Fisches
    }

    public function fly() {
        throw new Exception('Fish cannot fly');
    }

    public function swim() {
        // Code zum Schwimmen eines Fisches
    }
}
/*
In diesem Beispiel verletzt die `Animal`-Schnittstelle das Interface Segregation Principle, da sie Methoden für Fliegen, Schwimmen und Bewegung kombiniert, obwohl nicht alle Tiere diese Fähigkeiten besitzen. Die `Bird`-Klasse muss die `swim()`-Methode implementieren, obwohl Vögel nicht schwimmen können, und die `Fish`-Klasse muss die `fly()`-Methode implementieren, obwohl Fische nicht fliegen können. Dies führt zu unerwarteten Fehlern oder Ausnahmen, wenn der Code auf eine Instanz von `Bird` oder `Fish` zugreift und eine unerwartete Methode
```

```
aufruft.
*/

// good
interface Animal {
    public function move();
}

interface FlyableAnimal extends Animal {
    public function fly();
}

interface SwimmableAnimal extends Animal {
    public function swim();
}

class Bird implements FlyableAnimal {
    public function move() {
        // Code zur Bewegung eines Vogels
    }

    public function fly() {
        // Code zum Fliegen eines Vogels
    }
}

class Fish implements SwimmableAnimal {
    public function move() {
        // Code zur Bewegung eines Fisches
    }

    public function swim() {
        // Code zum Schwimmen eines Fisches
    }
}

/*
In diesem Beispiel haben wir das Interface Segregation Principle angewendet, indem wir separate Schnittstellen für Fliegen und Schwimmen definiert haben, die jeweils von der allgemeinen `Animal`-Schnittstelle erben. Dadurch können wir sicherstellen, dass Klassen, die nicht fliegen können, nicht gezwungen werden, eine `fly()`-Methode zu implementieren, und Klassen, die nicht schwimmen können, nicht gezwungen werden, eine `swim()`-Methode zu implementieren. In diesem Fall implementieren die `Bird`- und `Fish`-Klassen nur die Methoden, die für ihre jeweilige Art der Fortbewegung relevant sind.
*/
```

Dependency Inversion Principle (DIP)

Dependency Inversion Principle (DIP): Abhängigkeiten sollten auf abstrakten Schnittstellen basieren, nicht auf konkreten Implementierungen. Das bedeutet, dass höhere Module nicht von niedrigeren Modulen abhängig sein sollten und dass abstrakte Module nicht von konkreten Modulen abhängig sein sollten.

```
<?php
```

```
// bad
class MySqlDatabase {
    public function connect() {
        // Code zur Verbindung mit einer MySQL-Datenbank
    }
}

class UserRepository {
    private $database;

    public function __construct() {
        $this->database = new MySqlDatabase();
    }

    public function getUsers() {
        $this->database->connect();
        // Code zum Abrufen von Benutzern aus der Datenbank
    }
}
/*
In diesem Beispiel wird das Dependency Inversion Principle verletzt, da die `UserRepository`-Klasse direkt an die konkrete Implementierung `MySqlDatabase` gebunden ist. Wenn wir eine andere Datenbank wie PostgreSQL verwenden möchten, müssen wir den Code in der `UserRepository`-Klasse ändern. Das führt dazu, dass der Code starr und schwer erweiterbar wird. Um das DIP zu wahren, sollte die `UserRepository`-Klasse die Abstraktion `DatabaseInterface` verwenden, statt direkt an die konkrete Implementierung gebunden zu sein.
*/

// good
interface DatabaseInterface {
    public function connect();
}

class MySqlDatabase implements DatabaseInterface {
    public function connect() {
        // Code zur Verbindung mit einer MySQL-Datenbank
    }
}

class PostgreSqlDatabase implements DatabaseInterface {
    public function connect() {
        // Code zur Verbindung mit einer PostgreSQL-Datenbank
    }
}

class UserRepository {
    private $database;

    public function __construct(DatabaseInterface $database) {
        $this->database = $database;
    }

    public function getUsers() {
        $this->database->connect();
        // Code zum Abrufen von Benutzern aus der Datenbank
    }
}
/*
```

In diesem Beispiel haben wir das Dependency Inversion Principle angewendet, indem wir eine Abstraktion `DatabaseInterface` geschaffen haben, die von konkreten Implementierungen `MySqlDatabase` und `PostgreSqlDatabase` implementiert wird. Anstatt die konkreten Implementierungen direkt in der `UserRepository`-Klasse zu instanziieren, injizieren wir die Abstraktion durch den Konstruktor. Dadurch wird die `UserRepository`-Klasse nicht mehr direkt an konkrete Implementierungen gebunden, sondern nur an die Abstraktion `DatabaseInterface`. Dies erleichtert das Austauschen der verwendeten Datenbank und fördert die Flexibilität und Wiederverwendbarkeit des Codes.
*/

Pest

composer.json

```
{
    "require-dev": {
        "pestphp/pest": "^1.22",
        "pestphp/pest-plugin-laravel": "^1.3",
        "pestphp/pest-plugin-livewire": "^1.0",
        "phpstan/extension-installer": "^1.1",
        "phpstan/phpstan-deprecation-rules": "^1.0",
        "phpstan/phpstan-phpunit": "^1.0",
        "phpunit/php-code-coverage": "^9.2",
        "phpunit/phpunit": "^9.5"
    },
    "scripts": {
        "test:pest": "vendor/bin/pest --order-by default -d memory_limit=6144M",
        "test:pest-coverage": "php -dpcov.enabled=1 -dpcov.directory=. -dpcov.exclude='~vendor~' vendor/bin/pest -d
memory_limit=6144M --testdox --verbose --coverage --min=85",
    "test:unit": "vendor/bin/testbench package:test --no-coverage",
    "test:types": "vendor/bin/phpstan analyse",
    "test": [
        "@lint:fix",
        "@test:types",
        "@test:unit"
    ]
}
```

Git

Force push git branches

```
git push --force origin main
git push -f origin main
git push origin +main
```

- Original tweet by Stefan Judis

Github key anpassen bzw. bereinigen

```
composer update --lock
```

```
composer update --lock s
```

Issues verlinken

Sie können einen Pull-Request oder Branch mit einem Issue verknüpfen, um anzuzeigen, dass eine Behebung in Bearbeitung ist, und um das Issue automatisch zu schließen, wenn der Pull-Request oder Branch zusammengeführt wird.

Hinweis: Die speziellen Schlüsselwörter in einer Pull-Request-Beschreibung werden interpretiert, wenn die Pull-Request auf den *Default* - Branch des Repositorys abzielt. Wenn die Basis des PR jedoch *ein anderer Zweig* ist, werden diese Schlüsselwörter ignoriert, es werden keine Links erstellt und das Zusammenführen des PR hat keine Auswirkung auf die Ausgaben. **Wenn Sie einen Pull-Request über ein Schlüsselwort mit einem Issue verknüpfen möchten, muss sich der PR auf dem Standard-Branch befinden.**

Über verknüpfte Probleme und Pull-Requests

Sie können ein Problem manuell mit einer Pull-Anforderung verknüpfen oder ein unterstütztes Schlüsselwort in der Beschreibung der Pull-Anforderung verwenden.

Wenn Sie eine Pull-Anforderung mit dem Problem verknüpfen, auf das sich die Pull-Anforderung bezieht, können Mitbearbeiter sehen, dass jemand an dem Problem arbeitet.

Wenn Sie einen verknüpften Pull-Request mit dem Standard-Branch eines Repositorys zusammenführen, wird das verknüpfte Issue automatisch geschlossen. Weitere Informationen zum Standard-Zweig finden Sie unter „ Ändern des Standard-Zweigs “.

Verknüpfen einer Pull-Anforderung mit einem Problem mithilfe eines Schlüsselworts

Sie können eine Pull-Anforderung mit einem Problem verknüpfen, indem Sie ein unterstütztes Schlüsselwort in der Beschreibung der Pull-Anforderung oder in einer Commit-Nachricht verwenden. Die Pull-Anfrage **muss sich** im Standard-Branch befinden.

- nah dran
- schließt
- abgeschlossen
- Fix
- behebt
- Fest

- beschließen
- beschließt
- aufgelöst

Wenn Sie ein Schlüsselwort verwenden, um auf einen Pull-Request-Kommentar in einem anderen Pull-Request zu verweisen, werden die Pull-Requests verknüpft. Durch das Zusammenführen der referenzierenden Pull-Anforderung wird auch die referenzierte Pull-Anforderung geschlossen.

Die Syntax zum Schließen von Schlüsselwörtern hängt davon ab, ob sich das Problem im selben Repository wie die Pull-Anfrage befindet.

Verknüpftes Problem	**Syntax**	**Beispiel**
Ausgabe im selben Repository	*SCHLÜSSELWORT* # AUSGABE - *NUMMER*	`Closes #10`
Ausgabe in einem anderen Repository	*KEYWORD OWNER* / *REPOSITORY* # *ISSUE-NUMBER*	`Fixes octo-org/octo-repo#100`
Mehrere Probleme	Verwenden Sie für jedes Problem die vollständige Syntax	`Resolves #10, resolves #123, resolves octo-org/octo-repo#100`

Nur manuell verknüpfte Pull-Requests können manuell entkoppelt werden. Um die Verknüpfung eines Problems aufzuheben, das Sie mit einem Schlüsselwort verknüpft haben, müssen Sie die Pull-Request-Beschreibung bearbeiten, um das Schlüsselwort zu entfernen.

Sie können auch schließende Schlüsselwörter in einer Commit-Nachricht verwenden. Das Problem wird geschlossen, wenn Sie den Commit in den Standard-Branch zusammenführen, aber die Pull-Anforderung, die den Commit enthält, wird nicht als verknüpfte Pull-Anforderung aufgeführt.

Jeder mit Schreibberechtigungen für ein Repository kann eine Pull-Anforderung manuell mit einem Vorgang über die Seitenleiste für Pull-Anforderungen verknüpfen.

Sie können bis zu zehn Issues manuell mit jeder Pull-Anfrage verknüpfen. Das Issue und die Pull-Anforderung müssen sich im selben Repository befinden.

1. Navigieren Sie auf GitHub.com zur Hauptseite des Repositorys.
2. Klicken Sie unter Ihrem Repository-Namen auf **Pull-Anforderungen** .

   Auswahl der Registerkarte „Probleme und Pull-Requests“.
3. Klicken Sie in der Liste der Pull-Requests auf den Pull-Request, den Sie mit einem Issue verknüpfen möchten.
4. Klicken Sie in der rechten Seitenleiste im Abschnitt "Entwicklung" auf .

5. Klicken Sie auf das Problem, das Sie mit der Pull-Anforderung verknüpfen möchten. Drop-down zum Link-Problem

Jeder mit Schreibberechtigungen für ein Repository kann eine Pull-Anfrage manuell verknüpfen oder von der Issue-Seitenleiste zu einem Issue verzweigen.

Sie können bis zu zehn Issues manuell mit jeder Pull-Anfrage verknüpfen. Das Problem kann sich in einem anderen Repository befinden als der verknüpfte Pull-Request oder Branch. Ihr zuletzt ausgewähltes Repository wird gespeichert

1. Navigieren Sie auf GitHub.com zur Hauptseite des Repositorys.
2. Klicken Sie unter Ihrem Repository-Namen auf **Issues** .

   Registerkarte "Probleme".
3. Klicken Sie in der Liste der Issues auf das Issue, mit dem Sie eine Pull-Anfrage oder einen Branch verknüpfen möchten.
4. Klicken Sie in der rechten Seitenleiste auf **Entwicklung** . Entwicklungsmenü in der rechten Seitenleiste
5. Klicken Sie auf das Repository mit der Pull-Anfrage oder dem Zweig, den Sie mit dem Problem verknüpfen möchten. Drop-down, um das Repository auszuwählen
6. Klicken Sie auf den Pull-Request oder Branch, den Sie mit dem Issue verknüpfen möchten. Drop-down, um Pull-Request oder Branch zu verknüpfen
7. Klicken Sie auf **Anwenden** . Anwenden

Weiterlesen

- " Automatisch verlinkte Verweise und URLs "

Arr

Throw an exception

```php
<?php

$config = ['.....'];
$apiKey = Arr::get($config, 'api_key', fn () => throw new Exeption('your message here'));

// or
$apiKey = $config['api_key'] ?? throw new Exeption('your message here'));
```

- Original tweet by Steve Bauman

Collection

sort

```php
<?php

$exampleEntries = [
    'my example exa' => 'value_4',
    'my example' => 'value_1',
    'my example ex' => 'value_3',
    'my example e' => 'value_2',
];

$result = collect($exampleEntries)->sort()->toArray();

$result = [
    "my example" => "value_1",
    "my example e" => "value_2",
    "my example ex" => "value_3",
    "my example exa" => "value_4",
]
```

sortKeys

```php
<?php

$exampleEntries = [
    'key_1' => 'value_1',
    'key_3' => 'value_3',
    'key_2' => 'value_2',
    'key_7' => 'value_7',
    'key_4' => 'value_4',
    'key_6' => 'value_6',
    'key_5' => 'value_5',
];

$result = collect($exampleEntries)->sortKeys()->toArray();

$result = [
    "key_1" => "value_1",
    "key_2" => "value_2",
    "key_3" => "value_3",
    "key_4" => "value_4",
    "key_5" => "value_5",
    "key_6" => "value_6",
    "key_7" => "value_7",
]
```

Eloquent

Touch eloquent method

```php
<?php

$user = User::find(1);

// instead of
$user->update(['subscribed_at' => now()]);

// use
$user->touch('subscribed_at');
```

- Original tweet by Oussama Sid

Migrations

Unique columns

Laravel Example 1:

```
<?php

DB::update("
   ALTER TABLE tournament_league_game_days
   ADD COLUMN game_schedule_day_unique varchar (512)
   GENERATED ALWAYS AS
   (
      CONCAT(
         CONCAT(day, '#', game_schedule_id),
         '#',
         IF(deleted_at IS NULL, '-',  deleted_at)
      )
   ) VIRTUAL;

");

DB::update("
   CREATE UNIQUE INDEX game_schedule_day_unique ON tournament_league_game_days
(game_schedule_day_unique);
");
```

Laravel Example 2:

```
<?php

Schema::table('tournament_league_game_days', function (Blueprint $table) {
   $table->string('game_schedule_day_unique')
      ->virtualAs(
         DB::raw(
            "CONCAT(
               CONCAT(day, '#', game_schedule_id),
               '#',
               IF(deleted_at IS NULL, '-',  deleted_at)
            )"
         )
      );
});

Schema::table('tournament_league_game_days', function (Blueprint $table) {
   $table->unique(['game_schedule_day_unique'], 'game_schedule_day_unique_index');
});
```

- Original tweet by Tobias_Petry.sql

Route

Test-Route

```php
<?php

use Illuminate\Support\Facades\Route;

Route::get('/test-route', function () {
    // return your example
})
```

withoutMiddleware

```php
<?php

use App\Http\Middleware\FirstMiddleware;
use App\Http\Middleware\SecondMiddleware;
use Illuminate\Support\Facades\Route;

Route::prefix('my-prefix')
    ->middleware([
        FirstMiddleware::class,
        SecondMiddleware::class,
    ])
    ->group(function () {
        Route::get('my-route', function () {
            // Uses first & second middleware
        });
        Route::withoutMiddleware(FirstMiddleware::class)
            ->get('route-without-first-middleware', function () {
                // Uses second middleware
            });
        Route::withoutMiddleware(SecondMiddleware::class)
            ->get('route-without-second-middleware', function () {
                // Uses first middleware
            });
        Route::withoutMiddleware([
            FirstMiddleware::class,
            SecondMiddleware::class,
        ])->get('route-without-middleware', function () {
            // Uses no middleware
        }
    );
});
```

with Parameters

```php
<?php

namespace App\Http\Middleware;

use Closure;

class EnsureUserHasRole
{
    public function handle($request, Closure $next, $role)
    {
        if (! $request->user()->hasRole($role)) {
            // Redirect...
        }

        return $next($request);

    }
}
```

```php
<?php

use Illuminate\Support\Facades\Route;

Route::put('/post/{id}', function ($id) {
    //
})->middleware('role:editor');
```

Tests

expectException and expectExceptionMessage

```php
<?php

use Application\User\Queries\ListUserQuery;
use Illuminate\Http\Request;
use Spatie\QueryBuilder\Exceptions\InvalidFilterQuery;
use Tests\TestCase;

class ListUserQueryTest extends TestCase
{
    /** @test */
    public function it_throws_an_exception_when_the_key_for_filtering_is_not_supported(): void
    {
        $this->expectException(InvalidFilterQuery::class);
        $this->expectExceptionMessage('Requested filter(s) `key_not_supported` are not allowed. Allowed filter(s) are `id, email, nickname`.');

        $request = new Request(['filter' => ['key_not_supported' => 'value is irrelevant']]);

        new ListMyModelQuery($request);
    }
}
```

it_uses_the_right_query_filters

```php
<?php

    /** @test */
    public function it_uses_the_right_query_filters(): void
    {
        $this->assertQueryFilterEquals(
            ListMyModelQuery::class,
            [
                AllowedFilter::exact('id'),
                AllowedFilter::exact('handle'),
                AllowedFilter::partial('name', 'display_name'),
            ]
        );
    }
```

assertThrows | it_throws_an_error_if_model_doesnt_exist

```php
<?php
```

```
    /** @test */
    public function it_throws_an_error_if_model_doesnt_exist(): void
    {
        $className = User::class;

        $this->assertThrows(
            fn () => User::findOrFail(0),
            ModelNotFoundException::class,
            "No query results for model [{$className}] 1"
        );
    }
```

it_uses_the_right_query_class

```
<?php

    /** @test */
    public function it_uses_the_right_query_class(): void
    {
        $query = $this->mock(
            ListMyModelQuery::class,
            fn (MockInterface $mock) => $mock->shouldReceive('simplePaginate')
                ->once()
                ->andReturn(MyModelFactory::new()->create()->simplePaginate()),
        );

        $controller = new ViewMyModelListController();
        $controller($query, new Request());

        $this->assertControllerUsesClass(
            ViewMyModelListController::class,
            ListMyModelQuery::class
        );
    }
```

it_uses_the_right_collection

```
<?php

    /** @test */
    public function it_uses_the_right_collection(): void
    {
        $this->assertControllerReturns(
            ViewMyModelListController::class,
            MyModelCollection::class
        );
    }
```

it_uses_the_right_middleware

```php
<?php

    public function middlewares(): array
    {
        return [
            'group1' => ['group1_sub1', 'group1_sub2'],
            'group2' => ['group2_sub1', 'group2_sub2'],
            MyMiddleware::class => [MyMiddleware::class],
        ];
    }

    /**
     * @test
     *
     * @dataProvider middlewares
     */
    public function it_uses_the_right_middleware($middleware)
    {
        $this->assertControllerUsesMiddleware(
            ViewMyModelListController::class,
            $middleware,
        );
    }
```

it_returns_the_right_structure | Collection

```php
<?php

    /** @test */
    public function it_returns_the_right_structure(): void
    {
        MyModelFactory::new()->create();

        $response = $this->createResource(MyModelCollection::class, MyModel::simplePaginate());

        $response->assertJsonStructureExact([
            'data',
            'links',
            'meta',
        ]);

        $this->assertEquals(MyModelResource::class, MyModelCollection::make([])->collects);
    }
```

it_returns_the_right_structure | Resource

```php
<?php
```

```
    /** @test */
    public function it_returns_the_right_structure(): void
    {
        $response = $this->createResource(
            MyModelResource::class,
            MyModelFactory::new()->create()
        );

        $response->assertJsonStructureExact([
            'id',
            'my_column_one',
            'my_column_two',
            'created_at',
            'updated_at',
        ]);

        $response->assertJson(
            fn (AssertableJson $json) => $json
                ->whereAllType([
                    'id' => ['integer'],
                    'my_column_one' => ['string'],
                    'my_column_two' => [null, 'string'],
                    'created_at' => ['string'],
                    'updated_at' => ['string'],
                ])
                ->etc()
        );
    }
```

examples_for_mock_actions

```
<?php

    /** @test */
    public function examples_for_mock_actions(): void
    {
        $myAction = $this->spy(MyActionn::class);
        $myAction->shouldReceive('execute')
            ->once()
            ->withArgs(fn ($arg) => $arg === 'foo')
            ->andReturn('bar');

        // or
        $spy = $this->spy(MyActionn::class);
        $spy->shouldReceive('execute')
            ->once()
            ->with(MyExampleModel::class, MyExampleData::class)
            ->andReturn('bar');
    }
```

Event::fake

Event:: assertDispatched

```php
<?php

    /** @test */
    public function event_assert_dispatched(): void
    {
        Event::fake();

        // or

        Event::fake([
            ExampleCreated::class,
        ]);

        $myExample = MyExampleFactory::new()->create();

        // Assert a event was dispatched...
        Event::assertDispatched(ExampleCreated::class);

        // Assert a event was dispatched...
        Event::assertDispatched(ExampleCreated::class, function ($event) use ($myExample) {
            return $event->myExample->is($myExample);
        });
    }
```

Event:: assertNotDispatched

```php
<?php

    /** @test */
    public function event_assert_not_dispatched(): void
    {
        Event::fake();

        // or

        Event::fake([
            ExampleCreated::class,
        ]);

        $myExample = MyExampleFactory::new()->create();

        // Assert a event was not dispatched...
        Event::assertNotDispatched(ExampleCreated::class);

        Event::assertNotDispatched(ExampleCreated::class, function ($event, $payload) {
            return $payload[0]->name === 'John Doe';
        });
    }
```

Event:: assertDispatchedTimes

```php
<?php

    /** @test */
    public function event_assert_dispatched_times(): void
    {
        Event::fake();

        // or

        Event::fake([
            ExampleCreated::class,
        ]);

        $myExample = MyExampleFactory::new()->create();

        // Assert a event was dispatched exactly n times...
        Event::assertDispatchedTimes(ExampleCreated::class, 1);
    }
```

Event:: assertListening

```php
<?php

    /** @test */
    public function event_assert_listening(): void
    {
        Event::fake();

        // or

        Event::fake([
            ExampleCreated::class,
        ]);

        $myExample = MyExampleFactory::new()->create();

        // assert that a listener is listening to a given event
        Event::assertListening(ExampleCreated::class, ExampleListener::class);
    }
```

Queue::fake

Queue:: assertPushed

```php
<?php

    /** @test */
    public function examples_for_queue_fakes(): void
    {
```

```
        Queue::fake();

        // or

        Queue::fake([
            ExampleJob::class,
        ]);

        // queue a job
        ExampleJob::dispatch();

        // assert that a job was pushed...
        Queue::assertPushed(ExampleJob::class);

        // assert that a job was pushed a given number of times...
        Queue::assertPushed(ExampleJob::class, 1);

        // assert that a job was pushed with a given payload...
        Queue::assertPushed(ExampleJob::class, function ($job) {
            return $job->example == 'example';
        });
    }
```

Queue:: assertNotPushed

```
<?php

    /** @test */
    public function queue_assert_not_pushed(): void
    {
        Queue::fake();

        // or

        Queue::fake([
            ExampleJob::class,
        ]);

        // queue a job
        ExampleJob::dispatch();

        // assert that a job was not pushed
        Queue::assertNotPushed(ExampleJob::class);

        // asser that a job was not pushed with a given payload...
        Queue::assertNotPushed(ExampleJob::class, function (ExampleJob $job) {
            return $job->exampleProperty === 'exampleValue';
        });
    }
```

Queue:: assertNothingPushed

```php
<?php

    /** @test */
    public function queue_assert_nothing_pushed(): void
    {
        Queue::fake();

        // or

        Queue::fake([
            ExampleJob::class,
        ]);

        // queue a job
        ExampleJob::dispatch();

        // assert that no jobs were pushed...
        Queue::assertNothingPushed();
    }
```

Queue:: assertPushedOn

```php
<?php

    /** @test */
    public function queue_assert_pushed_on(): void
    {
        Queue::fake();

        // or

        Queue::fake([
            ExampleJob::class,
        ]);

        // queue a job
        ExampleJob::dispatch();

        // assert that a job was pushed on a given queue...
        Queue::assertPushedOn('queue-name', ExampleJob::class);

        // assert that a job was pushed a given number of times on a given queue...
        Queue::assertPushedOn('queue-name', ExampleJob::class, 1);

        // assert that a job was pushed with a given payload on a given queue...
        Queue::assertPushedOn('queue-name', ExampleJob::class, function ($job) {
            return $job->example == 'example';
        });
    }
```

Queue:: assertPushedWithChain

```php
<?php

    /** @test */
    public function queue_assert_pushed_with_chain(): void
    {
        Queue::fake();

        // or

        Queue::fake([
            ExampleJob::class,
        ]);

        // queue a job
        ExampleJob::dispatch();

        // assert that a job was pushed with a given chain...
        Queue::assertPushedWithChain(ExampleJob::class, [
            new AnotherJob,
            new YetAnotherJob,
        ]);

        // assert that a job was pushed with a given chain...
        Queue::assertPushedWithChain(ExampleJob::class, [
            new AnotherJob,
            new YetAnotherJob,
        ], function ($job) {
            return $job->user->id === 1;
        });
    }
```

Queue:: assertPushedWithoutChain

```php
<?php

    /** @test */
    public function queue_assert_pushed_without_chain(): void
    {
        Queue::fake();

        // or

        Queue::fake([
            ExampleJob::class,
        ]);

        // queue a job
        ExampleJob::dispatch();

        // assert that a job was pushed without a given chain...
        Queue::assertPushedWithoutChain(ExampleJob::class);

        // assert that a job was pushed without a given chain and with a given payload...
        Queue::assertPushedWithoutChain(ExampleJob::class, function ($job) {
```

```
        return $job->exampleProperty === 'exampleValue';
    });
}
```

Validation

prepareForValidation

```php
<?php

use Illuminate\Validation\Rule;
use Smake\Common\Http\Requests\Request;

class EditUserRequest extends Request
{
    protected function prepareForValidation()
    {
        parent::prepareForValidation();

        $this->merge([
            'nickname' => $this->input('nickname', '') . ' with Love',
        ]);
    }
}
```

Rule::unique with ignore method on update

Adding a record:

```php
<?php

use Illuminate\Validation\Rule;
use Smake\Common\Http\Requests\Request;

class AddUserRequest extends Request
{
    public function rules()
    {
        return [
            'email' => [
                'required',
                'email:strict',
                'max:255',
                Rule::unique('users', 'email'),
            ],
        ];
    }
}
```

Update an existing record:

```php
<?php

use Illuminate\Validation\Rule;
use Smake\Common\Http\Requests\Request;

class EditUserRequest extends Request
{
    public function rules()
    {
        return [
            'email' => [
                'required',
                'email:strict',
                'max:255',
                Rule::unique('users', 'email')->ignore($this->route('userId')),
            ],
        ];
    }
}
```

passedValidation

```php
<?php

use Illuminate\Validation\Rule;
use Smake\Common\Http\Requests\Request;

class EditUserRequest extends Request
{
    protected function passedValidation()
    {
        $this->merge([
            'nickname' => $this->input('nickname', '') . '❤️',
        ]);

        $this->getValidatorInstance()->setData(
            collect($this->input())
                ->only(collect($this->rules()]->keys())
                ->all()
        );
    }
}
```

Kommandos

mkdocs help

```
mkdocs --help
```

mkdocs new

```
mkdocs new
```

mkdocs build

```
mkdocs build
```

mkdocs serve

```
mkdocs serve
```

mkdocs serve --livereload

```
mkdocs serve --livereload
```

Install python3

Weitere Information siehe python3.de

Install mkdocs

```
pip3 install mkdocs
```

Pip Kommandos

pip3

```
pip3 --version
```

Plugins

mkdocs-glightbox

```
pip3 install mkdocs-glightbox
```

mkdocs-minify-plugin

```
pip3 install mkdocs-minify-plugin
```

mkdocs-static-i18n

```
pip3 install mkdocs-static-i18n
pip3 install mkdocs-static-i18n --use-pep517
```

mkdocs-admonition

```
pip3 install mkdocs-admonition
```

mkdocs-pdf-export-plugin

```
pip3 install mkdocs-pdf-export-plugin
```

Install python3

```
python3 -m ensurepip --upgrade
```

python3 --version

```
python3 --version
```

Templates

```
pip3 install mkdocs-material
```