



Published in Dev Genius

You have **2** free member-only stories left this month.

[Sign up for Medium and get an extra one](#)



.com software

Follow

Aug 9, 2022 · 2 min read · ✨ · [Listen](#)

Save



Clean code tricks in PHP everyone should follow

Improve your code drastically with these few simple tricks. Today I would like to share my coding techniques with you.



Photo by [Ben Griffiths](#) on [Unsplash](#)

I have a few techniques to share to improve your coding skills and make your code more friendly for people to read and digest.

I will write each technique before and after the change. Let us jump straight into the business.

Keep code indentation as small as possible

Before

```
1  <?php
2
3  class DiscountApplier
4  {
5      private DiscountCalculator $calculator;
6
7      public function __construct(DiscountCalculator $calculator)
8      {
9          $this->calculator = $calculator;
10     }
11
12     public function apply(Basket $basket): void
13     {
14         if ($basket->hasItems()) {
15             $discount = 0.0;
16
17             foreach ($basket->getItems() as $basketItem) {
18                 if ($basketItem->isTaxable()) {
19                     if ($basketItem->getPrice() > 0) {
20                         $discount += $this->calculator->getDiscount($basketItem);
21                     }
22                 }
23             }
24
25             $basket->applyDiscount($discount);
26         }
27     }
28 }
```

cc-indentation-before.php hosted with ❤ by [GitHub](#)[view raw](#)

After

```
1  <?php
2
3  class DiscountApplier
4  {
5      private DiscountCalculator $calculator;
6
7      public function __construct(DiscountCalculator $calculator)
8      {
9          $this->calculator = $calculator;
10     }
11
12     public function apply(Basket $basket): void
13     {
14         if (!$basket->hasItems()) {
15             return;
16         }
17
18         $discount = 0.0;
19
20         foreach ($this->getDiscountableItems($basket) as $basketItem) {
21             $discount += $this->calculator->getDiscount($basketItem);
22         }
23
24         $basket->applyDiscount($discount);
25     }
26
27     /**
28      * @return iterable<BasketItem>
29      */
30     private function getDiscountableItems(Basket $basket): iterable
31     {
32         foreach ($basket->getItems() as $basketItem) {
33             if (!$basketItem->isTaxable() || $basketItem->getPrice() === 0) {
34                 continue;
35             }
36
37             yield $basketItem;
38         }
39     }
40 }
```

cc-indentation-after.php hosted with ♥ by GitHub

[view raw](#)

Remove unnecessary condition clauses

Before

```
1  <?php
2
3  function mapOrderStatusToLabel(Order $order): string
4  {
5      if ($order->isComplete()) {
6          return 'Completed';
7      } elseif ($order->isPending()) {
8          return 'In transport';
9      } else {
10         return 'New';
11     }
12 }
```

cc-condition-clauses-before.php hosted with ♥ by GitHub

[view raw](#)

After

```
1  <?php
2
3  function mapOrderStatusToLabel(Order $order): string
4  {
5      if ($order->isComplete()) {
6          return 'Completed';
7      }
8
9      if ($order->isPending()) {
10         return 'In transport';
11     }
12
13     return 'New';
14 }
```

cc-condition-clauses-after.php hosted with ♥ by GitHub

[view raw](#)

Invert conditions to reduce code indentations

Before

```
1  <?php
2
3  function act(bool $success)
4  {
5      if ($success) {
6          // towering block of code
7          // towering block of code
8          // towering block of code
9          // towering block of code
10         // towering block of code
11         // towering block of code
12         // towering block of code
13     } else {
14         // one liner
15     }
16 }
```

cc-invert-conditions-before.php hosted with ❤ by GitHub

[view raw](#)

After

```
1  <?php
2
3  function act(bool $success)
4  {
5      if (!$success) {
6          // one liner
7          return;
8      }
9
10     // towering block of code
11     // towering block of code
12     // towering block of code
13     // towering block of code
14     // towering block of code
15     // towering block of code
16     // towering block of code
17 }
```

cc-invert-conditions-after.php hosted with ❤ by GitHub

[view raw](#)

Prefer switches over if-conditions when possible

Before

```
1  <?php
2
3  function mapOrderStatusToLabel(Order $order): string
4  {
5      $label = 'unknown';
6
7      if ('complete' === $order->getStatus()) {
8          $label = 'Completed';
9      }
10
11     if ('pending' === $order->getStatus()) {
12         $label = 'In transport';
13     }
14
15     if ('new' === $order->getStatus()) {
16         $label = 'New';
17     }
18
19     return $label;
20 }
```

cc-switches-before.php hosted with ♥ by GitHub

[view raw](#)

After

```
1  <?php
2
3  function mapOrderStatusToLabel(Order $order): string
4  {
5      switch ($order->getStatus()) {
6          case 'complete':
7              return 'Completed';
8
9          case 'pending':
10         case 'in_transport':
11             return 'In transport';
12
13         case 'new':
14             return 'New';
15
16         default:
17             return 'unknown';
18     }
19 }
```

cc-switches-after.php hosted with ♥ by GitHub

[view raw](#)

Reformat complex equations for brevity

Before

```
1  <?php
2
3  function getTotalWithTax(Payment $payment, int $taxPercent): float
4  {
5      // is the order correct at all?
6
7      return (float) (int) $payment->getTotal() * (1 + $taxPercent / 100);
8  }
```

cc-parenthesis-before.php hosted with ♥ by GitHub

[view raw](#)

After

```
1  <?php
2
3  function getTotalWithTax(Payment $payment, int $taxPercent): float
4  {
5      // descriptive variables with intermediary values for easier debugging
6      $paymentAmount = (int) $payment->getAmount();
7      $taxCoefficient = 1 + ($taxPercent / 100);
8
9      $totalWithTax = $paymentAmount * $taxCoefficient;
10
11     return (float) $totalWithTax;
12 }
```

cc-parenthesis-after.php hosted with ♥ by GitHub

[view raw](#)

Rearrange conditions in if-statements

Before

```
1  <?php
2
3  if ($httpApi->hasAccess($user) && $user->isActive() && $isSuccess) {
4      // all conditions must be met in order for this condition to execute
5      // in case of 'and' we should keep the heaviest conditions last
6      // and the fastest to execute in front
7  }
```

cc-conditions-before.php hosted with ♥ by GitHub

[view raw](#)

After

```
1  <?php
2
3  if ($isSuccess && $user->isActive() && $httpApi->hasAccess($user)) {
4      // if $isSuccess is false
5      // other conditions will not be evaluated
6  }
```

cc-conditions-after.php hosted with ♥ by GitHub

[view raw](#)

Prefer named methods over flags

Before

```
1  <?php
2
3  class UserRepository
4  {
5      public function getUsers(?bool $includingDeleted = false): array
6      {
7      }
8  }
```

cc-flags-before.php hosted with ♥ by GitHub

[view raw](#)

After


```
1  <?php
2
3  class UserRepository
4  {
5      public function getUsers(): array
6      {
7      }
8
9      public function getUsersIncludingDeleted(): array
10     {
11     }
12 }
```

cc-flags-after.php hosted with ❤ by GitHub

[view raw](#)

Use array destructuring

Before

```
1  <?php
2
3  $workers = [
4      ['id' => 1, 'name' => 'John Doe'],
5      ['id' => 3, 'name' => 'Jane Doe'],
6      ['id' => 4, 'name' => 'Monica Trullo'],
7      ['id' => 5, 'name' => 'Jonathan Bank'],
8  ];
9
10 foreach ($workers as $worker) {
11     $workerId = $worker['id'];
12     $workerName = $worker['name'];
13
14     // ...
15 }
```

cc-array-destruct-before.php hosted with ❤ by GitHub

[view raw](#)

After

```
1  <?php
2
3  $workers = [
4      ['id' => 1, 'name' => 'John Doe'],
5      ['id' => 3, 'name' => 'Jane Doe'],
6      ['id' => 4, 'name' => 'Monica Trullo'],
7      ['id' => 5, 'name' => 'Jonathan Bank'],
8  ];
9
10 foreach ($workers as ['id' => $workerId, 'name' => $workerName]) {
11     // ...
12 }
```

cc-array-destruct-after.php hosted with ♥ by GitHub

[view raw](#)

Document output structures with annotations

Before

```
1  <?php
2
3  /**
4   * @return array
5   */
6  function basketToViewModel(Basket $basket): array
7  {
8      $items = \array_map(function (BasketItem $item): array {
9          return [
10             'id' => $item->getProductId(),
11             'name' => $item->getProductName(),
12         ];
13     }, $basket->getItems());
14
15     return [
16         'total' => $basket->getTotal(),
17         'items' => $items,
18     ];
19 }
```

cc-output-before.php hosted with ♥ by GitHub

[view raw](#)

After

```
1  <?php
2
3  /**
4   * @return array{total: float, items: array<array{id: int, name: string}>}
5   */
6  function basketToViewModel(Basket $basket): array
7  {
8      // function body skipped for brevity
9
10     // output annotations especially useful if using tools like PsalmPHP
11     // PHPStorm is increasing its support for generics too
12
13     // see how amazing validation can you get
14     // https://psalm.dev/r/4a32324404
15 }
```

cc-output-after.php hosted with ❤ by GitHub

[view raw](#)

Document output primitives with annotations

Before

```
1  <?php
2
3  class User {
4      public const STATUS_NEW = 'new';
5      public const STATUS_ACTIVE = 'active';
6      public const STATUS_BLOCKED = 'blocked';
7
8      private string $status = self::STATUS_NEW;
9      private int $age;
10
11     public function getStatus(): string
12     {
13         return $this->status;
14     }
15
16     public function getAge(): int
17     {
18         return $this->age;
19     }
20
21     public function getClassName(): string
22     {
23         return get_class($this);
24     }
25 }
```

cc-output-prim-before.php hosted with ♥ by GitHub

[view raw](#)

After

[Open in app](#)[Sign up](#)[Sign In](#)

```
5  / **
6      * @return self::STATUS_*
7      */
8  public function getStatus(): string
9  {
10     return $this->status;
11 }
12
13 / **
14     * @return positive-int
15     */
16 public function getAge(): int
17 {
18     return $this->age;
19 }
20
21 / **
22     * @return class-string<User>
23     */
24 public function getClassName(): string
25 {
26     return get_class($this);
27 }
28 }
```

cc-output-prim-after.php hosted with ❤ by GitHub

[view raw](#)

I would also like to remind you about the following things when writing any code.

Remember that you're writing the code for your future self. The code may seem obvious at the moment. Let it stay like that next month:

- prefer descriptive variable names `$index` over `$i`
- prefer descriptive method names `getProductTotalForDisplay` over `getTotal`
- prefer descriptive constant names `const CACHE_TTL_WEEK = 7 * 86400` over `const TTL = 604800`

- use any static analysis tool, I recommend Psalm

[PHP](#)[Php Developers](#)[Php Development](#)[Clean Code](#)[Laravel](#)

Enjoy the read? Reward the writer.^{Beta}

Your tip will go to .com software through a third-party platform of their choice, letting them know you appreciate their story.

Give a tip

Sign up for DevGenius Updates

By Dev Genius

Get the latest news and update from DevGenius publication [Take a look.](#)

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.



Get this newsletter

[About](#) [Help](#) [Terms](#) [Privacy](#)

Get the Medium app

