Refactoring 101

By:
Adam Culp
Twitter: @adamculp

https://joind.in/**14927**

- **About me**

    – PHP 5.3 Certified

    – Consultant at Zend Technologies

    – Organizer SoFloPHP (South Florida)

    – Organized SunshinePHP (Miami)

    – Long distance (ultra) runner

    – Judo Black Belt Instructor

Refactoring 101

- **Fan of iteration**
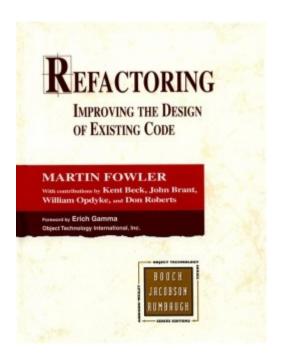 - Pretty much everything requires iteration to do well:
    - Long distance running
    - Judo
    - Development
    - Evading project managers
    - Refactoring!

Refactoring
Improving the Design of Existing Code

pnwphp

Refactoring 101

- **About talk**

  – Based on "Refactoring; Improving The Design of Existing Code" book, by Martin Fowler.

  – https://github.com/adamculp/refactoring101 – for PHP code samples
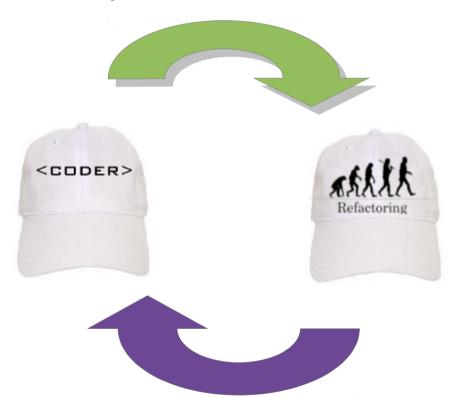
Refactoring 101

- ## What is "refactoring"?

  - "...process of changing a computer program's source code without modifying its external functional behavior..." *en.wikipedia.org/wiki/Refactoring*

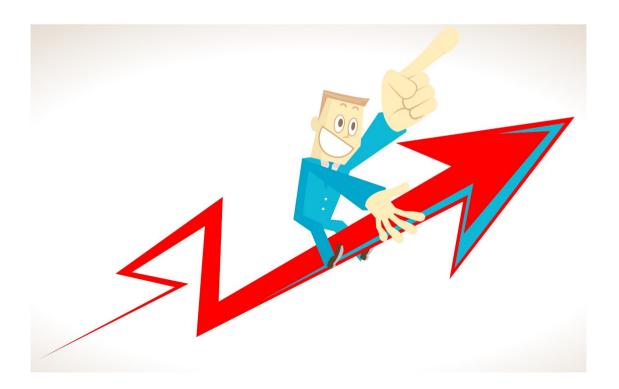 - Should not add functionality

  - Simplify code

  - Improve code readability

pnwphp

- **Two hats**

  - Adding Functionality Hat

  - Refactoring Hat

  - We add functionality, then refactor, then add more functionality …

Refactoring 101

- ## Then optimize

  - Do not optimize while refactoring

  - Separate step

  - Refactoring is NOT optimizing

pnwphp

Refactoring 101

- **Why refactor?**

  - Prevent decay

  - Preserve or fix design

  - Reduce duplication

  - Improve maintainability

  - Helps us code faster

  - Locate bugs

  - Code smells

pnwphp

- **Code "smells"**
  - What are "smells"?
    - Indications of spoiled code nearby
    - Not conclusive
    - The "smell" is not bad

Refactoring 101

- **Code "smells"**
  - "Smells" hinting a refactor may be needed:
    - Duplicate Code (rule of 3)
    - Long Methods
    - Large Class
    - Long Parameter (argument) List
    - Divergent Change – cascade change to accommodate another
    - Shotgun Surgery – change ripples as bugs
    - Feature Envy – method uses parts from other class
    - Switch Statements – sacrifice polymorphism

pnwphp

Refactoring 101

- **Code "smells"**
  - Cont'd:
    - Lazy Class – class not doing much
    - Speculative Generality – something built for possible future
    - Temporary Field/Variable
    - Message Chains – object asking object asking object
    - Middle Man – directors in place but serve no real purpose
    - Inappropriate Intimacy – classes share private parts
    - Data Class – getters and setters, but nothing else
    - Comments – where comments cover bad code

pnwphp
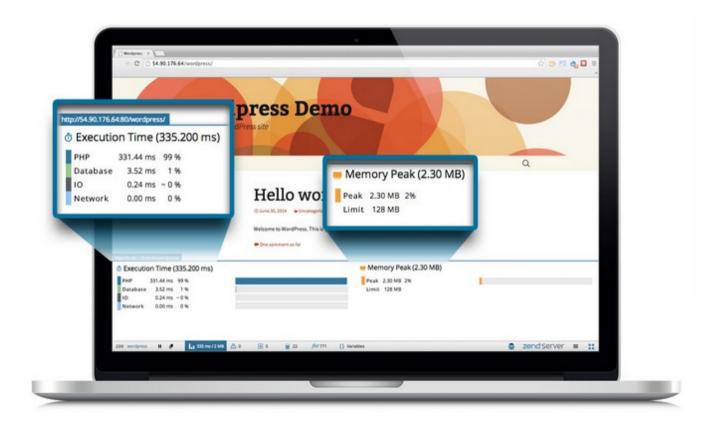
- **Tools to highlight smells**
  - PHPqatools.org
    - PHPUnit
    - PHPLoc
    - PHP_Codesniffer
    - PHP_Depend
    - PHP Copy/Paste Detector
    - PHP Mess Detector
    - PHP Dead Code Detector

pnwphp

Refactoring 101

- ## Realtime profiling
  - Zend Z-Ray

Refactoring 101

- **Rewrite vs Refactor**
  - Rewrite = perceived easy road
  - Refactor = best teacher
  - Business arguments
    - Budget
    - Time
    - Retain business logic

pnwphp

Refactoring 101

- ## When to rewrite

    - Want a new app

        - Not just better coded current app

    - Business logic change

    - Target market change

    - Framework integration or change

pnwphp

Refactoring 101

- ## When to refactor?

  - No "special" time

  - Short bursts

  - Refactor to gain something

  - Prior to adding functionality

  - When fixing a bug

  - During code review

pnwphp

Refactoring 101
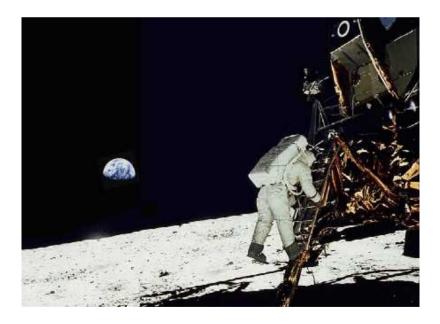
- **What do I tell my manager? (justification)**
    - **Tech savvy** manager = not be hard to explain the benefits.
    - **Quality centric** manager = stress quality aspects.
        - Introduce as a review process.
        - Many resources on Google.
    - **Schedule driven** manager = Don't tell (controversial?).
        - Find a way to work it in.
        - Overall it saves time, but some will never "see" it.

pnwphp

Refactoring 101

- **First steps**
  - Use source control (Git, SVN, etc.)
    - Records steps, provides rollback
    - Auditable
  - GET IT WORKING
    - **Do NOT refactor broken**
  - Create consistent data
  - Create tests

pnwphp

- **Tests and refactoring**
  - Basic refactor steps
    - Ensure tests pass
    - Plan refactor
    - Implement
    - Ensure tests still pass
      - Updating tests if needed
      - Add more tests to cover newly discovered items
    - Repeat!

PHPUnit

pnwphp

Refactoring 101

- **Example**

  - Lets look at a code example.

  - Tips and descriptions during steps.

  - **Our Task:**

    - Video Rental Company has asked for an HTML representation of their customer statement to be created.

Let's look at the code!

```
66
67 class Movie {
68     const CHILDRENS = 2;
69     const REGULAR = 0;
70     const NEW_RELEASE = 1;
71
72     public $title;
73     public $priceCode;
74
75     public function __construct($title, $priceCode) {
76         $this->title = $title;
77         $this->setPriceCode($priceCode);
78     }
79
80     public function getPriceCode() {
81         return $this->priceCode;
82     }
83
84     public function setPriceCode($priceCode) {
85         $this->priceCode = $priceCode;
86     }
87
88     public function getTitle() {
89         return $this->title;
90     }
91 }
92
```

23

pnwphp

```php
92
93 class Rental {
94     public $movie;
95     public $daysRented;
96
97     public function __construct(Movie $movie, $daysRented) {
98         $this->movie = $movie;
99         $this->daysRented = $daysRented;
100     }
101
102     public function getDaysRented() {
103         return $this->daysRented;
104     }
105
106     public function getMovie() {
107         return $this->movie;
108     }
109 }
110
```

pnwphp

```
2
3 class Customer {
4     public $name;
5     public $rentals;
6
7     public function __construct($name) {
8         $this->name = $name;
9     }
10
11     public function addRental(Rental $rental) {
12         $this->rentals[] = $rental;
13     }
14
15     public function getName() {
16         return $this->name;
17     }
18
19     //...
20
```

```php
 2
 3 class Customer...
 4     public function statement() {
 5         $totalAmount = 0;
 6         $frequentRenterPoints = 0;
 7         $result = "Rental Record for " . $this->getName() . "\n";
 8
 9         foreach ($this->rentals as $each) {
10
11             $thisAmount = 0;
12
13             switch ($each->movie->getPriceCode()) {
14                 case Movie::REGULAR:
15                     $thisAmount += 2;
16                     if ($each->getDaysRented() > 2) {
17                         $thisAmount += ($each->getDaysRented() - 2) * 1.5;
18                     }
19                     break;
20
21                 case Movie::NEW_RELEASE:
22                     $thisAmount += $each->getDaysRented() * 3;
23                     break;
24
25                 case Movie::CHILDRENS:
26                     $thisAmount += 1.5;
27                     if ($each->getDaysRented() > 3) {
28                         $thisAmount += ($each->getDaysRented() - 3) * 1.5;
29                     }
30                     break;
31             }
32
33             //...
34
```

pnwphp

```
47
48            //...
49
50            $frequentRenterPoints++;
51
52            // add bonus for a two day release rental
53            if (($each->movie->getPriceCode() == Movie::NEW_RELEASE) &&
54                ($each->getDaysRented() > 1)) {
55                $frequentRenterPoints++;
56            }
57
58            $result .= "\t" . $each->movie->getTitle() . "\t" . $thisAmount . "\n";
59            $totalAmount += $thisAmount;
60        }
61
62        // add footer lines
63        $result .= "Amount owed is " . $totalAmount . "\n";
64        $result .= "You earned " . $frequentRenterPoints . " frequent renter points";
65
66        return $result;
67    }
68 }
69
```

pnwphp

```
117
118 // define customer
119 $customer = new Customer('Adam Culp');
120
121 // choose movie to be rented, define rental, add it to the customer
122 $movie = new Movie('Gladiator', 0);
123 $rental = new Rental($movie, 1);
124 $customer->addRental($rental);
125
126 // choose 2nd movie to be rented, define rental, add it to the customer
127 $movie = new Movie('Spiderman', 1);
128 $rental = new Rental($movie, 2);
129 $customer->addRental($rental);
130
131 // print the statement
132 echo $customer->statement();
133
```

pnwphp

- **Code summary: What did we see?**
  - Method → statement()
    - Too long
    - Not reusable for HTML version
    - Switch sacrificing polymorphism
    - Determining class/type
    - Calculating rental price, frequent renter points, grant total

Refactoring 101

- **Additional notes**

  - Cannot change how movies are classified.

  - Customers always changes, not easy in current state.

    - Movie classification
    - Frequent renter points
    - Rental days per type
    - Price calculation

- **Objective:**
  - Clean up statement().
    - Shorten
      - Extract code to encapsulate functionality
      - Extract business logic to keep DRY

- TEST ! ! !

- **Extract method**
  - Moves a functionality to it's own method.
    - Encapsulate calculation of each rental.
    - Shorten statement() method.

```php
foreach ($this->rentals as $each) {

    $thisAmount = 0;

    switch ($each->movie->getPriceCode()) {
        case Movie::REGULAR:
            $thisAmount += 2;
            if ($each->getDaysRented() > 2) {
                $thisAmount += ($each->getDaysRented() - 2) * 1.5;
            }
            break;

        case Movie::NEW_RELEASE:
            $thisAmount += $each->getDaysRented() * 3;
            break;

        case Movie::CHILDRENS:
            $thisAmount += 1.5;
            if ($each->getDaysRented() > 3) {
                $thisAmount += ($each->getDaysRented() - 3) * 1.5;
            }
            break;
    }

    $frequentRenterPoints++;
```

```php
15  public function addRental(Rental $rental) {
16      $this->rentals[] = $rental;
17  }
18
19  public function getName() {
20      return $this->name;
21  }
22
23  public function statement() {
24      $totalAmount = 0;
25      $frequentRenterPoints = 0;
26      $result = "Rental Record for " . $this->get
27
28      foreach ($this->rentals as $each) {
29
30          $thisAmount = $this->amountFor($each);
31
32          $frequentRenterPoints++;
33
34          // add bonus for a two day release rent
35          if (($each->movie->getPriceCode() == Mo
36              $frequentRenterPoints++;
37          }
38
39          $result .= "\t" . $each->movie->getTitl
40          $totalAmount += $thisAmount;
41      }
```

hp

- **Extract method cont'd.**
  - We now have a new method amountFor().

```
47          return $result;
48      }
49
50      public function amountFor($each) {
51          $thisAmount = 0;
52
53          switch ($each->movie->getPriceCode()) {
54              case Movie::REGULAR:
55                  $thisAmount += 2;
56                  if ($each->getDaysRented() > 2) {
57                      $thisAmount += ($each->getDaysRented() - 2) * 1.5;
58                  }
59                  break;
60
61              case Movie::NEW_RELEASE:
62                  $thisAmount += $each->getDaysRented() * 3;
63                  break;
64
65              case Movie::CHILDRENS:
66                  $thisAmount += 1.5;
67                  if ($each->getDaysRented() > 3) {
68                      $thisAmount += ($each->getDaysRented() - 3) * 1.5;
69                  }
70                  break;
71          }
72
73          return $thisAmount;
74      }
```

pnwphp

Refactoring 101

- **Rename variables**
  - Renaming $each to $rental
  - Improves readability.
  - Relate intent.

```
30
foreach ($this->rentals as $each) {        → ←  31        foreach ($this->rentals as $rental) {
                                                32
    $thisAmount = $this->amountFor($each);  → ←  33            $thisAmount = $this->amountFor($rental);
                                                34
    $frequentRenterPoints++;                      35            $frequentRenterPoints++;
                                                36
    // add bonus for a two day release rental     37            // add bonus for a two day release rental
    if (($each->movie->getPriceCode() == Movie::NEW_RELEASE → ←  38    if (($rental->movie->getPriceCode() == Movi
        $frequentRenterPoints++;                  39                $frequentRenterPoints++;
    }                                             40            }
                                                41
    $result .= "\t" . $each->movie->getTitle() . "\t" . $th → ←  42        $result .= "\t" . $rental->movie->getTitle(
    $totalAmount += $thisAmount;                  43            $totalAmount += $thisAmount;
}                                                 44        }
                                                45
```

pnwphp

- **Renaming variables, cont'd.**
  - Renamed $each to $rental, and also changed $thisAmount to become $result for clarity.

```php
                                                    52
public function amountFor($each) {          → ←    53    public function amountFor($rental) {
    $thisAmount = 0;                                54        $result = 0;
                                                    55
    switch ($each->movie->getPriceCode()) {  → ←   56        switch ($rental->movie->getPriceCode())
        case Movie::REGULAR:                        57            case Movie::REGULAR:
            $thisAmount += 2;                → ←    58                $result += 2;
            if ($each->getDaysRented() > 2) {       59                if ($rental->getDaysRented() > 2
                $thisAmount += ($each->getDaysRented() - 2) * 1    60                    $result += ($rental->getDays
            }                                       61                }
            break;                                  62                break;
                                                    63
        case Movie::NEW_RELEASE:             —     64            case Movie::NEW_RELEASE:
            $thisAmount += $each->getDaysRented() * 3;  → ←  65                $result += $rental->getDaysRente
            break;                                  66                break;
                                                    67
        case Movie::CHILDRENS:                      68            case Movie::CHILDRENS:
            $thisAmount += 1.5;              → ←    69                $result += 1.5;
            if ($each->getDaysRented() > 3) {       70                if ($rental->getDaysRented() > 3
                $thisAmount += ($each->getDaysRented() - 3) * 1    71                    $result += ($rental->getDays
            }                                       72                }
            break;                                  73                break;
    }                                               74        }
                                                    75
    return $thisAmount;                      → ←   76        return $result;
}                                                   77    }
```

pnwphp

- **Rename method**
  - Rename amountFor() to getCharge().
  - Self documenting.

```
     $thisAmount = $this->amountFor($rental);           32
                                              → ←  33    $thisAmount = $this->getCharge($rental);
                                                    34
     $frequentRenterPoints++;                       35    $frequentRenterPoints++;
                                                    36
     // add bonus for a two day release rental      37    // add bonus for a two day release rental
     if (($rental->movie->getPriceCode() == Movie::NEW_RELEA 38    if (($rental->movie->getPriceCode() == Mov
         $frequentRenterPoints++;                   39        $frequentRenterPoints++;
     }                                              40    }
                                                    41
     $result .= "\t" . $rental->movie->getTitle() . "\t" . $ 42    $result .= "\t" . $rental->movie->getTitle
     $totalAmount += $thisAmount;                   43    $totalAmount += $thisAmount;
   }                                                44  }
                                                    45
   // add footer lines                              46  // add footer lines
   $result .= "Amount owed is " . $totalAmount . "\n";  47  $result .= "Amount owed is " . $totalAmount .
   $result .= "You earned " . $frequentRenterPoints . " freque 48  $result .= "You earned " . $frequentRenterPoir
                                                    49
   return $result;                                  50  return $result;
}                                                   51  }
                                                    52
public function amountFor($rental) {           → ←  53  public function getCharge($rental) {
   $result = 0;                                     54    $result = 0;
                                                    55
```

pnwphp

- **Move method**

  - Move getCharge() from Customer to Rental.

    - Relies on Rental data.

  - Already have Rental object, no need to pass $rental.

```
foreach ($this->rentals as $rental) {

    $thisAmount = $this->getCharge($rental);

    $frequentRenterPoints++;
```

	31
—	32
→ ←	33
	34
	35
	36

```
foreach ($this->rentals as $rental) {

    $thisAmount = $rental->getCharge();

    $frequentRenterPoints++;
```

pnwphp

- ## Move method cont'd

  - Now calls getDaysRented() directly.

  - Returns charge of Rental, as it should.

    - Building rental charge in customer was misplaced.

```
 94        return $this->movie;
←95    }
 96
 97    public function getCharge() {
 98        $result = 0;
 99
100        switch ($this->movie->getPriceCode()) {
101            case Movie::REGULAR:
102                $result += 2;
103                if ($this->getDaysRented() > 2) {
104                    $result += ($this->getDaysRented() - 2) * 1.5;
105                }
106                break;
107
108            case Movie::NEW_RELEASE:
109                $result += $this->getDaysRented() * 3;
110                break;
111
112            case Movie::CHILDRENS:
113                $result += 1.5;
114                if ($this->getDaysRented() > 3) {
115                    $result += ($this->getDaysRented() - 3) * 1.5;
116                }
117                break;
118        }
119
120        return $result;
121    }
```

```
aysRented) {


it to the customer


add it to the cust
```

pnwphp

Refactoring 101

- **Replace temp with query**
  - Remove temporary variable and call Rental->getCharge() direct.
    - Less future maintenance.
    - Makes code clearer.

- **Extract method**
  - $frequentRenterPoints calculation extracted to getFrequentRenterPoints(), **and** move it in the Rental class.

```php
foreach ($this->rentals as $rental) {

    $frequentRenterPoints++;

    // add bonus for a two day release rental
    if (($rental->movie->getPriceCode() == Movie::NEW_RELEASE) &&
        $frequentRenterPoints++;
    }

    $result .= "\t" . $rental->movie->getTitle() . "\t" . $rental·
```

```php
25      $frequentRenterPoints = 0;
26      $result = "Rental Record for " . $this->getName() . "\n";
27
28      foreach ($this->rentals as $rental) {
29          $frequentRenterPoints += $rental->getFrequentRenterPoints();
30
31          $result .= "\t" . $rental->movie->getTitle() . "\t" . $rental
32          $totalAmount += $rental->getCharge();
33      }
34
```

```php
107             }
108
109             return $result;
110         }
111
112     public function getFrequentRenterPoints() {
113             // add bonus for a two day release rental
114             if (($this->movie->getPriceCode() == Movie::NEW_RELEASE) && ($thi
115                 return 2;
116             } else {
117                 return 1;
118             }
119         }
120 }
121
```

- **Replace temp with query**

  - Encapsulate logic and generation of grand total.

  - Promotes DRY.

  - Remove $totalAmount temporary variable.

```php
public function statement() {
    $totalAmount = 0;
    $frequentRenterPoints = 0;
    $result = "Rental Record for " . $this->getName() . "\n";

    foreach ($this->rentals as $rental) {
        $frequentRenterPoints += $rental->getFrequentRenterPoints();

        $result .= "\t" . $rental->movie->getTitle() . "\t" . $rental·
        $totalAmount += $rental->getCharge();
    }

    // add footer lines
    $result .= "Amount owed is " . $totalAmount . "\n";
    $result .= "You earned " . $frequentRenterPoints . " frequent ren

    return $result;
}

ss Movie {
 const CHILDRENS = 2;
 const REGULAR = 0;
 const NEW_RELEASE = 1;

public $title;
```

```php
23    }
24
25    public function statement() {
26        $frequentRenterPoints = 0;
27        $result = "Rental Record for " . $this->getName() . "\n";
28
29        foreach ($this->rentals as $rental) {
30            $frequentRenterPoints += $rental->getFrequentRenterPoint
31
32            $result .= "\t" . $rental->movie->getTitle() . "\t" . $r
33        }
34
35        // add footer lines
36        $result .= "Amount owed is " . $this->getTotalCharge() . "\n
37        $result .= "You earned " . $frequentRenterPoints . " frequer
38
39        return $result;
40    }
41
42    public function getTotalCharge() {
43        $result = 0;
44
45        foreach ($this->rentals as $rental) {
46            $result += $rental->getCharge();
47        }
48
```

pnwphp

Refactoring 101

- ## Replace temp with query

  - Encapsulate logic and generation of frequent renter points.

  - Promotes DRY.

  - Remove $frequentRentalPoints temporary variable.

- ## Create HTML statement

  – Create HTML version.

  – Rename original as text version.

- **Execution**
  - Can call either Text or HTML versions.

```
152 $customer->addRental($rental);
153
154 // print the statement
155 echo $customer->statement();
156
```

```
170
171 // print the statement
172 echo $customer->statementText();
173 echo $customer->statementHtml();
174
```

Refactoring 101

- **Recap**
    - Most refactoring reduces code
        - More self-documenting
        - More flexibility
        - More testable
    - 3 loops (getFrequentRenterPoints, getTotalCharge, and statement)
        - Isolates calculations
        - Enabled multiple statements (text/html)
    - Optimizing and refactoring = different
        - Refactor, then optimize
    - Future = separation of concerns

pnwphp

- **Conclusion**
    - Do not refactor a broken application
    - **Always** have tests in place prior to refactor
        - Unit tests or
        - Functional tests or
        - Manual tests
    - Leave code cleaner than you got it
    - Try NOT to rewrite
    - Learn to "smell" problems
    - Love iteration!

pnwphp

- **Thank you!**

  - Code: https://github.com/adamculp/refactoring101

  - Please rate at: https://joind.in/**14927**

Adam Culp

http://www.geekyboy.com

http://RunGeekRadio.com

Twitter @adamculp

pnwphp