Clean Application Development

Adam Culp @adamculp

Clean Application Development

- **About me**
  - OSS Contributor
  - PHP Certified
  - Zend Certification Advisory Board
  - PHP-Fig voting member (IBM i Toolkit)
  - Consultant at Zend Technologies
  - Organizer SoFloPHP (South Florida)
  - Organizer SunshinePHP (Miami)
  - Long distance (ultra) runner
  - Photography Enthusiast
    - Judo Black Belt Instructor

SoFloPHP Users Group
WWW.SOFLOPHP.COM

php

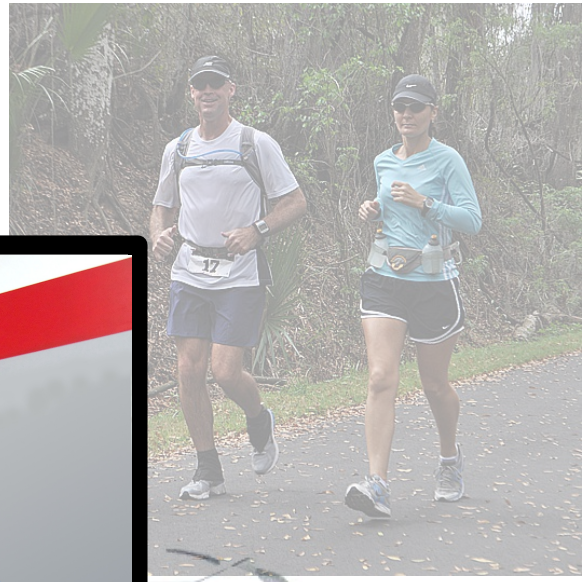
zend CERTIFIED ENGINEER PHP 5.3

zend CERTIFICATION ADVISORY BOARD

SUNSHINEPHP CONFERENCE

Clean Application Development

- About me
  - OSS Contributor
  - PHP Certified
  - Zend Certification Advisory Board
  - PHP-Fig voting r
  - Consultant at Ze
  - Organizer SoFlo
  - Organizer Sunsh
  - Long distance (u
  - Photography Ent
  - Judo Black Belt I



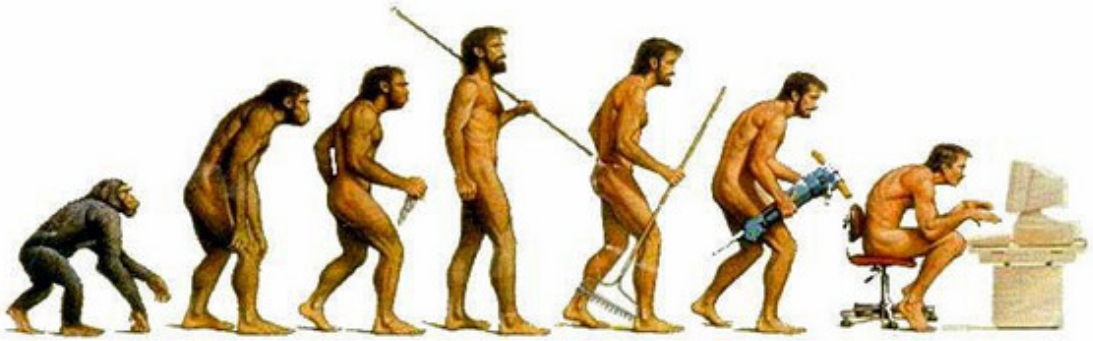
I am the PHP Ninja!!!

SoFloPHP
WWW.SOFLOPHP.COM

SUNSHINEPHP CONFERENCE

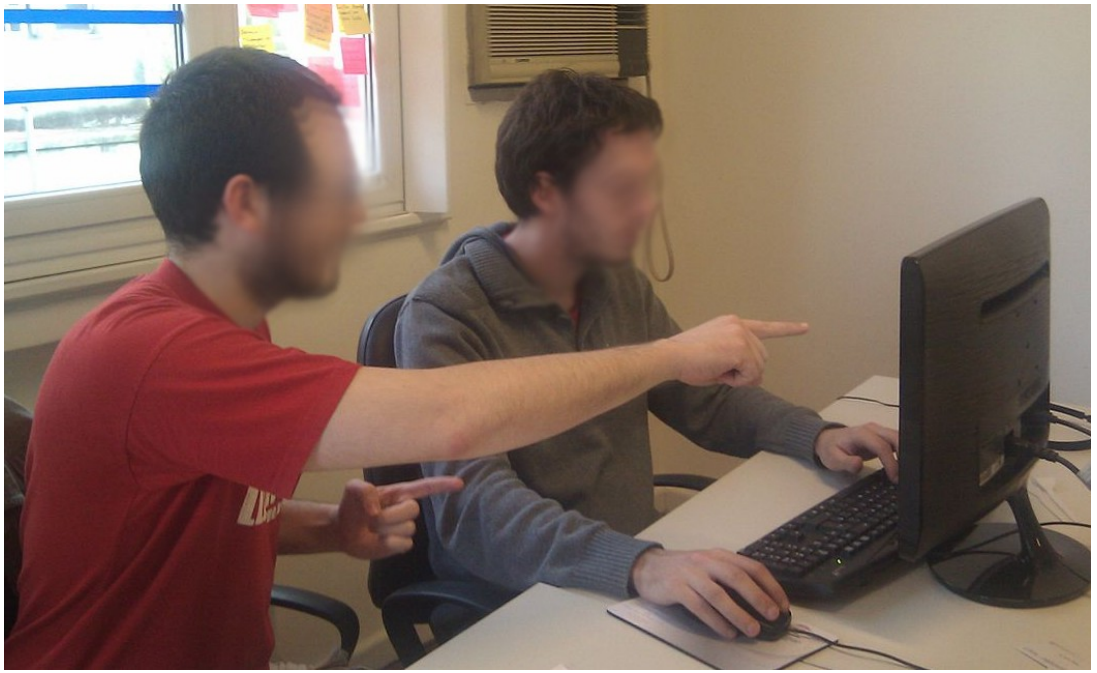
Clean Application Development – Iteration

- **I love iteration!**
  - Evolution.
  - Learning to walk.
  - Training to run.
  - Evading project managers.
  - Clean development.



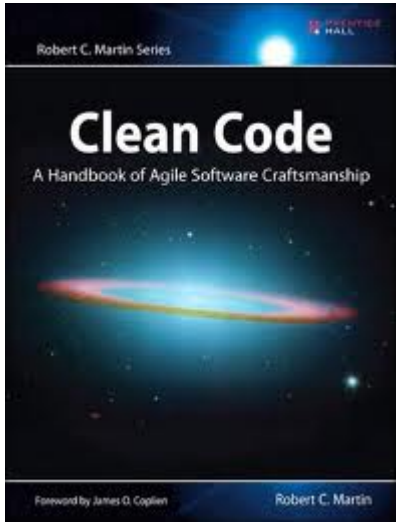
Clean Application Development – Learning

- **Clean application development cannot be taught in 45 minutes**
  - Practice, Practice, Practice.
  - Leave the code better than you found it.
  - Always watch for new techniques.
  - No “silver bullet“.



Clean Application Development – Resources

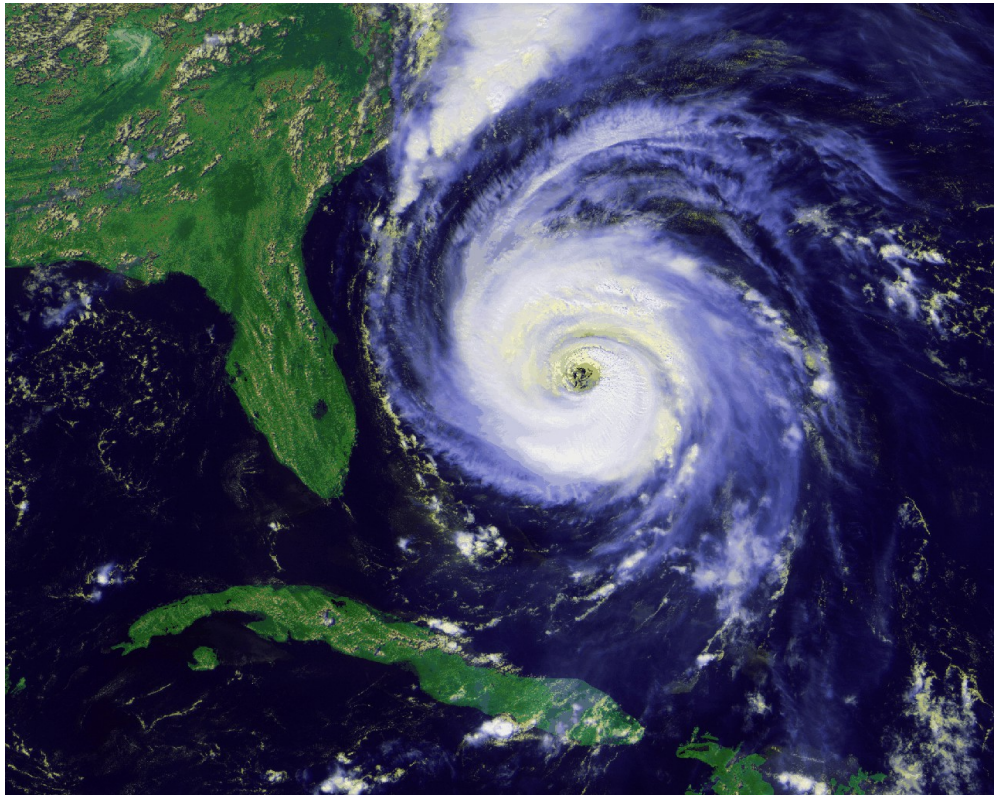
- **A great resource on code quality.**



Clean Code
By Robert C. Martin

Clean Application Development – Causes of bad...

- **Disasters happen, resulting in bad...**
  - It's easy
  - Short deadlines
  - Boss
  - Laziness
  - Lack of “how-to“
  - Come back and clean later
  - **We are the cause!**



Clean Application Development – Typical Scenario

- **Bad things starts innocently**
- **Hire a new professional/developer**
  - Clean slate, no expectations
- **Start a new project**
  - Initial output is quick and easy, setting expectation
  - Slows down over time
  - Complexity increases
  - Domino effect

Clean Application Development – Defend the code

- **Our responsibility to say “NO“**
  - Managers job = defend schedule and features
  - Our job = defend the code
  - Managers respect realistic reasons and explanations

Clean Application Development – We are judged

- **Others judge us on our code**
  - We are @authors.
  - 75% of time reading code
    - Others read our code also
  - And they talk about it!
  - How developers talk about us = “CRED“

And they talk about it!

Clean Application Development – Result of bad code

- **Side-effects of bad code:**
  - Wasted Time
  - Bugs
  - Excessive debugging
  - Procrastination
  - Missed deadlines
  - Technical debt
  - Financial losses

Clean Application Development – The aftermath

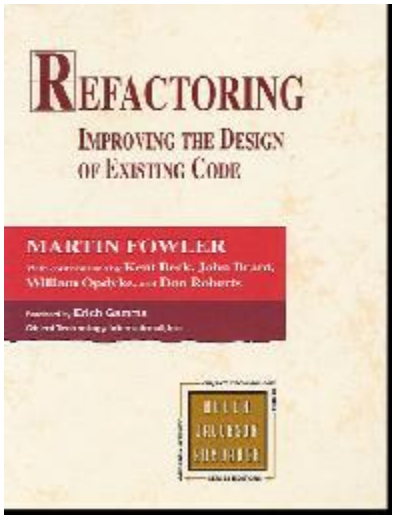
- **How we typically react to a dirty application:**
  - Padded estimates
  - Developers Hide
  - Become defensive
  - Blame others/requirements
  - Add developers
  - Rewrite!

Clean Application Development – Rewrite problems

- **The problem with a rewrite:**
  - Development team split, old/new
  - Legacy application enhanced
  - New application Scope creep
  - Is it done yet?
  - **Ends with more bad code!**

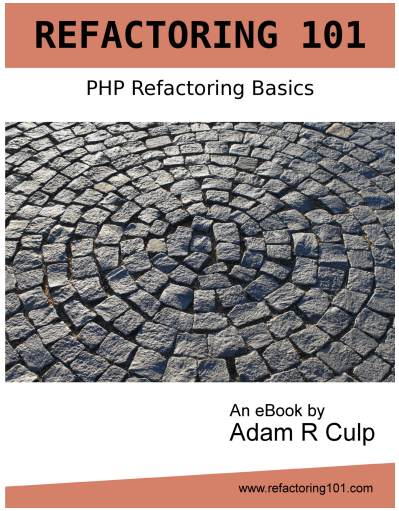
Clean Application Development – Resources

- **Learn to Refactor.**



Refactoring
By Martin Fowler

https://github.com/adamculp/refactoring101



Refactoring 101
By Adam Culp

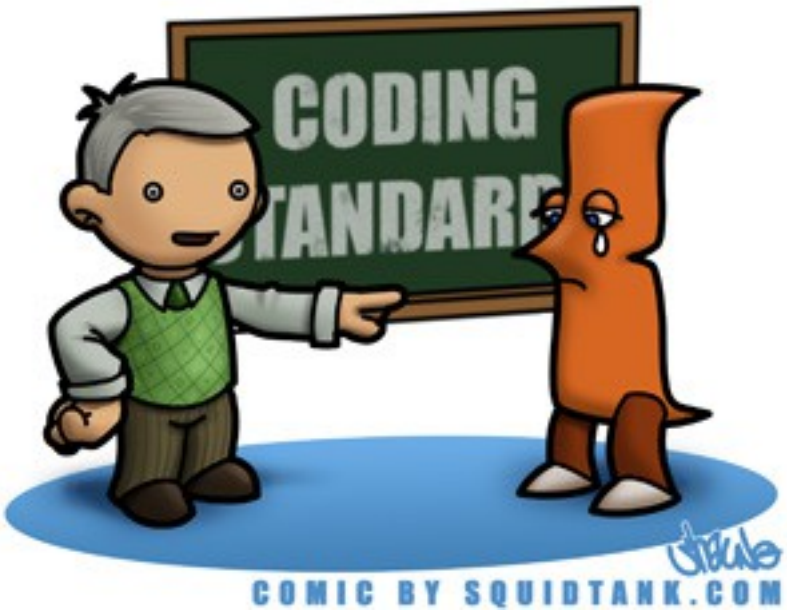
https://refactoring101.com

Clean Application Development – Common sense

- **With all of these problems, clean applications makes sense**
  - Shortens development time.
  - On-boarding of developers easier.
  - Less bugs.
  - Happier end-users.
  - Predictable schedules.
  - It's the professional thing to do.

Clean Application Development – Coding standards

- **Coding Standards save time**
  - Gives direction
  - PHP Framework Interoperability Group (https://www.php-fig.org).
  - Standard NOT important
    - Unless it‘s public code
    - Choose one
    - Stick with it
  - **Consistency** is key



Clean Application Development – Clear names

- **Names should be clear**
  - Functions and variables should tell a story.

Bad

```
$elapsed;
$createdDays;
$modifiedDays;
$age;
```

Good

```
$elapsedTimeInDays;
$daysSinceCreation;
$daysSinceModified;
$fileAgeInDays;
```

Clean Application Development – No confusion

- **Shy away from variations of the same name**
  - To ensure names are different enough to portray difference.

What is the difference between these?

$product
$productInfo
$productData
$productDescription

Clean Application Development – Bad characters

- **Certain characters are hard to understand**

Bad

Lower case L
Uppercase O (oh)
Uppercase I (eye)

Clean Application Development – Name grammar

- **Technical names help developers who actually read the code.**
- **Non-technical terms for client documentation.**
- **Class names should be nouns**
  - Describe.
  - Ex. - Customer, Account, Product, Company.
- **Method names should be verbs**
  - Action.
  - Ex. - getCustomer, closeAccount, updateProduct, addCompany.
  - Pick a set of keywords and stick with them.
    - Ex. - fetch, get, add, update, remove, delete

Clean Application Development – Clean code

- **More on Clean Code**
  - Functions:
    - Short
    - Single purpose
    - As expected
    - Well named
  - Recognizing bad doesn't mean we know how to make good
    - We know a good/bad song, but are not song writers
  - Clean code = caring developer
  - Does not require many comments

Clean Application Development – Code comments

- **Comments can also be a bad “smell“**
  - Comments are often used to cover up bad code.
  - Code should be self-documenting

```php
<?php

// Check to see if the employee is eligible for full benefits
if (($employee['flags'] & HOURLY_FLAG) && ($employee['age'] > 65)) {
    /* Do something */
}

// Or this?

if ($this->Employee->isEligibleForFullBenefits()) {
    /* Do something */
}
```

Clean Application Development – Smells of bad code

- **How to spot bad code (smells)**
  - Incomplete error handling
  - Memory leaks
  - Race conditions
  - Inconsistent naming convention (class, function, variable)
  - Inconsistent coding standard
  - Un-initialized variables
  - Code lacks clear purpose
  - Functions do too much (more than one thing)
  - Globals used
  - Too many comments in the code
  - Notices, Warnings, Fatal Errors

Clean Application Development – Code sniffer

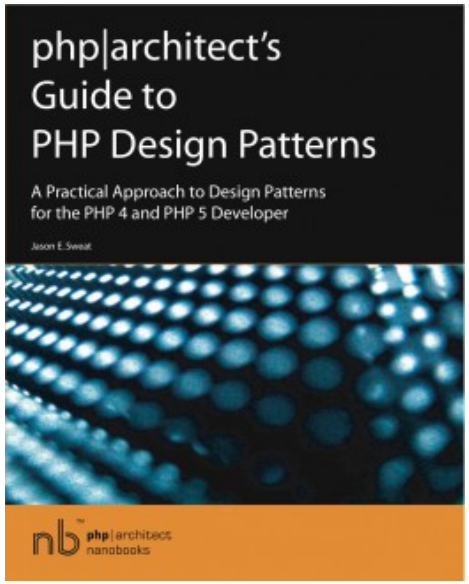
- **Let PHP CodeSniffer detect bad smells**
  - Set rules to detect if coding standard is not followed
  - Run during commits in version control
  - IDE integration

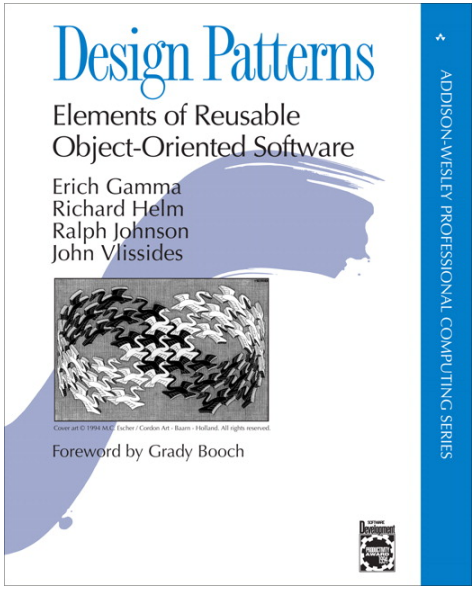
Clean Application Development – Peer code reviews

- **Peer code review great learning tool**
  - Peers help train each other on strong points.
  - Fresh set of eyes.
  - Builds better teams.

Clean Application Development – Design Patterns

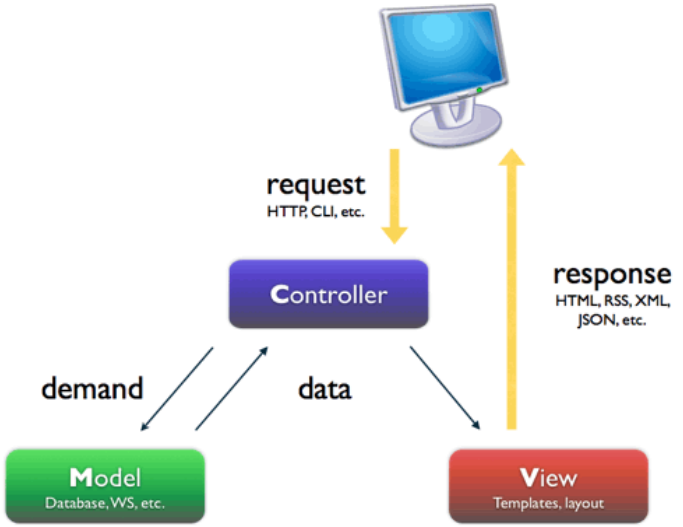
- **Standard and quick solutions to common coding problems**
  - Provide standard ways of dealing with common code problems.
  - “Guide to PHP Design Patterns“ by Jason Sweat.
  - “Design Patterns, Elements of Reusable Object-Oriented Software“ by Gang of four

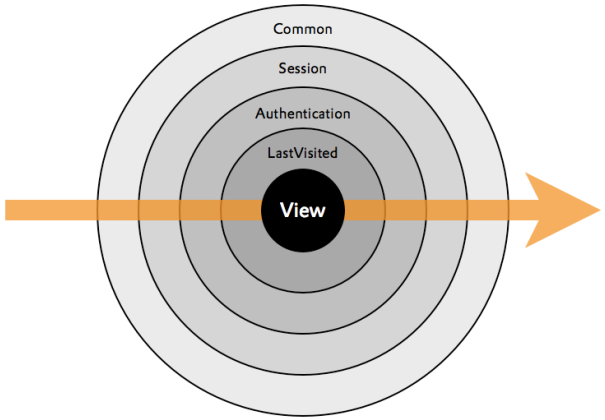




Clean Application Development – Frameworks

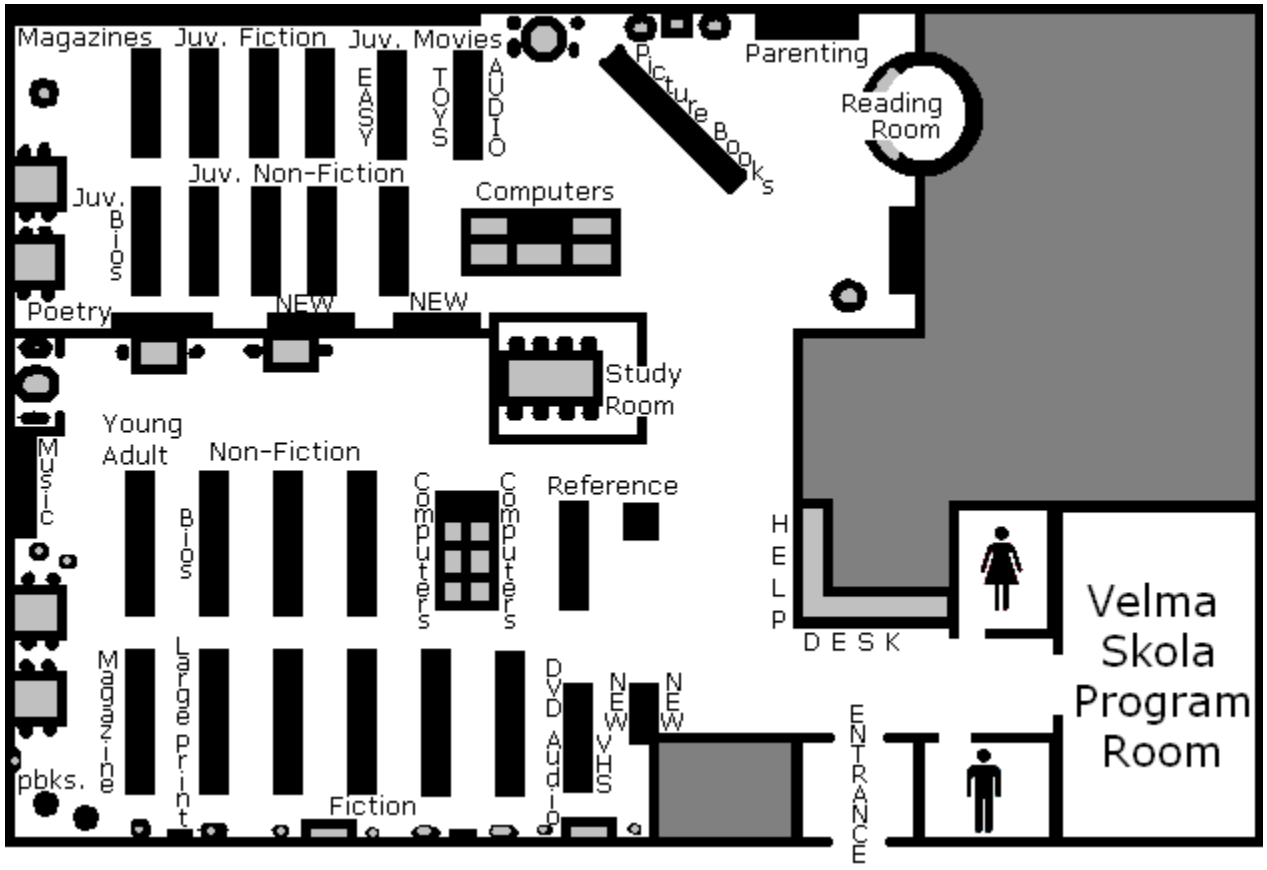
- **Frameworks help keep things in line**
  - $evil = 'roll-your-own'
    - Knowledge transfer
    - Onboarding
    - Insecure
  - Allows our code to be lighter, simpler
  - Does heavy lifting
  - Most modern frameworks are:
    - MVC
    - Service driven
    - Middleware
  - STICK TO IT!!!





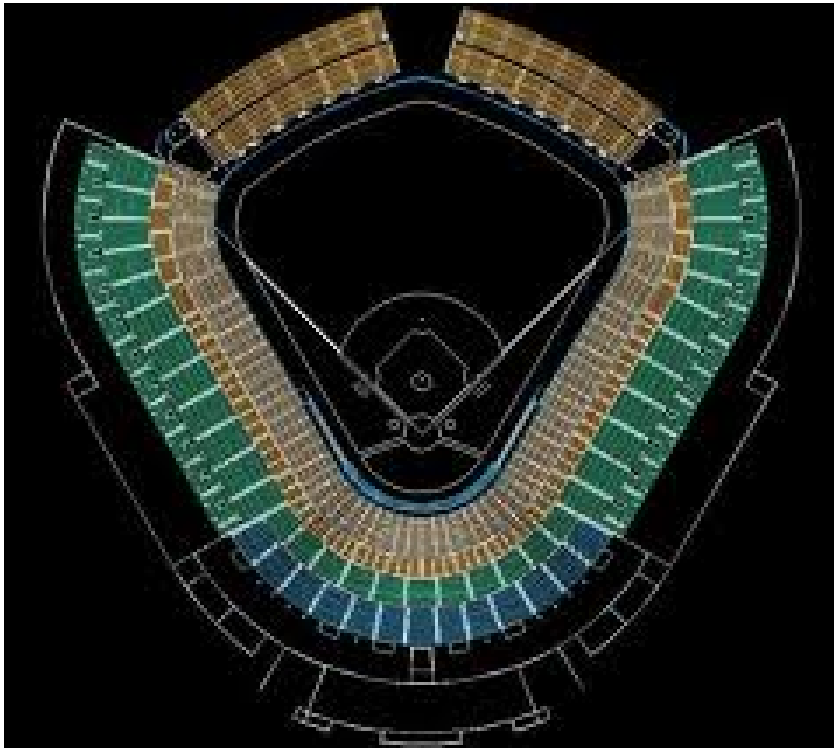
Clean Application Development – Clear architecture

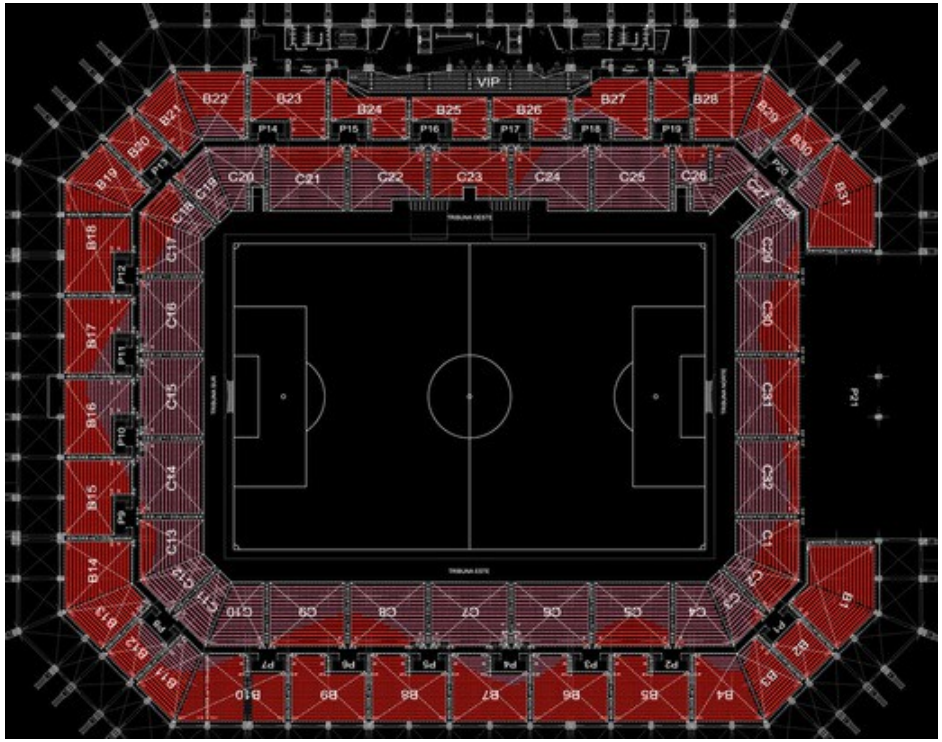
- We can tell pretty simply this “looks“ like a library. (bookshelves, computers, book categories)



Clean Application Development – Obvious purpose

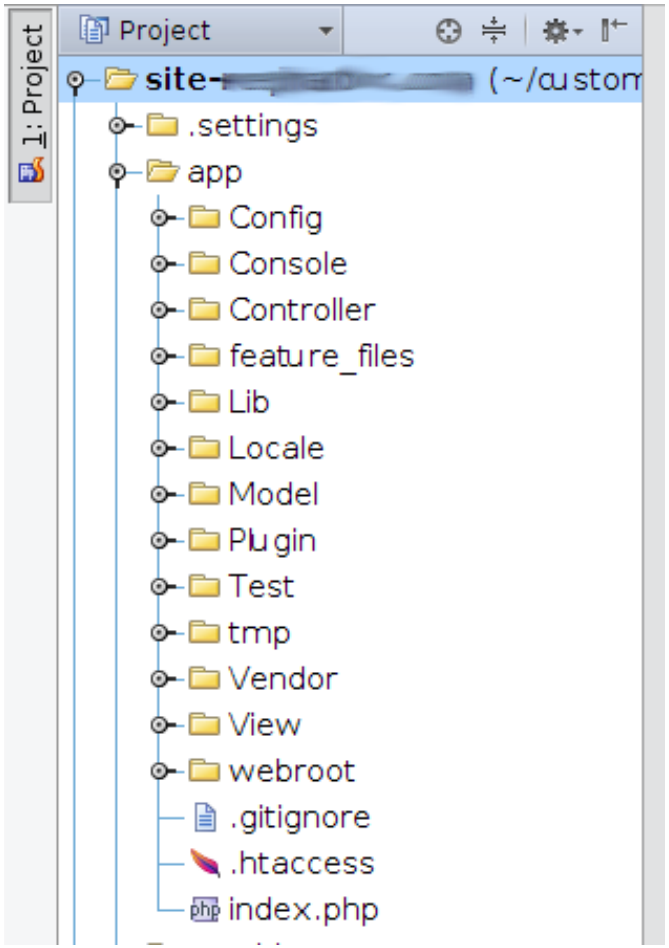
- These are pretty obvious without any explanation.





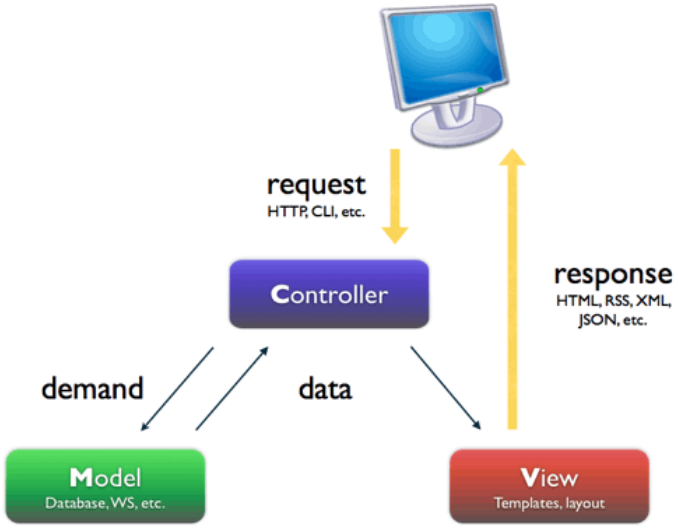
Clean Application Development – MVC architecture?

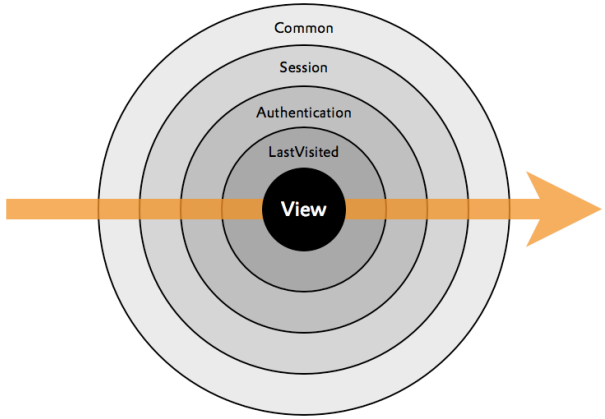
- **This would take a bit more digging to figure out.**



Clean Application Development – Evolution

- **Framework evolution**
  - Component Libaries
  - Full (kitchen sink) Stack
  - MVC
  - Micro
  - Action → Response
  - Component-ized
  - Middleware

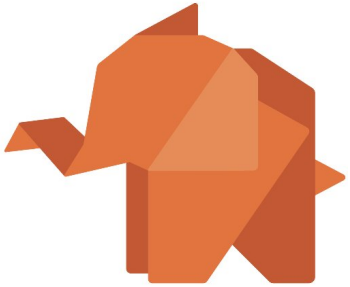




Clean Application Development – Dependencies

- **Composer and Packagist**
  - Prevents NIH
  - Standardized autoloading
  - Easier upgrades
  - Promotes OSS
  - Faster onboarding
  - Public and Private code

OR



Clean Application Development – Testing

- **Unit testing = parachute**
  - Each test = one thing
  - Ensures system functioning
  - Makes refactoring easier
  - The “Way of Testivus“


```php
/**
 *  Test if the Application is owned by specific user
 */
public function testIsOwnedBy()
{
    $result = $this->Application->isOwnedBy($this->id, $this->user_id);

    $this->assertTrue($result);
}
```

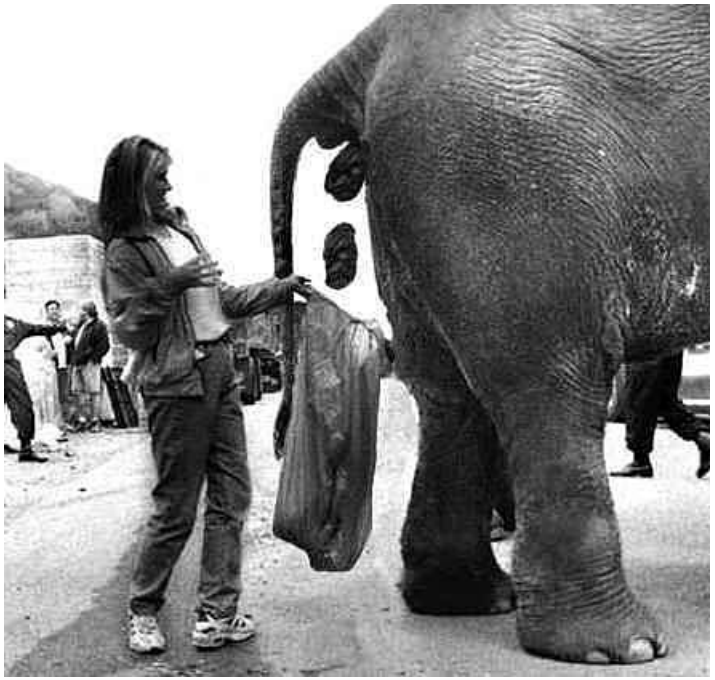
Clean Application Development – TDD

- **Important things should be done first**
  - Write failing tests first, then write the code required to make it pass.

```php
public function testApplicationsArrayNotEmpty()
{
    $resultArray = $this->Application->getApplications();

    $this->assertNotEmpty($resultArray);
}

// ... //

public function getApplications()
{
    $applications = $this->find('all');

    return $applications;
}
```

Clean Application Development – QA and unit tests

- **QA at the begining instead of the end**
  - QA waits for code to test.
  - Create tests based on requirements, first.
  - Developers write code to pass tests.
    - The end is not so bad now.



Clean Application Development – Agile

- **Agile = Project Iteration**
  - Average sprint is 2 weeks
  - At the end of the sprint EVERYTHING is “done“
    - Tests
    - Development
    - **Documentation**
  - Velocity charts, MAKE THEM PUBLIC
  - Charts allow business to recover gracefully

Clean Application Development – Human resources

- **Our clients hired a professional, they should get one**
  - Professionals are:
    - Trusted
    - Reliable estimates
    - Quality
    - Replaceable
      - Promotable
      - Documented
      - Repeatable
      - Vacation
    - Use standards/conventions

Clean Application Development – Close

- **Clean application development is:**
  - Learning, repetition, practice.
  - Clear architecture.
  - Coding standards and styles.
  - Framework and best practices.
  - Testing.
  - Agile.
  - Learning to say “NO“, and defend the code.
  - Living up to the title **“Professional“**

Clean Application Development – Resources

- **Resources**
  - Adam Culp @adamculp
  - http://www.GeekyBoy.com
  - http://RunGeekRadio.com
  - Book: “Clean Code“ by Robert C. Martin
  - Book: “Refactoring“ by Martin Fowler
    - https://github.com/adamculp/refactoring101
  - Book: “Refactoring 101“ by Adam Culp http://refactoring101.com
  - Book: “Guide to PHP Design Patterns“ by Jason Sweat
  - Book: “Design Patterns“ by Gang of four
  - http://www.php-fig.org

Thank You!