



# Modern PHP

Ben Ramsey





# My PHP Story








# Old-school PHP



PHP3: PHP: Hypertext Prep

web.archive.org/web/19980701120538/http://www.php.net/



# PHP: Hypertext Preprocessor

sourcesearch

downloads

documentation

support

in the news

projects

links

mirror sites

## PHP 3.0 Final is out!

See the [In The News](#) page for the release announcement and the accompanying press release.

## What is PHP?

PHP 3.0 is a server-side HTML embedded scripting language and it is a complete rewrite of the popular PHP/FI 2.0 language. This rewrite is faster, more robust and uses less memory than version 2. All users of version 2 are encouraged to upgrade.

If you are new to PHP, a good place to start discovering the power of this language is by clicking on the "Source" button in the top right corner of all the pages on this site. What you see is the actual PHP files that are behind each of the pages you are looking at.

## So, how much does it cost?

This may sound a little foreign to all you folks coming from a non-Unix background, but PHP doesn't cost anything. You can use it for commercial and/or non-commercial use all you want. You can give it to your friends, print it out and hang it on your wall or eat it for lunch. Welcome to the world of [Open Source](#) software! Smile, be happy, the world is good. For the full legalese, see the official [license](#).

## Year 2000 Compliance

Just like your C compiler or your favourite text editor, PHP is just about as year 2000 compliant as your pencil. If you're still worried (perhaps all pencils are built to break on December 31st 1999?) - [read this](#).

## Help DevShed Promote PHP Awareness, Literacy, and Development



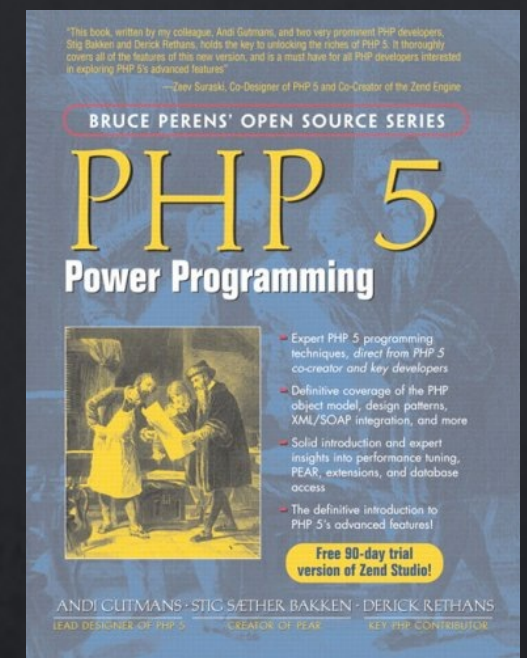
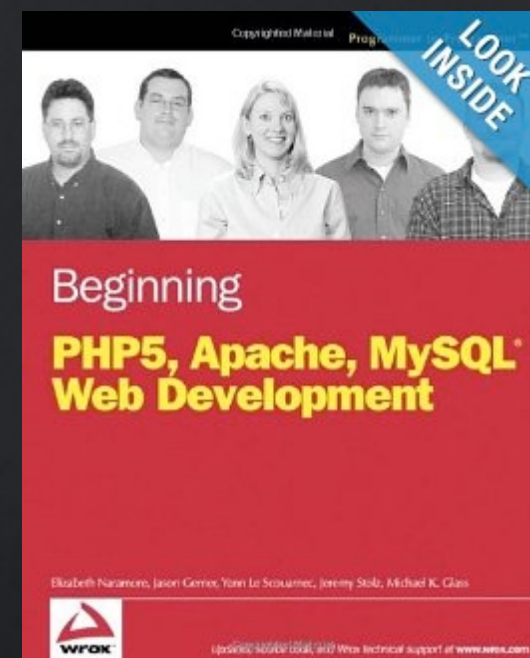
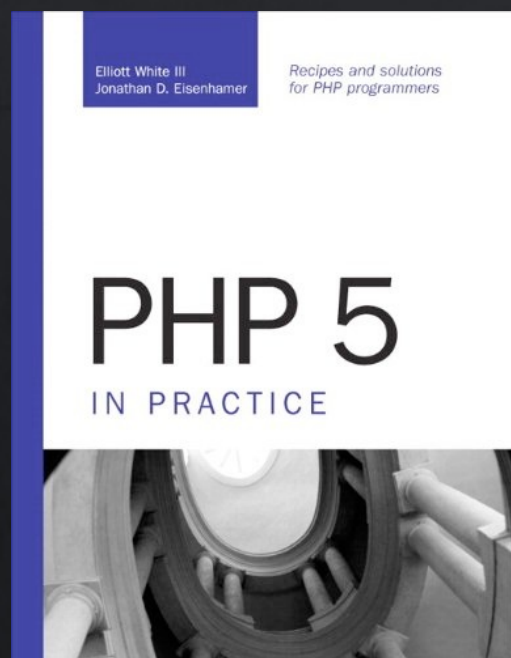
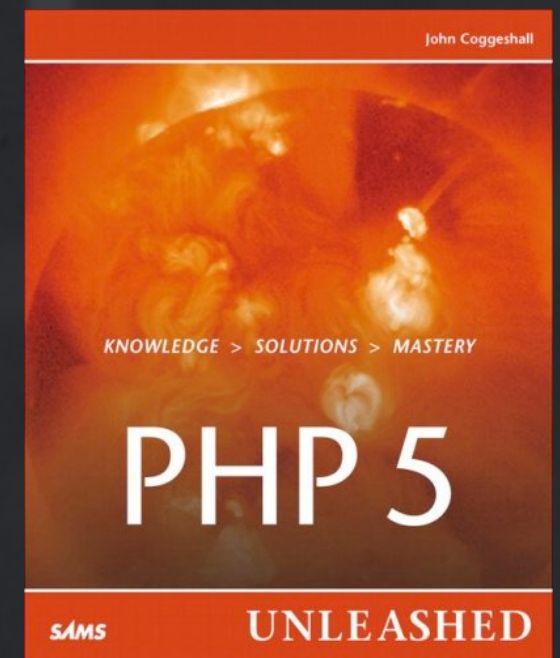
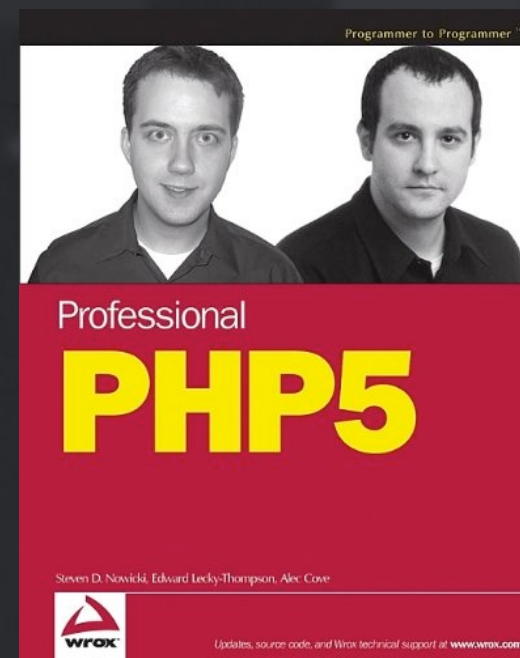
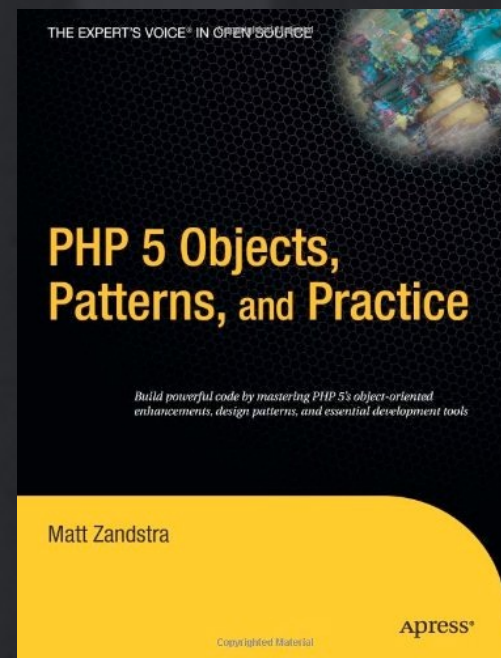
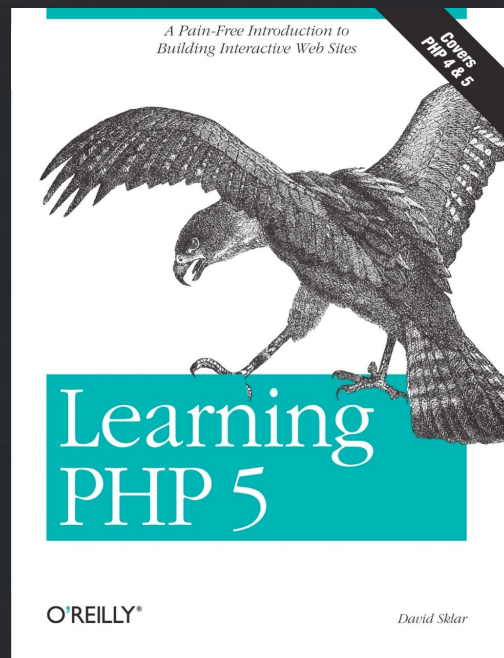
**.php3**  
**.php4**





# The Advent of PHP 5





# PHP 5 Features

- Brand new object model
- Standard PHP library, incl. iterators
- Type hinting
- Exceptions
- SimpleXML & DOM
- PDO



# PHP 5 Object Model

- Passed by reference
- Class constants
- Static methods/properties
- Visibility
- Abstract classes & interfaces
- Magic methods
- `__autoload()`

## **More changes in 5.1**

- Rewrite of date handling code, with improved timezone support.
- Significant performance improvements compared to PHP 5.0.X.
- PDO extension is now enabled by default.
- And more...



## **More still in 5.2**

- New memory manager
- Input filtering extension added
- JSON extension was added
- Hooks for tracking file upload progress were introduced
- Introduced DateTime and DateTimeZone objects
- And more...

## **Tons more in 5.3**

- Support for namespaces
- Late static binding
- Lambda Functions and Closures
- Syntax additions: NOWDOC, ternary short cut "?:" and goto, \_\_callStatic()
- Optional garbage collection
- Optional mysqlnd PHP native driver
- And more...



## **Keeping up the pace in 5.4**

- Traits, shortened array syntax
- Improved performance and reduced memory consumption
- Built-in webserver in CLI mode
- Register globals, magic quotes, and safe mode were removed
- And more...

## **Still going with 5.5**

- Generators and coroutines
- The finally keyword
- Simplified password hashing API
- Non-scalar Iterator keys in foreach
- list() constructs in foreach statements
- Zend OPcache extension
- And more...



## **And more in 5.6**

- **Constant scalar expressions**
- **Variadic functions**
- **Argument unpacking**
- **Support for large (>2GiB) file uploads**
- **SSL/TLS improvements**
- **New CLI debugger phpdbg**
- **And more...**

Modern PHP development isn't as much about changes in the language as it is about changes in how we build software with PHP.

The changes in the language support the ability to build software in new ways with new tools.



# OOP & Design Patterns





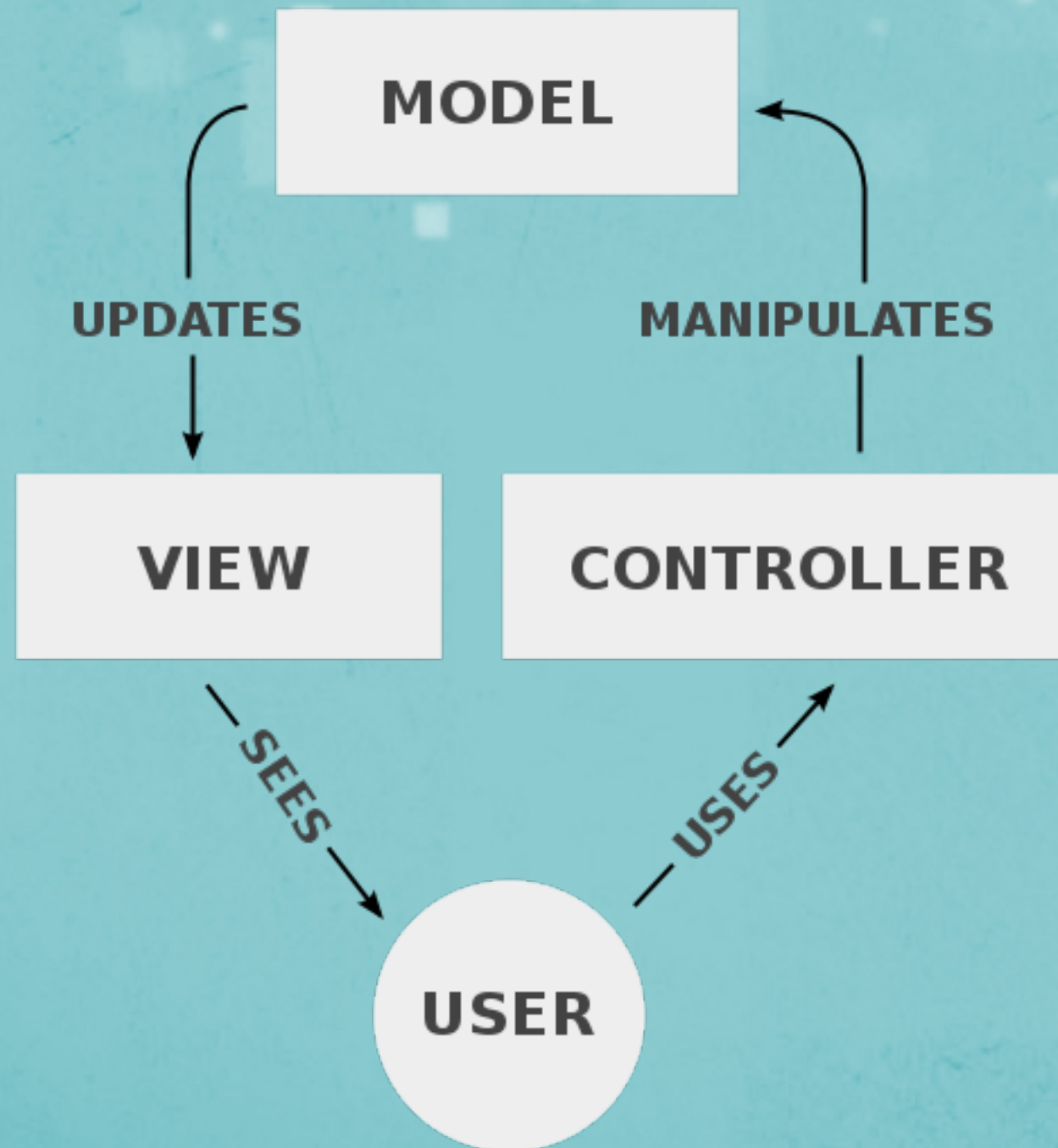
# Features of OOP

- Dynamic dispatch
- Encapsulation
- Polymorphism
- Inheritance
- Open recursion
- Abstraction
- Classes, instances, methods...

# Design Patterns

- Singleton
- Factory
- Decorator
- Iterator
- Adapter
- Front Controller





# OOP & Design Pattern Reading

- *Mastering Object Oriented PHP*  
by Brandon Savage  
[masteringobjectorientedphp.com](http://masteringobjectorientedphp.com)
- *php|architect's Guide to PHP Design Patterns* by Jason Sweat  
[www.phparch.com/books/phparchitects-guide-to-php-design-patterns](http://www.phparch.com/books/phparchitects-guide-to-php-design-patterns)
- *Learning PHP Design Patterns*  
by William Sanders  
[www.php5dp.com](http://www.php5dp.com)





# Security



# Attacks

- Cross-site scripting
- SQL injection
- Cross-site request forgery
- Session hijacking
- Session fixation

# Filter input

```
$clean = array();
$clean['widgetId'] = filter_input(
    INPUT_GET,
    'widgetId',
    FILTER_VALIDATE_INT
);

if ($clean['widgetId']) {
    $dbh = new \PDO($dsn, $user, $password);
    $sth = $dbh->prepare('
        SELECT
            id,
            name
        FROM widgets
        WHERE id = :widgetId
    ');
    $sth->execute($clean);
    $widget = $sth->fetch(\PDO::FETCH_ASSOC);
}
```



# Escape output

```
echo htmlentities($widget['name']);
```

# PHP's Data Filter Extension

- Introduced in PHP 5.2
- Provides validation and sanitization
- Selected functions:
  - `filter_input()`
  - `filter_var()`
  - `filter_input_array()`
  - `filter_var_array()`



Frameworks do much of this  
for us, now.

But we need to be diligent  
and learn and understand  
the principles.

# Security Reading

- *Essential PHP Security*  
by Chris Shiflett  
[phpsecurity.org](http://phpsecurity.org)
- Websec.io
- Anthony Ferrara's blog  
[blog.ircmaxell.com](http://blog.ircmaxell.com)



# Version Control





**FTP**

**Dreamweaver MX**

**CVS**



# Subversion

# Git Mercurial Bazaar

You have no excuse.

Just use **GitHub** or  
**BitBucket**.



Learn Git at [try.github.io](https://try.github.io).

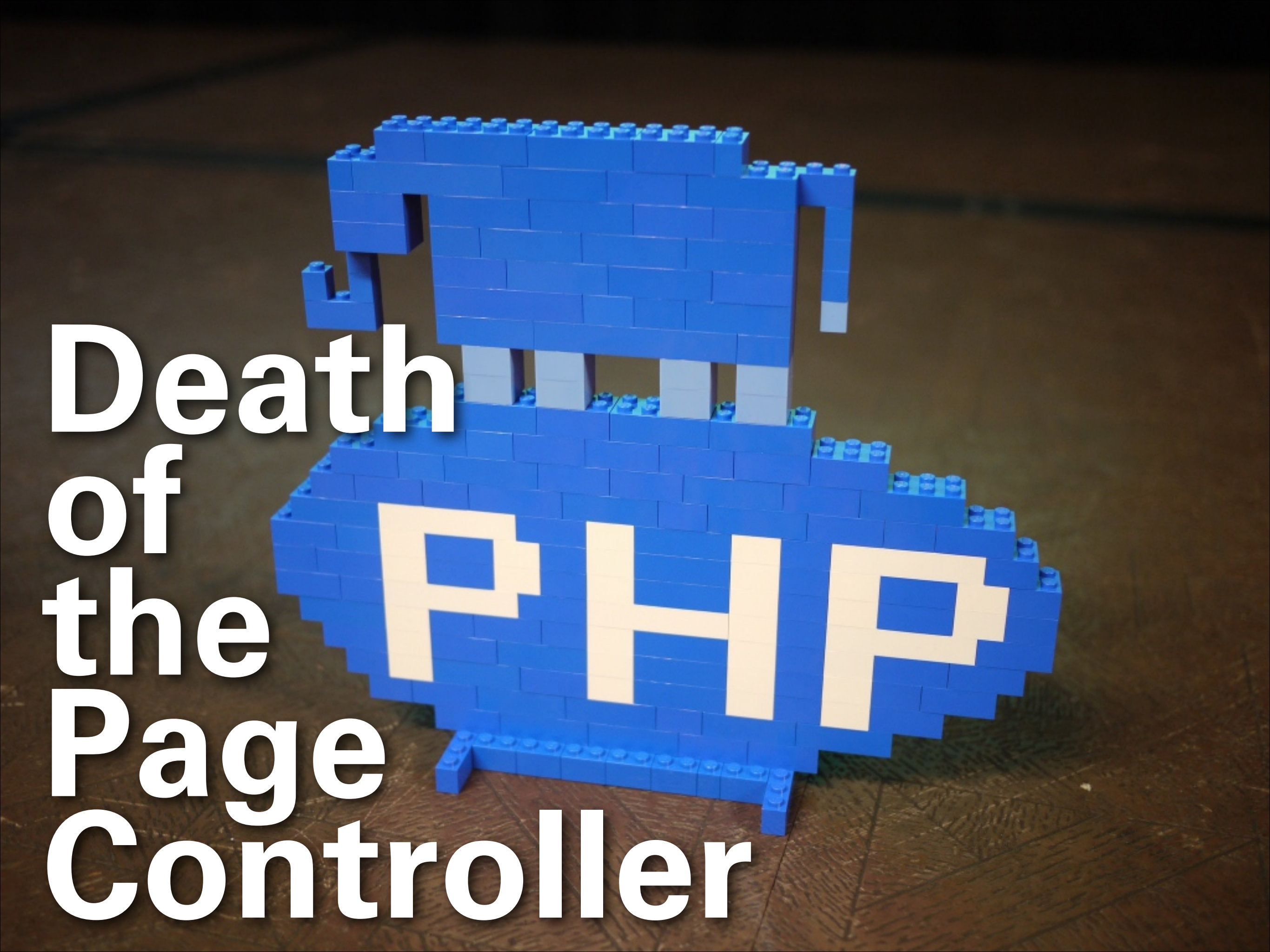


# Autoloading Practices

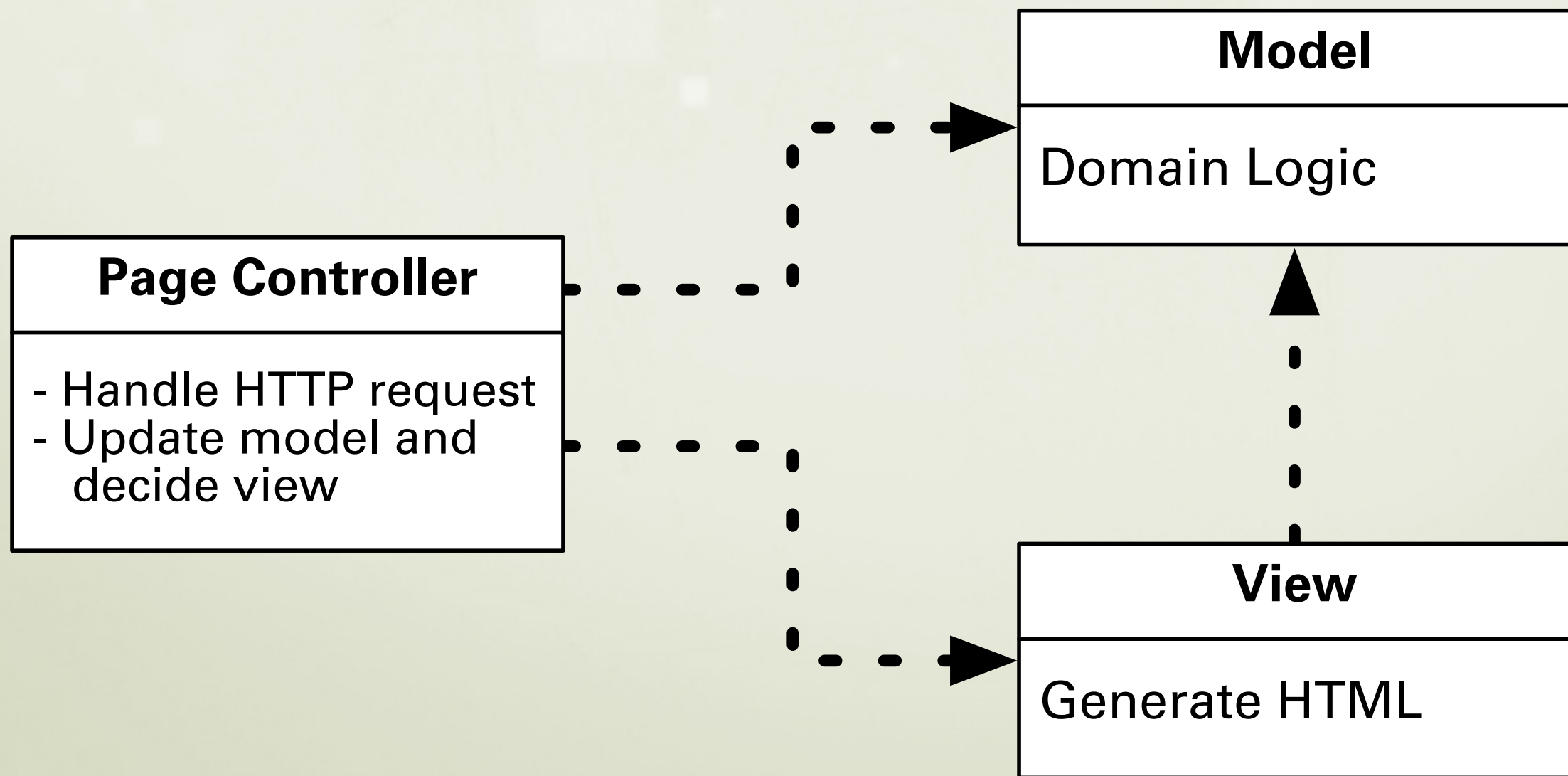
# Autoloading

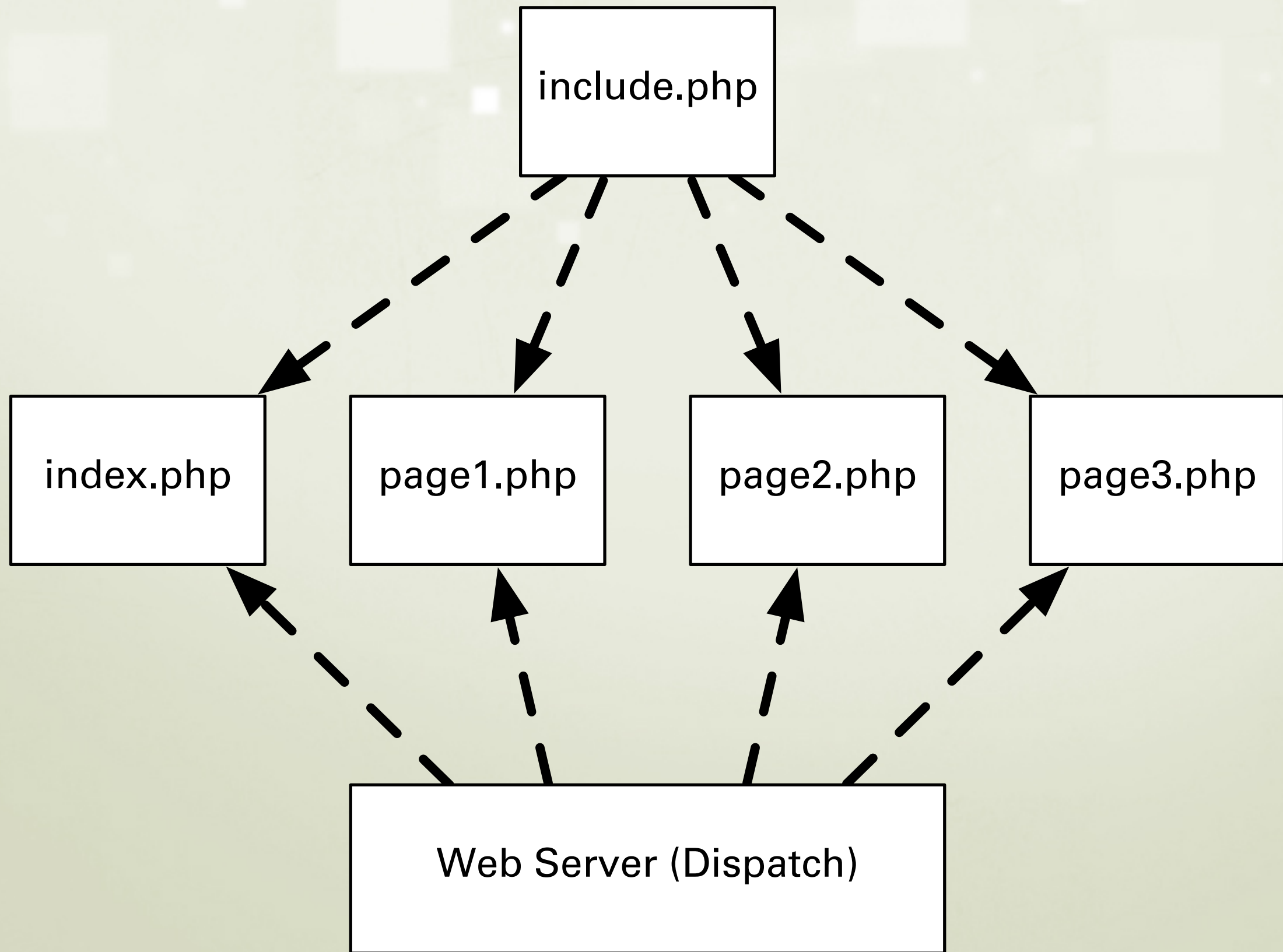
- Prior to PHP 5, we had to include/require every single class file we wanted to have available
- PHP 5 introduced `__autoload()`
- This has significantly changed the way we build applications





# Death of the Page Controller







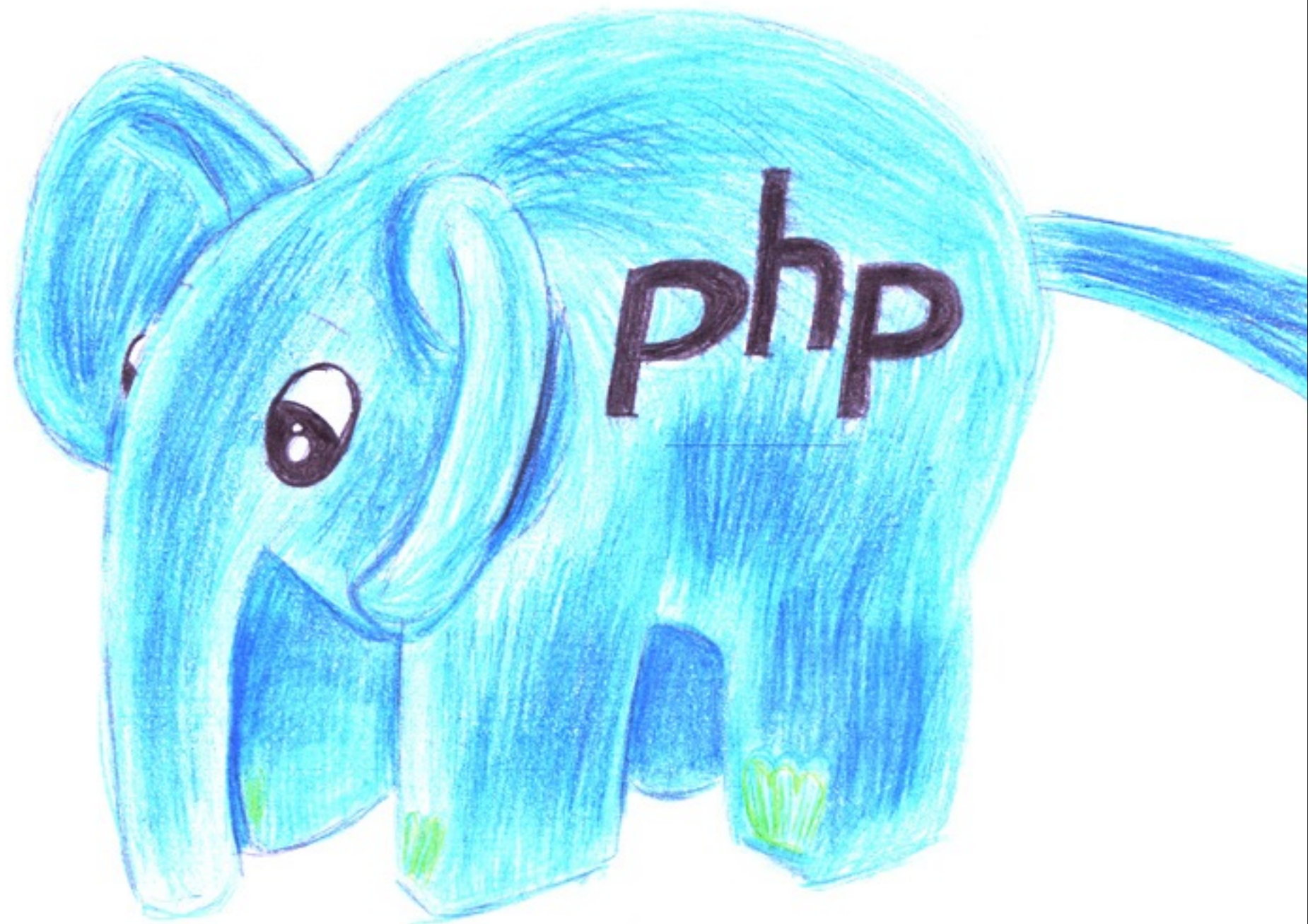
**web\_root/**  
├── **classes/**  
├── **config.php**  
├── **css/**  
├── **include.php**  
├── **index.php**  
├── **javascript/**  
└── **page1.php**

```
project/  
├── config.php  
├── lib/  
└── web/  
    ├── css/  
    ├── images/  
    ├── index.php  
    └── js/
```

Autoloading and design  
patterns paved the way to  
better code structure...



# Frameworks



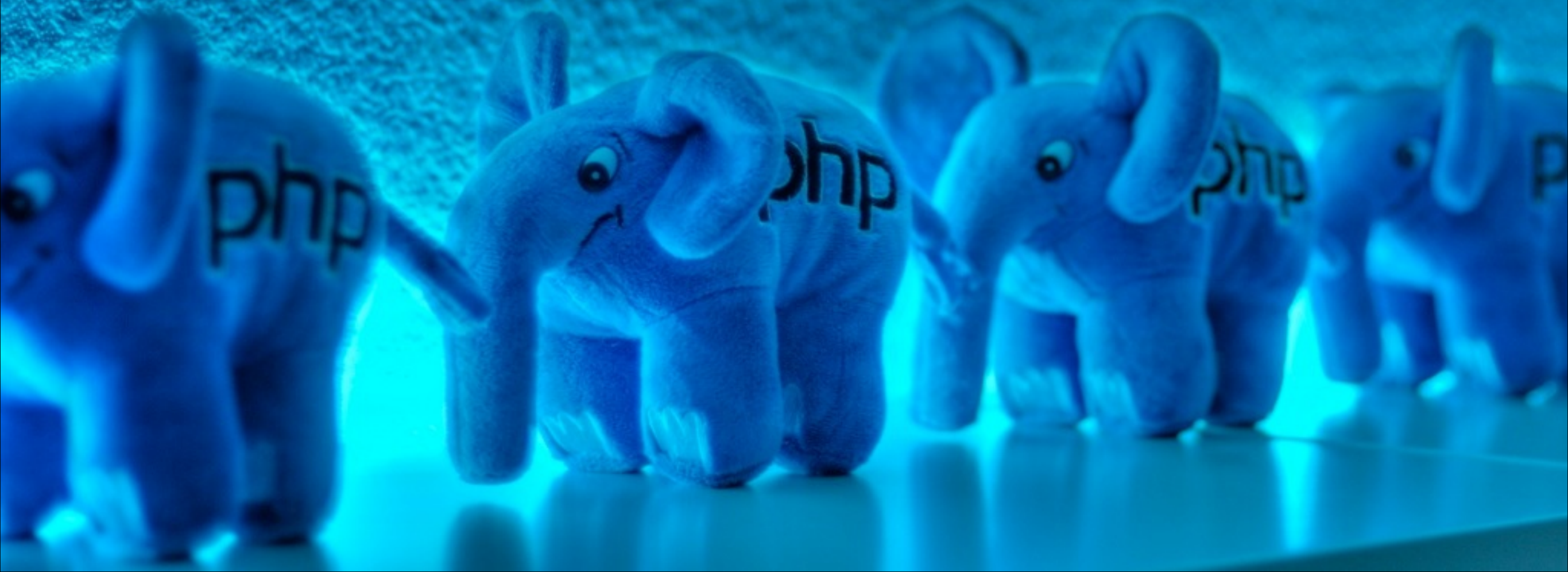
**Frameworks have ushered in  
a new era of constructing  
software.**

- Aura
- CakePHP
- CodeIgniter
- FuelPHP
- Joomla
- Laravel
- Lithium
- Symfony
- Zend Framework
- and more!



Not only have they made building software easier and faster, but they have created new communities, workflows, and toolsets.

# Framework Interoperability Group



# PHP-FIG



- PSR-0: Autoloading standard
- PSR-1: Basic coding standard
- PSR-2: Coding style guide
- PSR-3: Logger interface
- PSR-4: Improved autoloading

# PSR-0 Example

```
rhumsaa-uuid/  
├── src/  
│   ├── Rhumsaa/  
│   │   └── Uuid/  
│   │       └── Uuid.php  
└── tests/  
    ├── Rhumsaa/  
    │   └── Uuid/  
    │       └── UuidTest.php
```

```
use \Rhumsaa\Uuid\Uuid;  
use \Rhumsaa\Uuid\UuidTest;
```



# PSR-4 Example

```
rhumsaa-uuid/  
├── src/  
│   └── Uuid.php  
└── tests/  
    └── UuidTest.php
```



# Coding Standards

**The tabs vs. spaces war is  
over, and spaces have won.**

**;-)**

Consistency is the key.





# Tests

# **New terms in the PHP lexicon**

- Unit tests
- Functional tests
- TDD
- BDD
- Continuous integration
- Code coverage

# Testing Frameworks

- PHPUnit
- SimpleTest
- Behat
- Codeception

```
project/  
├── config/  
├── src/  
├── tests/  
└── web/  
    ├── css/  
    ├── images/  
    ├── index.php  
    └── js/
```






















Code Coverage for /home/ x



luthien/~ramsey/coverage/ramsey/uuid/



/home/ramsey/source/ramsey/uuid/src/Rhumsaa/Uuid ([Dashboard](#))

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	474 / 474		100.00%	62 / 62		100.00%	9 / 9
 <a href="#">Console</a>		100.00%	150 / 150		100.00%	9 / 9		100.00%	5 / 5
 <a href="#">Doctrine</a>		100.00%	14 / 14		100.00%	5 / 5		100.00%	1 / 1
 <a href="#">Exception</a>								100.00%	2 / 2
 <a href="#">Uuid.php</a>		100.00%	310 / 310		100.00%	48 / 48		100.00%	1 / 1

## Legend

**Low:** 0% to 35%   **Medium:** 35% to 70%   **High:** 70% to 100%

Generated by [PHP\\_CodeCoverage 1.2.13](#) using [PHP 5.5.4-1+debphp.org~precise+1](#) and [PHPUnit 3.7.29](#) at Fri Jan 31 18:19:08 UTC 2014.

# Continuous Integration Tools

- Jenkins, [jenkins-ci.org](http://jenkins-ci.org)
- Template for Jenkins Jobs for PHP Projects, [jenkins-php.org](http://jenkins-php.org)

# Testing Reading

- *The Grumpy Programmer's Guide To Building Testable PHP Applications* by Chris Hartjes

[grumpy-testing.com](http://grumpy-testing.com)

- *The Grumpy Programmer's PHPUnit Cookbook* by Chris Hartjes

[grumpy-phpunit.com](http://grumpy-phpunit.com)



# Dependency Injection





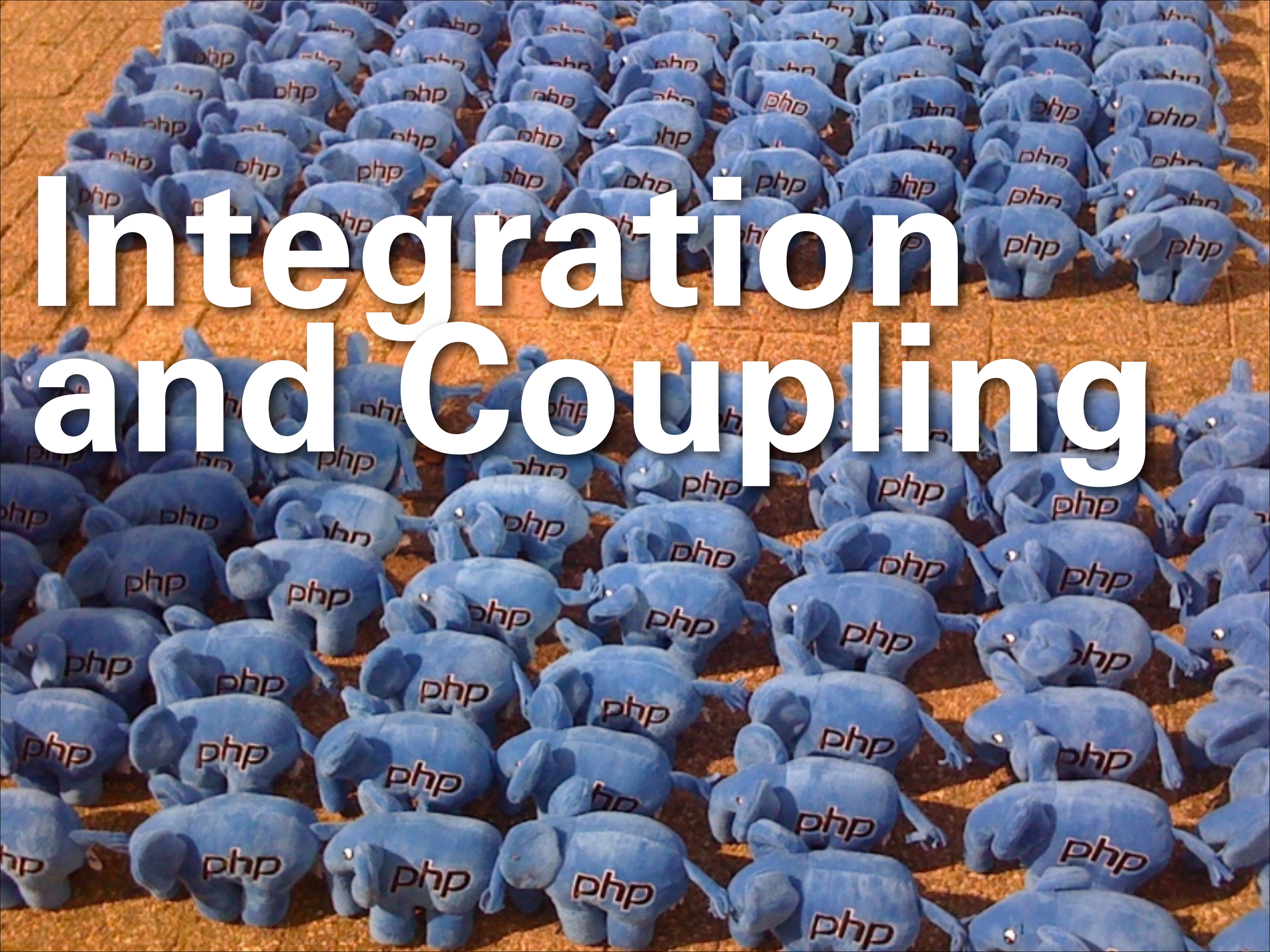
```
class Widget
{
    public function getById($id)
    {
        $db = new Database();
        $result = $db->query('
            SELECT *
            FROM widgets
            WHERE id = ?
        ', array($id));

        return $result;
    }
}
```

```
class Widget
{
    public function getById($id, Database $db)
    {
        $result = $db->query('
            SELECT *
            FROM widgets
            WHERE id = ?
        ', array($id));

        return $result;
    }
}
```





# Integration and Coupling



# APIs & Libraries

# Composer

# Composer

- Dependency manager for PHP
- [getcomposer.org](https://getcomposer.org)
- composer.json

```
{  
    "require": {  
        "rhumsoa/uuid": "~2.7"  
    }  
}
```



**PEAR?**





PHP is not  
only PHP



- Vagrant
- VirtualBox
- The cloud (AWS, Rackspace, etc.)
- PaaS (EngineYard, PagodaBox, AppFog, etc.)
- Web servers (Apache, Nginx, etc.)
- Databases (MySQL, MongoDB, etc.)
- Queuing (Gearman, RabbitMQ, SQS, etc.)





# JavaScript & HTML5





# Modern PHP



**project/**

— **.puppet/**

— **bin/**

— **config/**

— **src/**

— **tests/**

— **web/**

— **css/**

— **images/**

— **js/**

— **index.php**

— **.bowerrc**

— **.gitignore**

— **.travis.yml**

— **Gemfile**

— **README.md**

— **Vagrantfile**

— **bower.json**

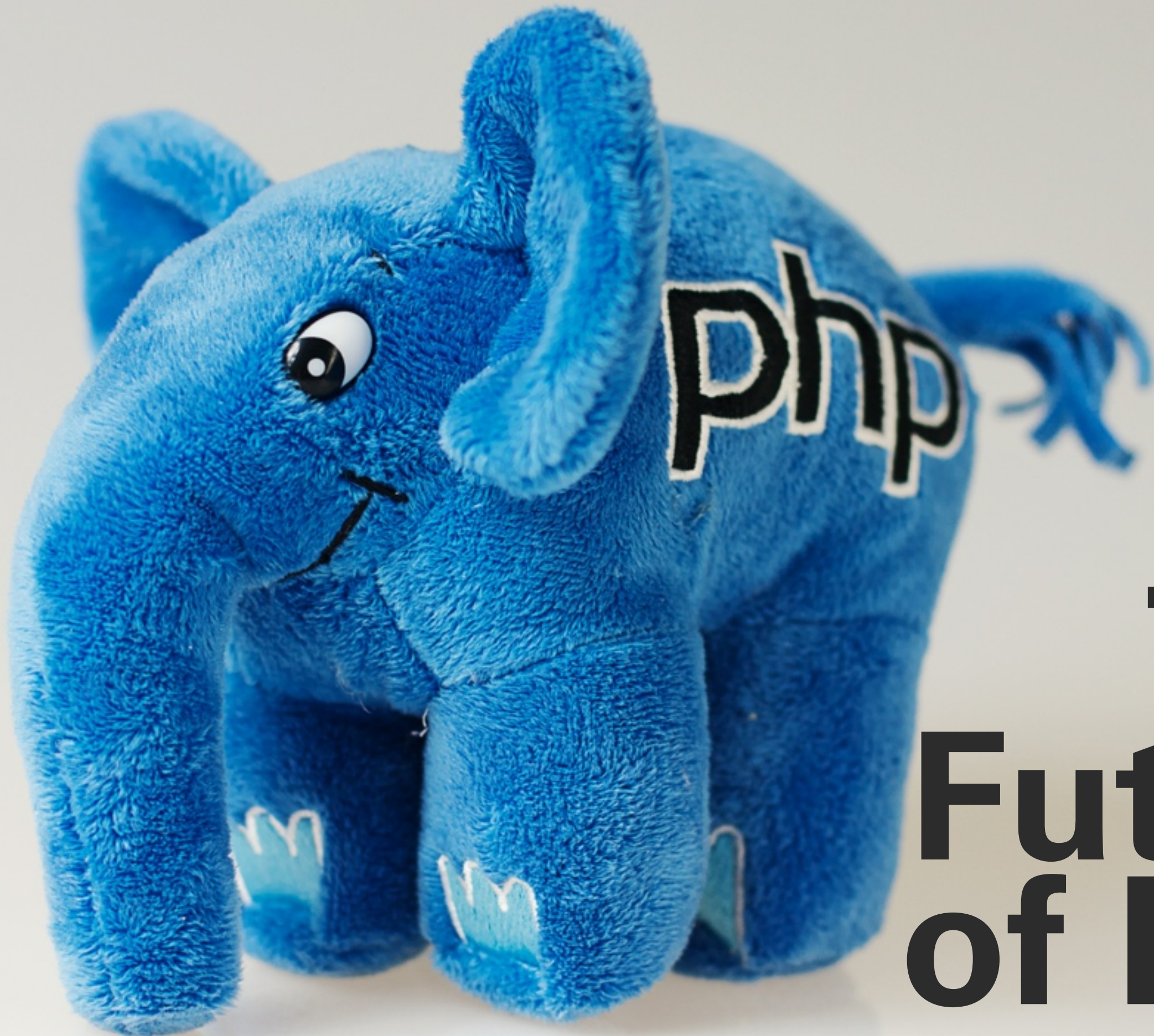
— **build.xml**

— **composer.json**

— **package.json**

— **phpunit.xml.dist**

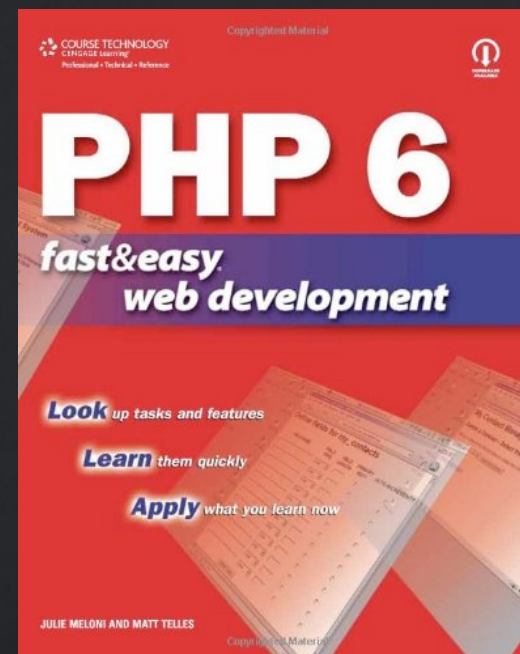
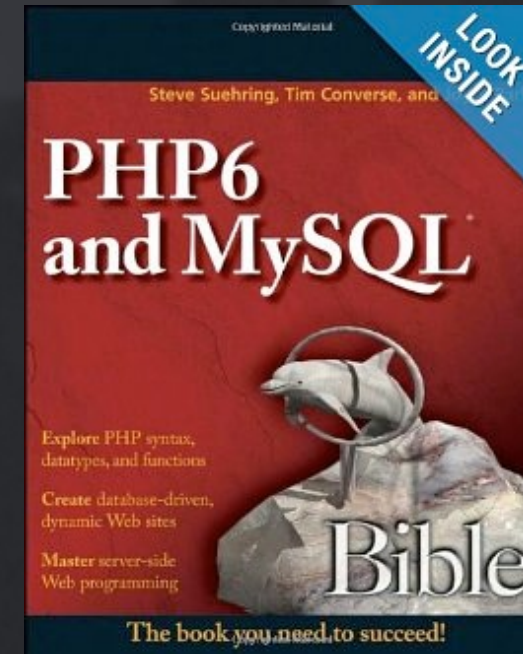
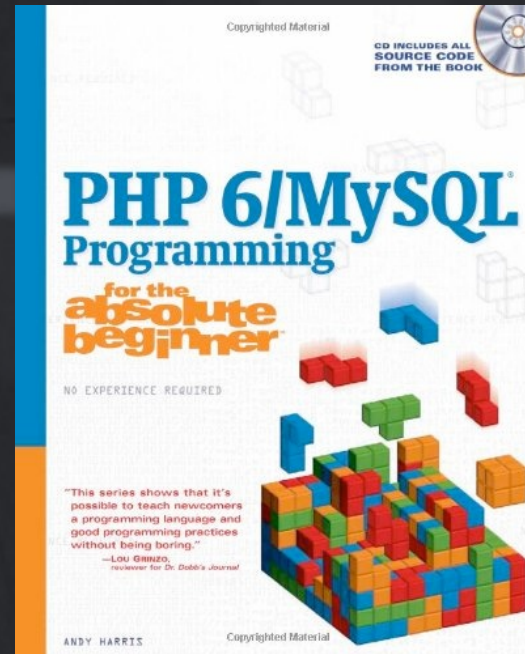
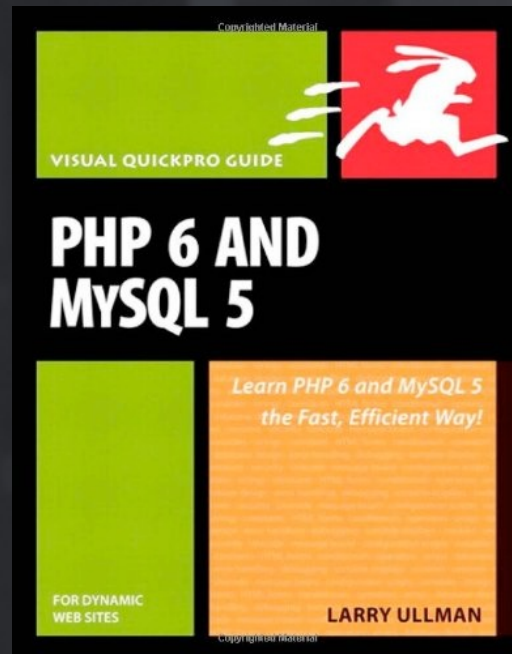




# The Future of PHP

# PHP 6?







PHP needs you.





# The Future of the PHP Community



# User groups



# Community conferences

# Frameworks

The PHP community needs  
you.



# Thank you

Ben Ramsey

[benramsey.com](http://benramsey.com)

[@ramsey](#)

[joinind.in/10790](https://joinind.in/10790)

# Check out...

PHP: The Right Way - [phptherightway.com](http://phptherightway.com)

Modern PHP

Copyright © Ben Ramsey. Some rights reserved.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported.

For uses not covered under this license, please contact the author.

Ramsey, Ben. "Modern PHP." AtlantaPHP User Group.  
Hypepotamus, Atlanta, GA. 6 Mar. 2014. Conference  
Presentation.



# Photo Credits

1. "Work in progress" by Loïc Doubinine,  
[flickr.com/photos/ztec/9204770134/](https://www.flickr.com/photos/ztec/9204770134/)
2. "Ben Ramsey" by Sebastian Bergmann,  
[flickr.com/photos/sebastian\\_bergmann/286847543](https://www.flickr.com/photos/sebastian_bergmann/286847543)
3. "PHPers out to do Amsterdam" by Aaron Wormus,  
[flickr.com/photos/aaron/200158232](https://www.flickr.com/photos/aaron/200158232)
4. "Part of the PHP Core Team" by Arnaud Limbourg,  
[flickr.com/photos/arnaudlimbourg/5164654691](https://www.flickr.com/photos/arnaudlimbourg/5164654691)
5. Untitled by Jeremy Kendall,  
[flickr.com/photos/jeremykendall/9088961213/](https://www.flickr.com/photos/jeremykendall/9088961213/)
6. "ElePHPants escaping from big giant mug" by Loïc Doubinine,  
[flickr.com/photos/ztec/9184943239/](https://www.flickr.com/photos/ztec/9184943239/)



# Photo Credits

7. "Elephpants at the pavilion" by Derick Rethans,  
[flickr.com/photos/derickrethans/6208407534](https://www.flickr.com/photos/derickrethans/6208407534)
8. "Two elePHPant parked" by Loïc Doubinine,  
[flickr.com/photos/ztec/9187378656/](https://www.flickr.com/photos/ztec/9187378656/)
9. "Elephpants in a row" by Rob Allen,  
[flickr.com/photos/akrabat/8128252662](https://www.flickr.com/photos/akrabat/8128252662)
10. Untitled by Eli White,  
[flickr.com/photos/eliw/8805534617/](https://www.flickr.com/photos/eliw/8805534617/)
11. "elePHPant" by Anna Filina,  
[flickr.com/photos/afilina/3308579171](https://www.flickr.com/photos/afilina/3308579171)
12. "elePHPants walking through the light" by Jakob Westhoff,  
[flickr.com/photos/jakobwesthoff/3213917240](https://www.flickr.com/photos/jakobwesthoff/3213917240)

# Photo Credits

13. Untitled by Terry Chay,  
[flickr.com/photos/tychay/1382823666](https://www.flickr.com/photos/tychay/1382823666)
14. "Chris practices being grumpy" by Rob Allen,  
[flickr.com/photos/akrabat/8421560178](https://www.flickr.com/photos/akrabat/8421560178)
15. "Secret ElePHPant date" by Tobias Schlitt,  
[flickr.com/photos/tobiasschlitt/2678580514/](https://www.flickr.com/photos/tobiasschlitt/2678580514/)
16. "Elephpant alliance" by Michelangelo van Dam,  
[flickr.com/photos/dragonbe/3411273755](https://www.flickr.com/photos/dragonbe/3411273755)
17. "Read a lot" by Martin Hassman,  
[flickr.com/photos/hassmanm/4754428088](https://www.flickr.com/photos/hassmanm/4754428088)
18. "Elephpants at Brighton Beach" by Derick Rethans,  
[flickr.com/photos/derickrethans/6207891017](https://www.flickr.com/photos/derickrethans/6207891017)

# Photo Credits

19. "elePHPant" by Drew McLellan,  
[flickr.com/photos/drewm/3191872515](https://www.flickr.com/photos/drewm/3191872515)
20. Untitled by Eli White,  
[flickr.com/photos/eliw/8806095443](https://www.flickr.com/photos/eliw/8806095443)