

Lecture 9

Web Framework



Outline

- ❄ Evolution Of Web Development
- ❄ Web Framework
- ❄ Zend Framework

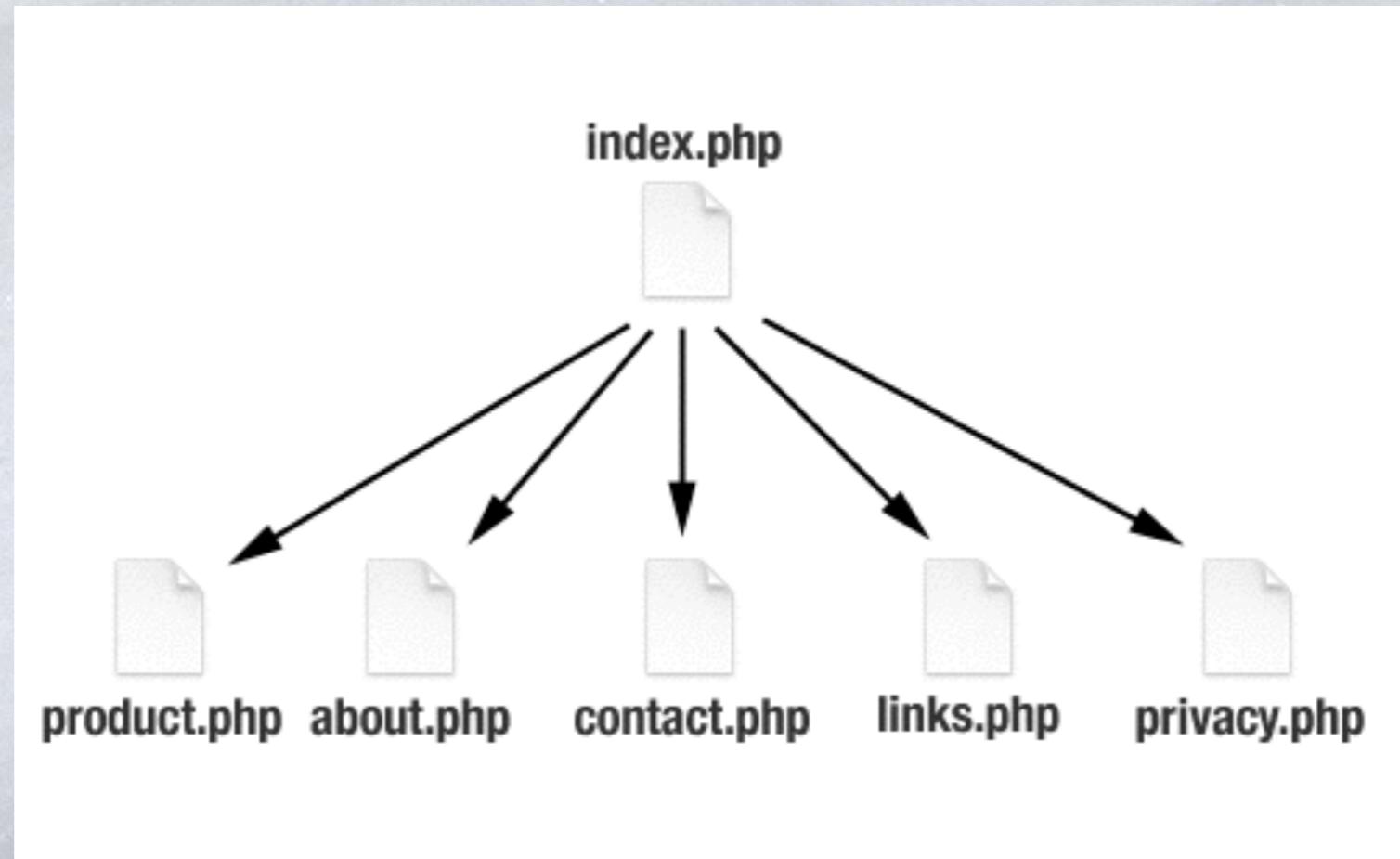
Evolution of Web Development

File
Inclusion

Template
Engine

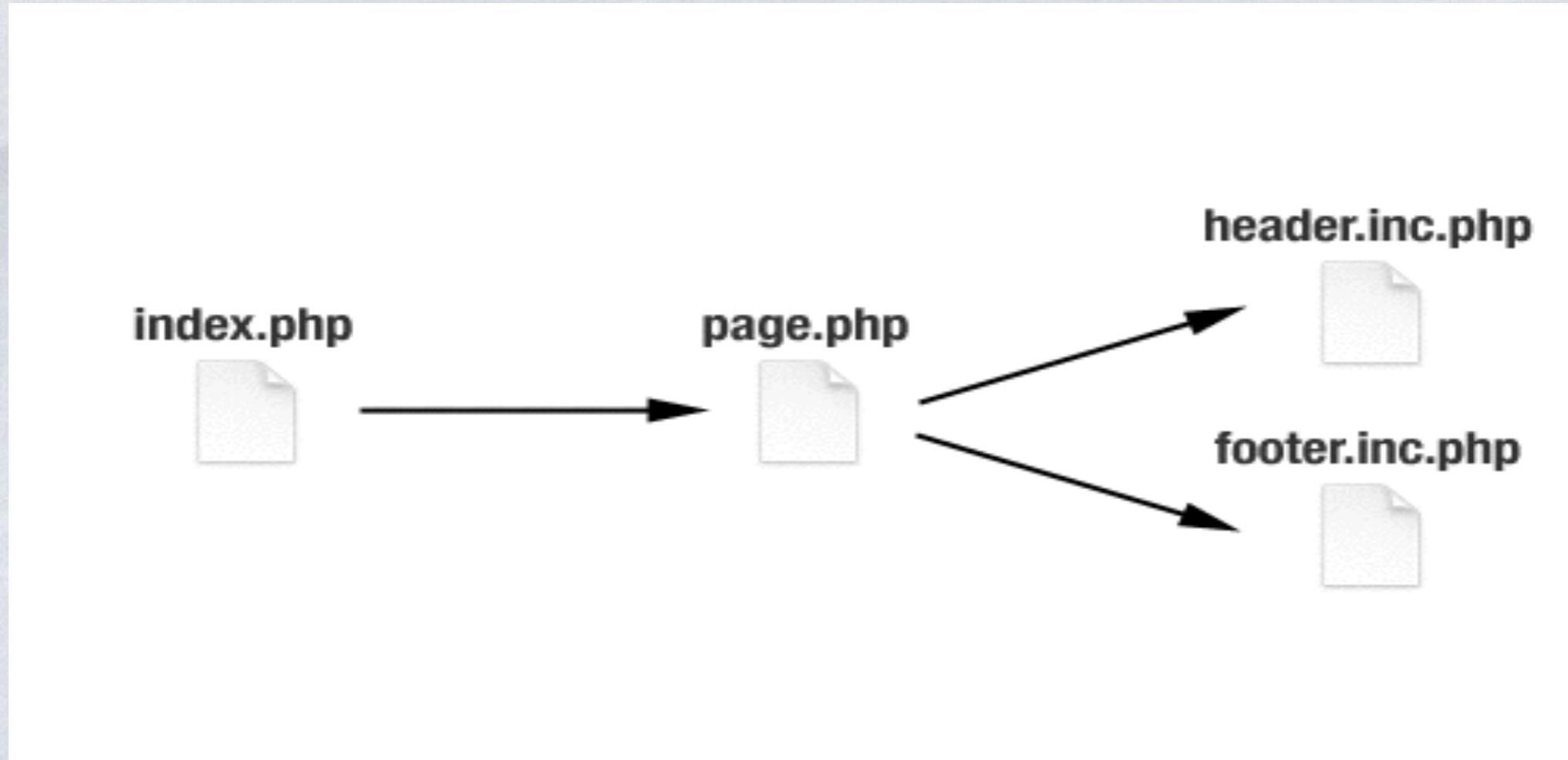
MVC
Framework

Evolution of Web Development



How you first started building
websites.

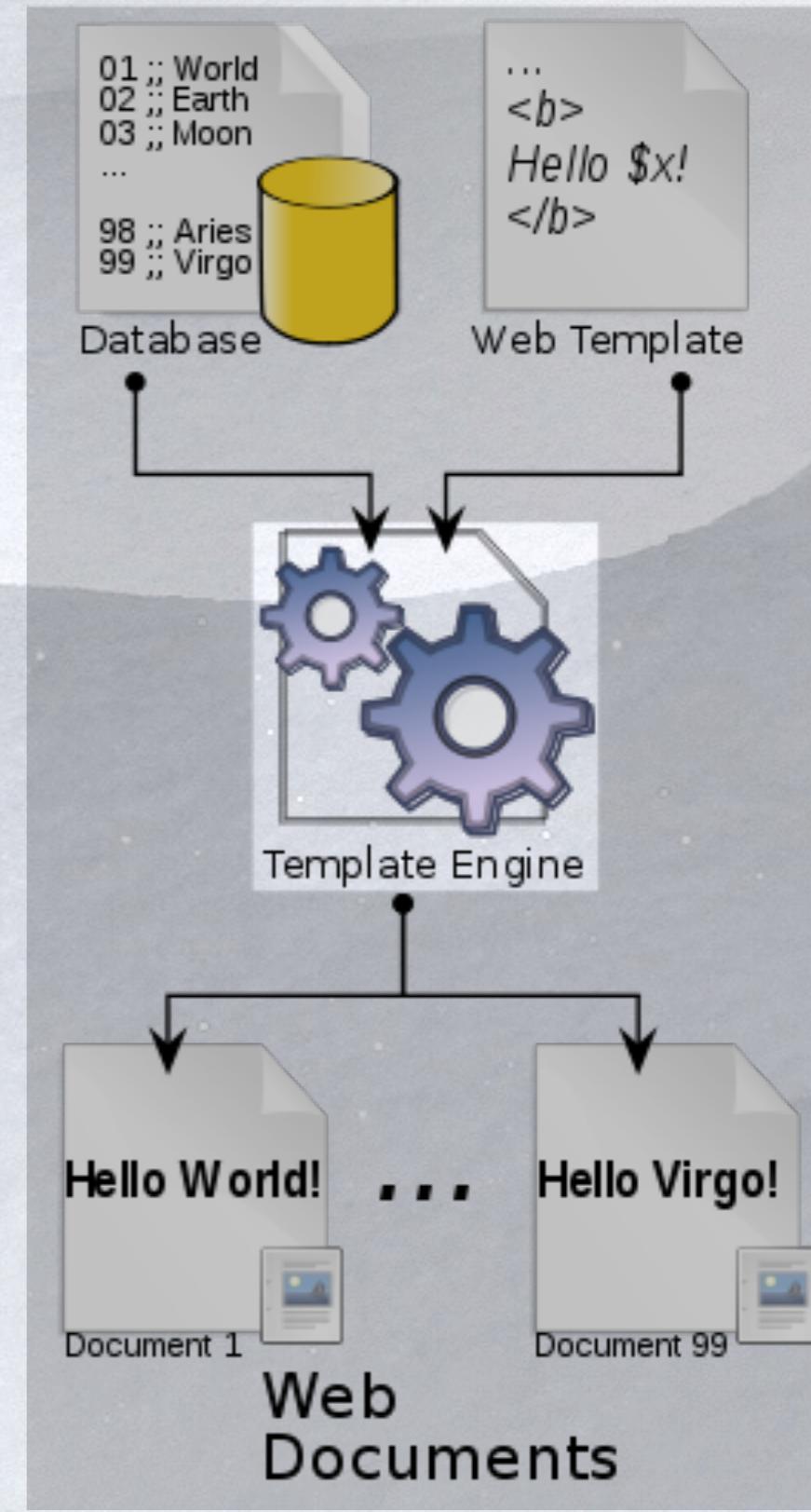
Evolution of Web Development



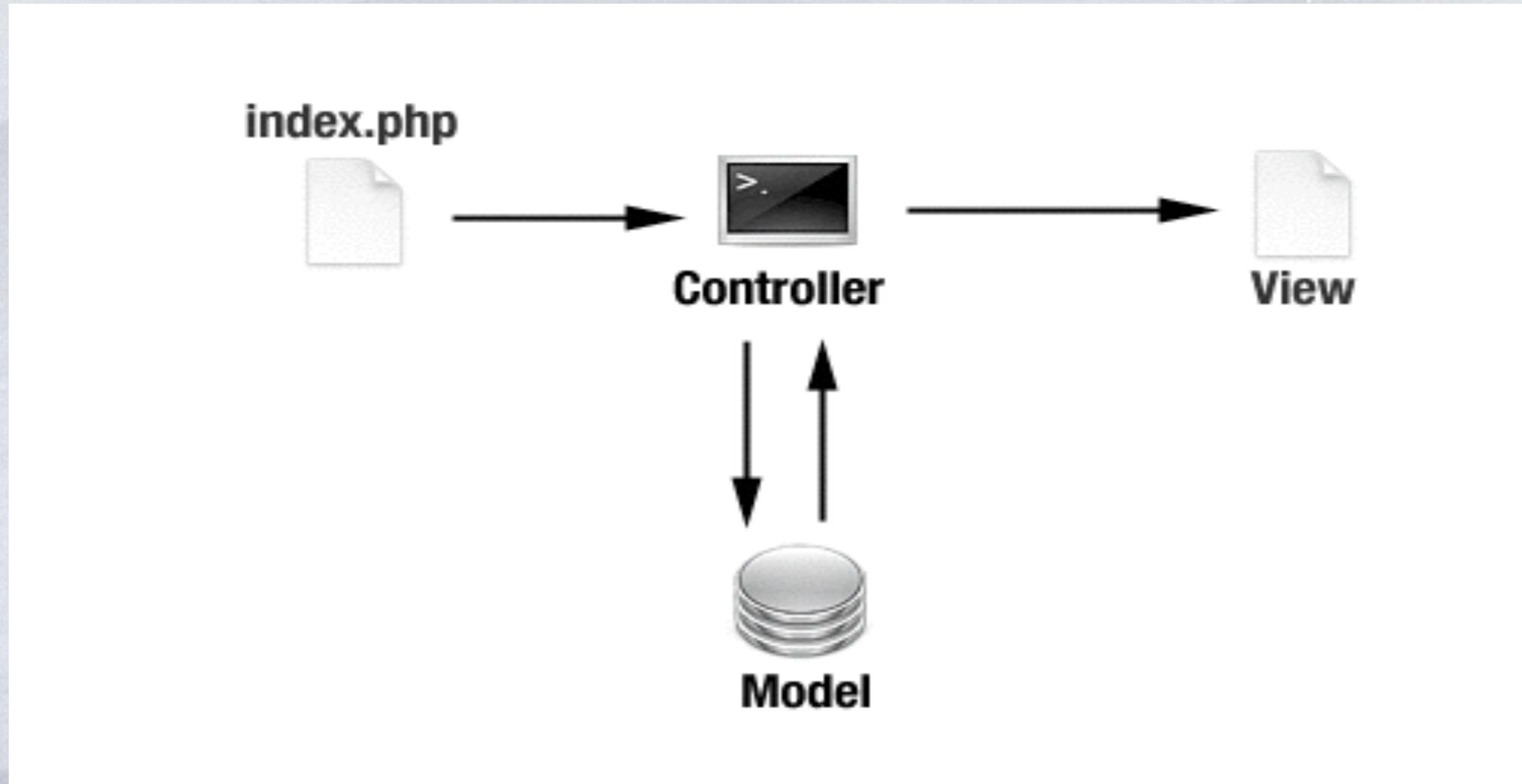
How you're building websites
now.

Evolution of Web Development

- ❖ Separate content from presentation on the web design.
- ❖ Suitable for websites often require regular content updates, and standardization of appearance.



Evolution of Web Development



How you build websites with a framework

Outline

- ❄ Evolution Of Web Development
- ❄ Web Framework
- ❄ Zend Framework

Basic Idea Of Web Framework

❖ A web application framework

- * Is a Software framework
- * Designed to support the development of
 - * Dynamic websites
 - * Web applications
 - * Web services
- * Aims to alleviate the overhead associated with common activities used in Web development.
 - * Libraries for database access
 - * Templating frameworks
 - * Session management
 - * Often promote code reuse
 - * Many more

Framework vs. Library / Toolkit

- ❖ A software framework, in computer programming, is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality
- ❖ A Library is a collection of useful material for common use, and it is defined as a collection of related software constructs such as classes, functions and subroutines used to develop software.
- ❖ Toolkit is an alias of Library in software engineering.
- ❖ A framework calls your code, but a library is called by your code.

Why Framework Not Scratch ?

* Key Factors of a Development

- * Interface Design
- * Business Logic
- * Database Manipulation
- * User Access Control

* Advantage of Framework

- * Templating
- * Provide Solutions to Common problems
- * Abstract Levels of functionality
- * Make Rapid Development Easier

* Disadvantage of Scratch Development

- * Make your own Abstract Layer
- * Solve Common Problems Yourself
- * The more Typing Speed the more faster

Outline

- ❖ Evolution Of Web Development
- ❖ Web Framework
- ❖ Zend Framework

Tonight we will cover:

❖ Zend Framework (ZF) basics

- * What, why, how

❖ ZF's flexible model-view-controller (MVC) implementation

- * Keeps your code organized logically and handles “plumbing”

❖ Components (libraries)

- * Including classes to access web services

❖ What's new and what's next

What is ZF?

★ An open source, MVC-based PHP framework

★ Is it a framework? Yes

- * "Glue" to build applications
- * Components developed, tested, distributed together

★ But loosely coupled

- * “Use at will” architecture
- * Dependencies documented in manual

Birth and early years

- ❖ 2005: PHP Collaboration Project at ZendCon
 - * Started as collection of components but coalesced
 - * PHP 5, object oriented (OO) from the start
 - * Set example of OO design patterns and practices
- ❖ July 2007: GA version 1.0
- ❖ February 17, 2009: version 1.75
- ❖ Sep, 2012: 2.0
- ❖ Frequent minor releases

What's it made of?

❖ Robust, high quality object-oriented PHP 5 libraries

- * Test-driven development
 - * 80+% code coverage
- * Peer review of all code
- * Coding standards
 - * <http://framework.zend.com/manual/en/coding-standard.html>
- * Components not released until documented in manual and inline
 - * phpDocumentor format: <http://phpdoc.org>

Flexible License

“New BSD” license

- * <http://framework.zend.com/license>
- * Use code however you like

Apache-like contributor license agreement (CLA)

- * Safety for users/corporations
- * All original code; no patent claims

Why I use it

❄ As I learn what it can do, the less boring code I write

- * At first, I reinvented the wheel
- * Realized that ZF already provided functionality
- * I can write less “plumbing” code

❄ Can customize and extend

- * Numerous integration points
- * Interfaces and small methods give you control

❄ It keeps up with trends and APIs

- * Compatibility with diverse database systems, authentication and other APIs
- * Web services

Community

Contributors include individuals and companies.

Companies include:

- * Zend (of course)
- * IBM
- * OmniTI

Technology partners:

- * Adobe, Google, IBM, Microsoft, nirvanix, Strikelron

Help available at zfforum.com, e-mail lists, IRC

Model-View-Controller

Model – View – Controller design pattern

Model

- * Classes that access your data

View

- * Templates to present that data (e.g. HTML for browser)

Controller (action controller)

- * Application flow
- * Connects model and view

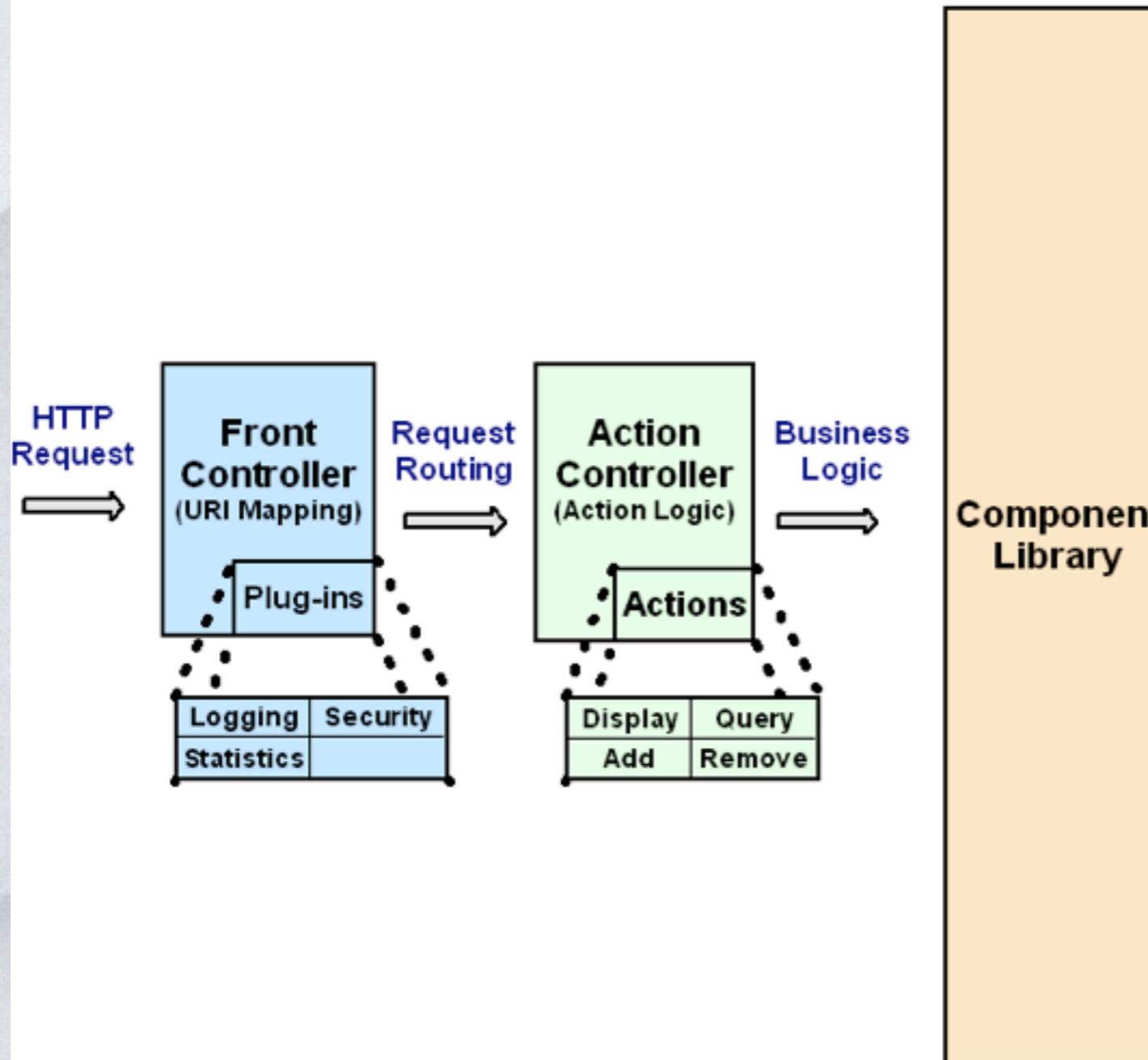
(Bonus: front controller!)

Front controller pattern

- ❖ Front controller sits in front of MVC
- ❖ All PHP requests funneled through index.php (bootstrap file)
- ❖ Front controller gets your application started
 - * Initializes request/response objects
 - * Can handle common settings and functionality
 - * “Include” paths
 - * Configurations
 - * Location of MVC components (if necessary)
 - * Logging, db (perhaps), authentication/authorization
 - * Converts URL to a “request” object with distinct parts
 - * Routes requests to appropriate action controllers
- ❖ Receives exceptions

Front controller to action controller

Zend Framework - Extremely Simple

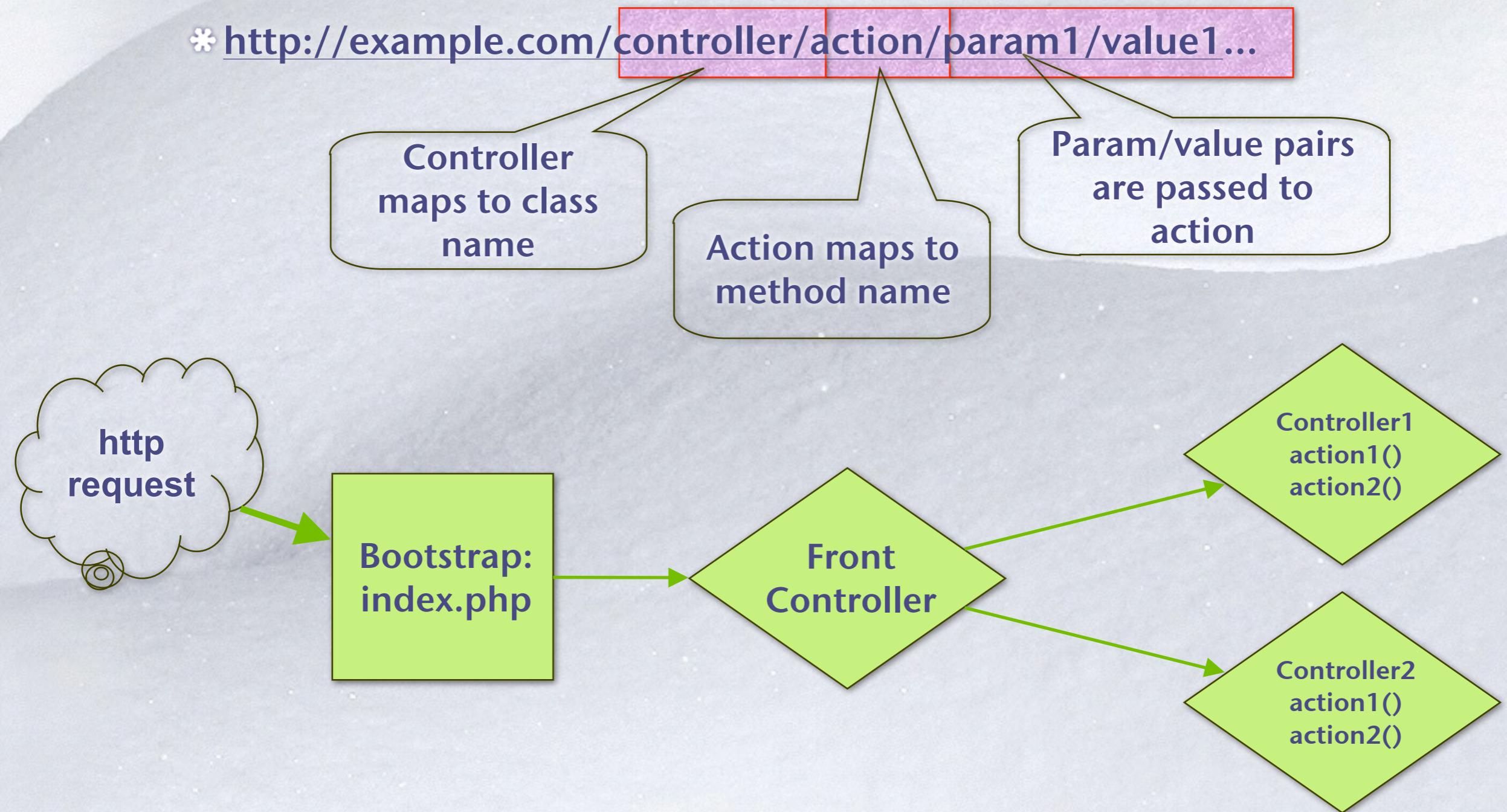


Frontend Generators	
ZTemplate	ZForm
ZAjax	ZGetText
ZPdf	
Data	
ZDbTable	ZActiveRecord
ZSearch	ZInputFilter
ZXQuery	ZDbSelect
ZRequest	ZXmlRepository
Web Services	
ZRest (Google, Yahoo!, eBay, Amazon)	ZSoap, ZWsdl
ZXmlRpc	ZFeed
User	
ZAuth	ZAcl
ZSession	
Misc	
ZNavigation	ZConfiguration
ZMail	ZLog
ZUri	ZComponent
Documentation	
phpDoc	Manual (DocBook)

Routes URL request

Default routing convention:

* http://example.com/controller/action/param1/value1...

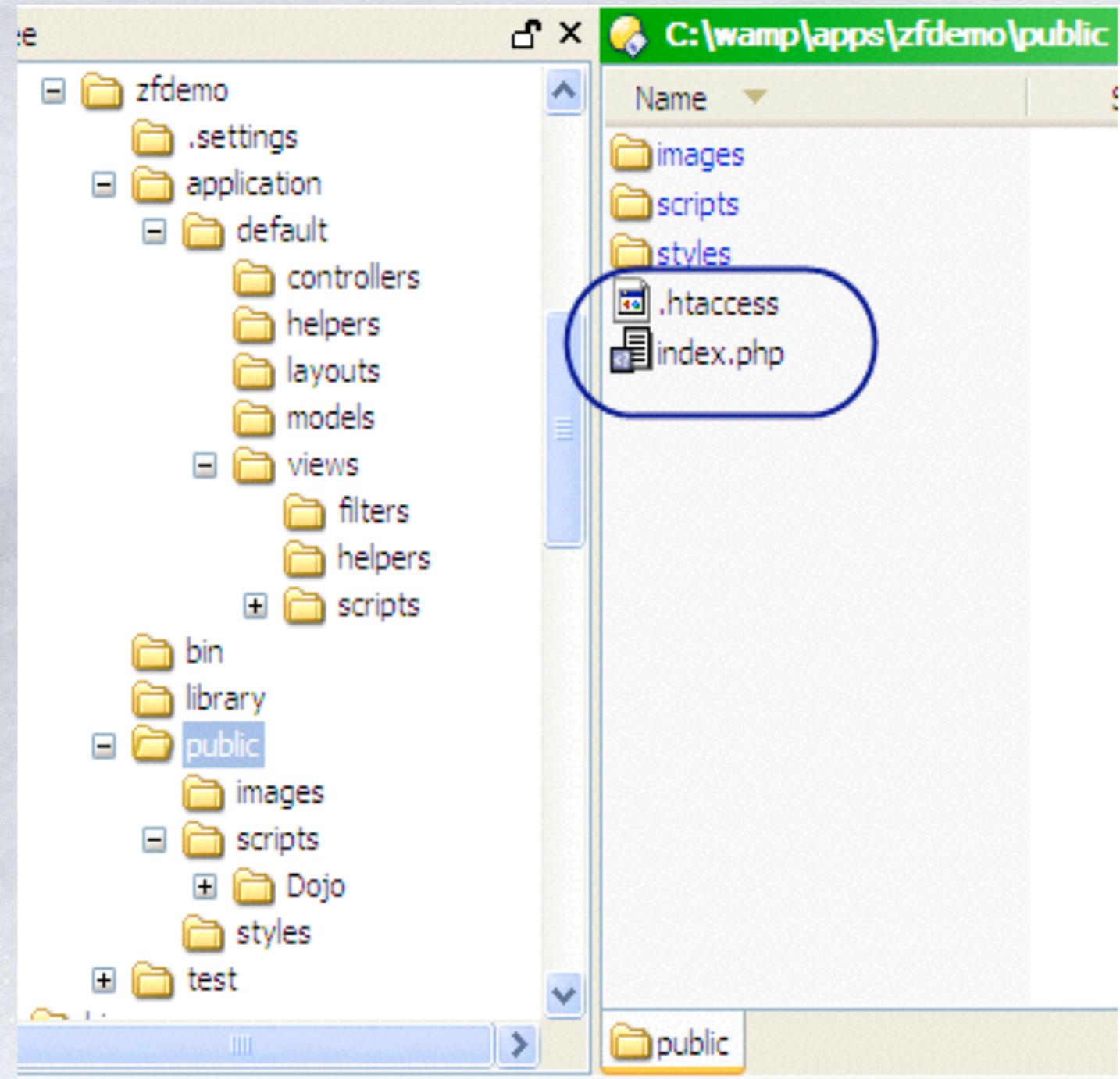


Front controller needs two files in public folder

❄ In document root (public folder):

❄ .htaccess
redirects requests
to bootstrap script
(index.php)

❄ index.php
instantiates
Front Controller



Front controller file #1: .htaccess

```
RewriteEngine on
```

```
# funnel all requests to index.php
```

```
# except requests for static resources
```

```
RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php
```

Front controller file #2: index.php

```
<?php  
// bootstrap file  
  
setincludepath('' . PATHSEPARATOR . '../library' .  
PATHSEPARATOR . '../application/default/models/' .  
PATHSEPARATOR . getincludepath());  
  
// Prepare the front controller  
$frontController = Zend_Controller_Front::getInstance();  
  
// Dispatch the request using the front controller  
$frontController->dispatch();
```

Action Controller

* Controller classes handle groups of request URLs

- * `http://example.com/controller/action`
- * Default: `IndexController`
- * Organizes and groups functionality
- * One class (`extending Zend_Controller_Action`) for each controller

* Action methods in each controller class handle requests

- * `http://example.com/controller/action`
- * Default: `indexAction()`
- * Named like `actionAction()`
- * Example: If action is “edit” then method is `editAction()`

More controller functionality

❖ Several standard methods help organize and control the flow

- * `init()` – called by the constructor
- * `preDispatch()` – called before the action's method
- * `postDispatch()` – called after the action's method

❖ Utility methods

- * `forward()`, `redirect()`, `getParam()`, `getRequest()`, `getResponse()`, `render()`

❖ Action helpers add functionality

- * Built-in helpers. Example: `gotoSimple()`
- * Your own helpers
- * Avoids the need to build your own base controller class

Controller example

The screenshot shows a PHP development environment with the following interface elements:

- PHP Explorer**: A tree view on the left showing the project structure:
 - demo
 - application
 - default
 - controllers (containing ErrorController.php and IndexController.php)
 - helpers
 - layouts
 - models
 - views
 - bootstrap.php
 - Initializer.php
 - library
 - public
 - test
 - Include Paths
 - JavaScript Support
- IndexController.php**: An open editor window on the right containing the following PHP code:

```
<?php
require_once 'Zend/Controller/Action.php';

class IndexController extends Zend_Controller_Action
{
    /**
     * The default action - show the home page
     */
    public function indexAction()
    {
        // Use default value of 1 if id is not set
        $id = $this->getParam('id', 1);

        // assign id to view
        $this->view->id = $id;
    }
}
```

Action helpers

❖ They extend Zend_Controller_Action_Helper_Abstract

❖ Built-in action helpers

- * ActionStack
- * AjaxContext
- * ContextSwitch
- * AutoComplete: Dojo, Scriptaculous
- * FlashMessenger
- * Json
- * Redirector
- * Url
- * ViewRenderer

❖ Create your own

- * Place in the "My/Helper/" directory of your library (or any directory on your includepath)

Action helper example: Redirector gotoSimple()

```
*class Forward_Controller extends Zend_Controller_Action
{
    protected $redirector = null;

    public function init()
    {
        $this->redirector = $this->helper->getHelper('Redirector');
    }

    public function myAction()
    {
        /* do some stuff */

        // Redirect to 'my-action' of 'my-controller' in the current
        // module, using the params param1 => test and param2 => test2
        $this->redirector->gotoSimple('my-action',
            'my-controller',
            null,
            array('param1' => 'test',
                  'param2' => 'test2'
                  )
            );
    }
}
```

View

❄ Scripts (templates)

- * PHP-based script templates to present data
- * Should contain only display logic, not business logic
- * Default naming: “myaction.phtml”

❄ Helpers

- * Classes and methods that provide reusable view functionality
 - * Examples of built in view helpers: escape(), formText(), partial(), partialLoop(), headTitle()
 - * Write your own, too

❄ Output filters

❄ Layout

❄ Placeholders

Zend_Layout

❖ Two-step view pattern

- * Uses Zend_View for rendering

❖ Placeholders useful for setting javascript, titles, other variable data

❖ Layout view helper

- * shortcut to layout placeholder

- * These are equivalent:

```
// fetch 'content' key using layout helper:  
echo $this->layout()->content;  
  
// fetch 'content' key using placeholder helper:  
echo $this->placeholder('Zend_Layout')->content;
```

❖ Enable in bootstrap

```
// accept default path  
Zend_Layout::startMvc();  
  
// or specify path  
$layoutPath = realpath(dirname(__FILE__) . '/../application/layouts');  
Zend_Layout::startMvc(array("layoutPath" => $layoutPath));
```

My own view helper: TitleCase.php

```
<?php

Require_once 'Zend/View/Interface.php';

/**
 * TitleCase helper
 */

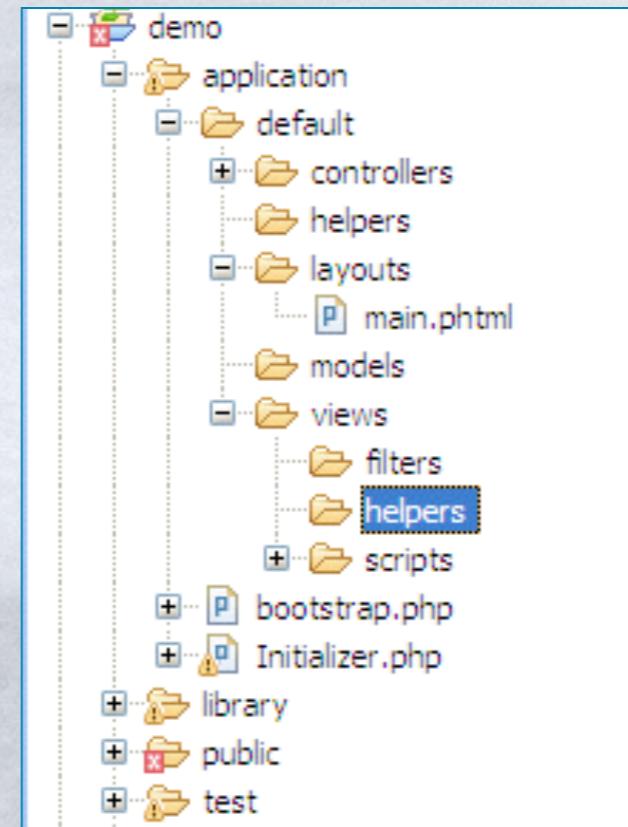
class Zend_View_Helper_Title_Case {
    public $view;

    public function titleCase($string = '')
    {
        // convert incoming string so that
        // first letter of each word is capitalized/
        // but all other letters are lowercase.

        // Also trim the string.

        return ucwords(strtolower(trim($string)));
    } // (public function titleCase())

    public function setView(Zend_View_Interface $view) {
        $this->view = $view;
    }
}
```



Usage:

```
echo $this->titleCase('mozilla
firefox');
// Mozilla Firefox
```

Model

 **Models are abstract representations of data**

- * Can be extended from:
 - * `Zend_Db_Table_Row` – For database abstraction
 - * `Zend_Feed_Element` – For RSS abstraction
 - * `Yahoo_Result` – For Yahoo abstraction
 - * Or any other class that fits your needs
 - * Or build your own own abstract representations of your data

Models (more)

- ❖ Model classes contain business logic to prepare complex data for presentation
- ❖ I stuff any “weird” code in models so that controllers/views are clean

MVC your way: custom routing rules

* Can declare custom routing rules with Zend_Controller_Router_Route classes

- * Not limited to “controller/action/param” format
- * Example: “year/month/day/title” as in a blog posting:
- * URL: <http://myblog.com/2008/02/24/i-like-routers>

```
$route = new Zend_Controller_Router_Route(
    ':year/:month/:day/:title',
    array(
        'year'      => 2009,
        'controller' => 'articles',
        'action'     => 'show'
    ),
    array(
        'year'  => '\d+',
        'month' => '\d+',
        'day'   => '\d+',
    )
);
$router->addRoute('posts', $route);
```

MVC your way: multi-MVC

- * Can separate app sections into modules (Mitch P.'s “atomic MVC”)

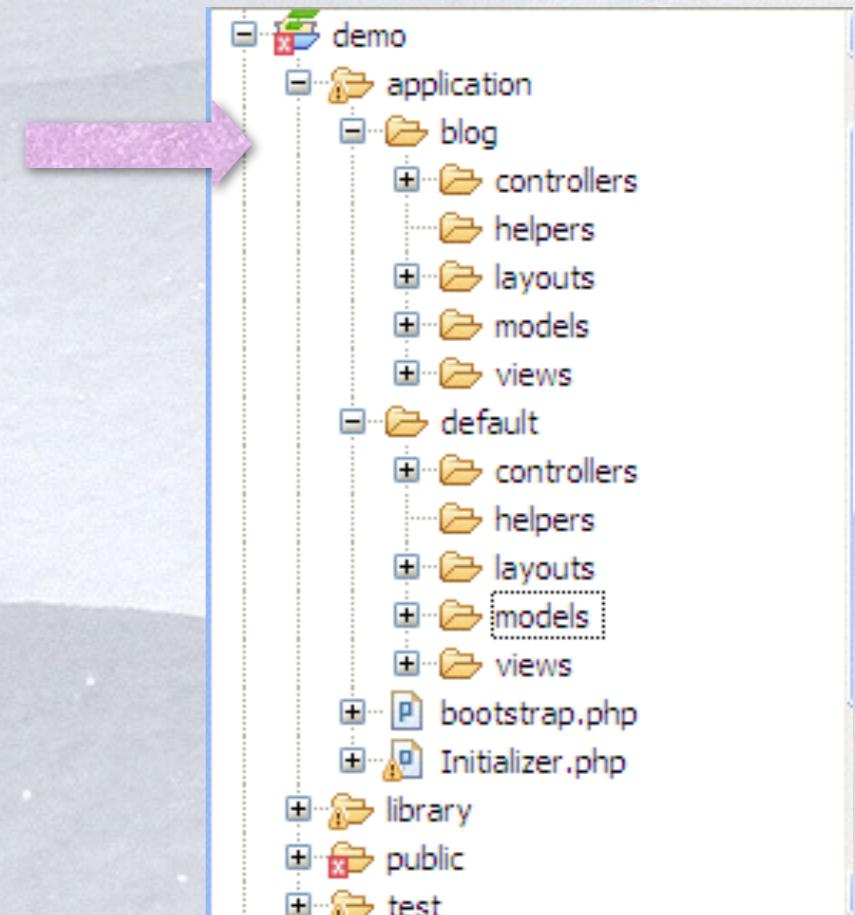
```
$front->setControllerDirectory(array(  
    'default' => '/demo/application/controllers',  
    'blog' => '/demo/application/blog/controllers' ));
```

- * Blog_IndexController is class name of index action controller within blog app/module

- * URL: example.com/blog or /blog/index or /blog/index/index

- * IndexController is still class name within default app/module

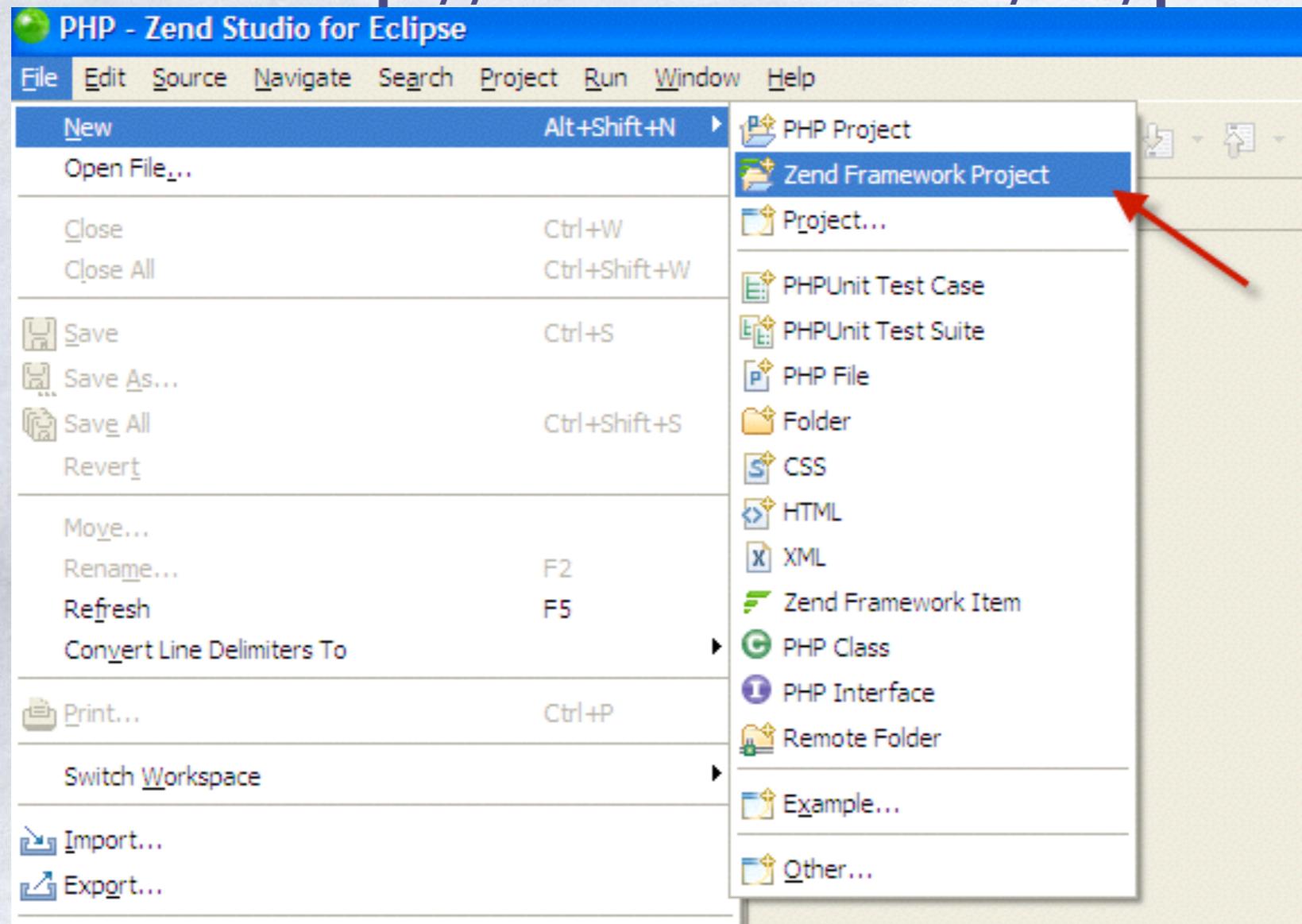
- * URL: / or /index or /index/index



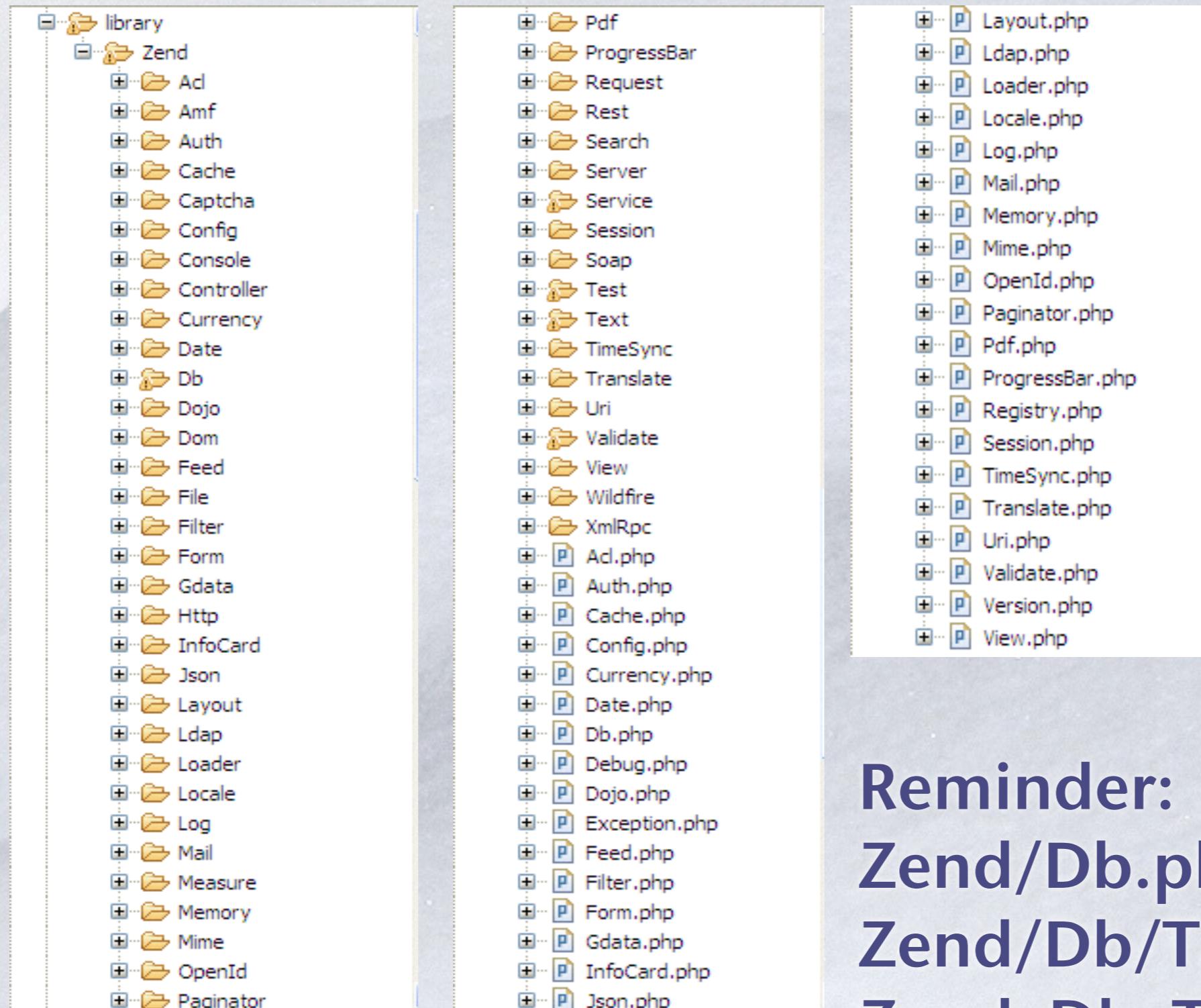
Tools can help get you started

* Zend Studio for Eclipse creates default directory structures and classes for you

* Free trial: <http://www zend com/en/products/studio/>



Library of Zend components



Reminder:
Zend/Db.php = Zend_Db
Zend/Db/Table.php =
Zend_Db_Table

Components vs. straight PHP

❄ Additional functionality

- * Zend_Session vs. straight \$_SESSION
- * Zend_Debug::dump() vs. var_dump

❄ Thread-safety

- * Zend_Translate vs. gettext
- * Zend_Locale vs. setlocale

❄ OO

- * OO Syntax
- * May be extended with custom functionality
- * ZF components reduce coding when used together

❄ They've been tested and integrated for you

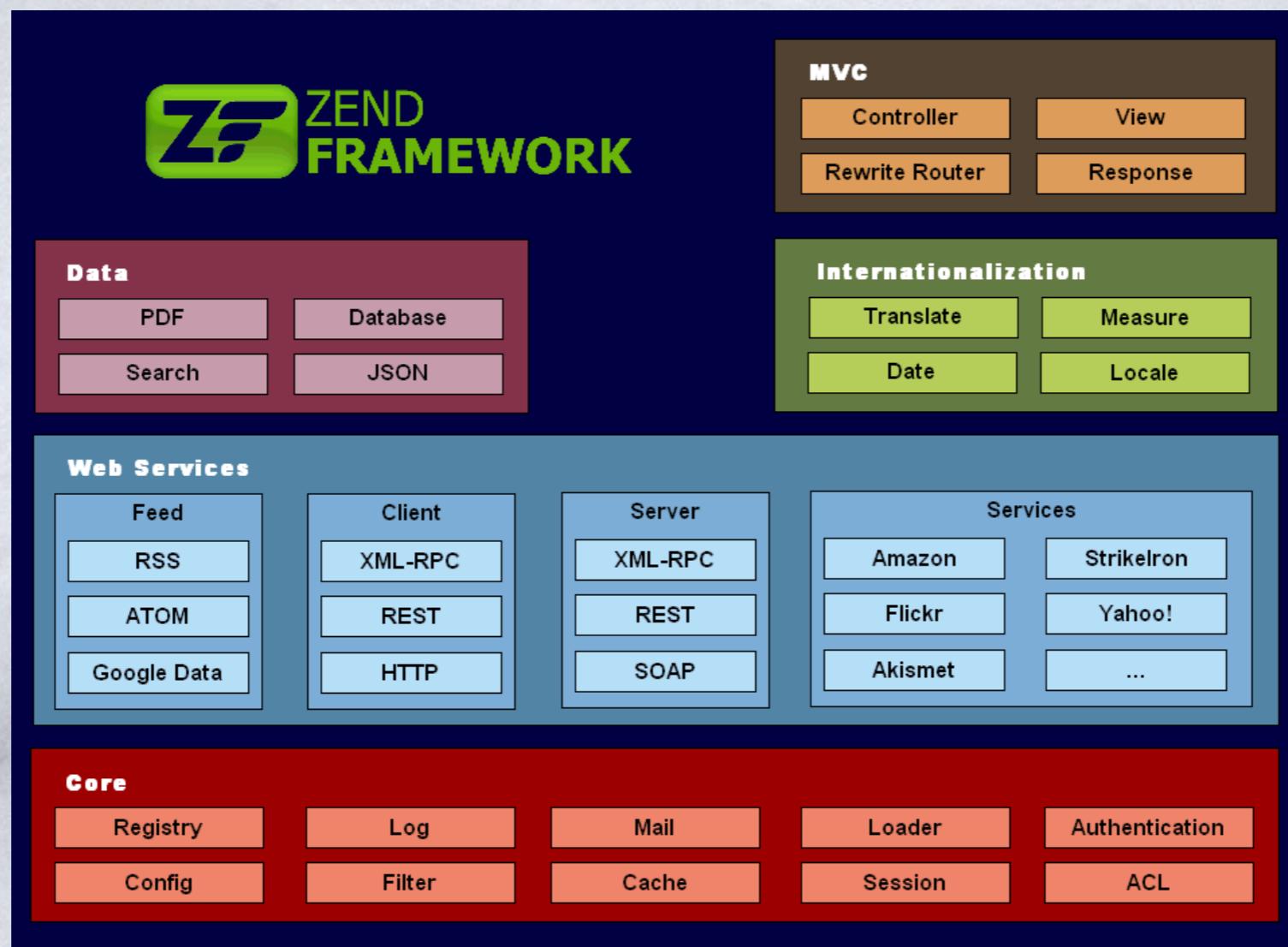
Components in 1.75

Zend_Acl
Zend_Amf
Zend_Auth
Zend_Cache
Zend_Captcha
Zend_Config
Zend_Config_Writer
Zend_Console_Getopt
Zend_Controller
Zend_Currency
Zend_Date
Zend_Db
Zend_Debug
Zend_Dojo
Zend_Dom
Zend_Exception
Zend_Feed
Zend_File
Zend_Filter
Zend_FilterInput
Zend_Form
Zend_Gdata
Zend_Http

Zend_Infocard
Zend_Json
Zend_Layout
Zend_Ldap
Zend_Loader
Zend_Locale
Zend_Log
Zend_Mail
Zend_Measure
Zend_Memory
Zend_Mime
Zend_OpenId
Zend_Paginator
Zend_Pdf
Zend_ProgressBar
Zend_Registry
Zend_Rest
Zend_Search_Lucene
Zend_Server_Reflection
Zend_Akismet
Zend_Amazon
Zend_Audioscrobbler
Zend_Delicious

Zend_Flickr
Zend_Nirvanix
Zend_ReCaptcha
Zend_Simpy
Zend_SlideShare
Zend_Strikelron
Zend_Technorati
Zend_Twitter
Zend_Yahoo
Zend_Session
Zend_Soap
Zend_Test
Zend_Text
Zend_Timesync
Zend_Translate
Zend_Uri
Zend_Validate
Zend_Version
Zend_View
Zend_Wildfire
Zend_XmlRpc
Zend_XConsoleProcessUnix
Zend_XJQuery

Zend Framework Architecture



Thanks!!!

