

# Optimising a Zend Framework application

Rob Allen

PHPUK February 2011

# Rob Allen?

<http://akrabat.com>

@akrabat

“Optimisation is a game of eliminating measurable inefficiencies.

If you can't reliably measure or predict the impact of an optimisation, then how do you know it's a worthwhile optimisation?”

Pádraic Brady

# Measuring performance

# Siege

<http://www.joedog.org/index/siege-home>

```
edit ~/.siegerc  
  verbose = false  
  logging = false or logfile=~/.siege.log  
  concurrent = 10  
  benchmark = true
```

# Running Siege

```
siege -t 30s http://localhost/info.php
```

Time based: `-t NUMx` where `x = S,M or H`

Request based: `-r NUM`

Add `-c NUM` for the number of concurrent users

i.e.

```
$siege -t 30s http://localhost/
```

# Siege output

```
$ siege -t 30s http://localhost/info.php
** SIEGE 2.70
** Preparing 10 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.
Transactions:          25054 hits
Availability:          100.00 %
Elapsed time:           31.98 secs
Data transferred:     1362.59 MB
Response time:         0.01 secs
Transaction rate:    783.43 trans/sec
Throughput:           42.61 MB/sec
Concurrency:           9.97
Successful transactions: 25054
Failed transactions:    0
Longest transaction:   7.68
Shortest transaction:  0.00
```

# Test a basic app

- zf create project
- zf enable layout
- Run siege

**Transaction rate: 79.38 trans/sec**



# Profile with xdebug

<http://www.xdebug.org>

- Install: `pecl install xdebug`
- Output file compatible with:
  - KCachegrind (Linux)
  - WinCacheGrind (Windows)
  - MacCallGrind (OS X)
  - WebGrind (PHP)

# Profile with xdebug

php.ini:

```
xdebug.profiler_enable = 0  
xdebug.profiler_enable_trigger = 1  
xdebug.profiler_output_dir = "/Users/rob/xdebug"
```

Enable per request:

[http://localhost/places/public/?XDEBUG\\_PROFILE=1](http://localhost/places/public/?XDEBUG_PROFILE=1)

(or use the Firefox extension)

# webgrind<sup>v1.0</sup>

profiling in the browser

Show  of 
 in 

  
 Hide PHP functions

/www/phpuk11/zf/public/index.php

cachegrind.out.96486 @ 2011-02-15 06:16:10



453 different functions called in 71 milliseconds (1 runs, 373 shown)

| Function  | Invocation Count | Total Self Cost | Total Inclusive Cost |
|---|------------------|-----------------|----------------------|
| ▶ {main}  | 1                | 2               | 71                   |
| ▶ Zend_Application->run                         | 1                | 0               | 34                   |
| ▶ Zend_Application_Bootstrap_Bootstrap->run     | 1                | 0               | 34                   |
| ▶ Zend_Controller_Front->dispatch               | 1                | 5               | 33                   |
| ▶ Zend_Application->__construct                 | 1                | 1               | 23                   |
| ▶ Zend_Loader_Autoloader::autoload              | 9                | 1               | 21                   |
| ▶ Zend_Loader::loadClass                        | 9                | 1               | 18                   |
| ▶ Zend_Controller_Dispatcher_Standard->dispatch | 1                | 0               | 18                   |
| ▶ Zend_Loader_Autoloader->_autoload             | 9                | 0               | 18                   |
| ▶ Zend_Loader::loadFile                         | 9                | 7               | 16                   |
| ▶ Zend_Application->setOptions                  | 1                | 0               | 15                   |
| ▶ Zend_Application->setBootstrap                | 1                | 0               | 14                   |

# Profile with xhprof

- <http://techportal.ibuildings.com/?p=1527>
- <http://phpadvent.org/2010/profiling-with-xhgui-by-paul-reinheimer>
- Install xhprof
- Set up `append.php` & `prepend.php`
- Profile!
- Sort results by Excl. CPU usage

## Run Report

Run #4d5a1c53bf66d: XHPProf Run (Namespace=myapp)

[View Top Level Run Report](#)

## Tip

Click a function name below to drill down.

## Overall Summary

**Total Incl. Wall Time (microsec):** 90,439 microsecs  
**Total Incl. CPU (microsecs):** 44,069 microsecs  
**Total Incl. MemUse (bytes):** 4,386,856 bytes  
**Total Incl. PeakMemUse (bytes):** 4,426,632 bytes  
**Number of Function Calls:** 1,829

[\[View Full Callgraph\]](#)

 Displaying top 100 functions: Sorted by Incl. Wall Time (microsec) [\[display all\]](#)

| <a href="#">Function Name</a>                             | <a href="#">Calls</a> | <a href="#">Calls%</a> | <a href="#">Incl. Wall Time (microsec)</a> | <a href="#">IWAll%</a> | <a href="#">Excl. Wall Time (microsec)</a> | <a href="#">EWAll%</a> | <a href="#">Incl. CPU (microsecs)</a> | <a href="#">ICpu%</a> | <a href="#">Excl. CPU (microsec)</a> | <a href="#">ECPU</a> |
|---|-----------------------|------------------------|--|------------------------|--|------------------------|---------------------------------------|-----------------------|--------------------------------------|----------------------|
| <a href="#">main()</a>                                    | 1                     | 0.1%                   | 90,439                                     | 100.0%                 | 1,296                                      | 1.4%                   | 44,069                                | 100.0%                | 477                                  | 1.1                  |
| <a href="#">Zend_Application::run</a>                     | 1                     | 0.1%                   | 44,383                                     | 49.1%                  | 21   | 0.0%                   | 21,446                                | 48.7%                 | 9                                    | 0.0                  |
| <a href="#">Zend_Application_Bootstrap_Bootstrap::run</a> | 1                     | 0.1%                   | 44,356                                     | 49.0%                  | 84   | 0.1%                   | 21,433                                | 48.6%                 | 29                                   | 0.1                  |
| <a href="#">Zend_Controller_Front::dispatch</a>           | 1                     | 0.1%                   | 44,063                                     | 48.7%                  | 1,104                                      | 1.2%                   | 21,287                                | 48.3%                 | 485                                  | 1.1                  |
| <a href="#">Zend_Application::_construct</a>              | 1                     | 0.1%                   | 26,694                                     | 29.5%                  | 239  | 0.3%                   | 13,005                                | 29.5%                 | 111                                  | 0.3                  |
| <a href="#">Zend_Loader_Autoloader::autoload</a>          | 6                     | 0.3%                   | 21,942                                     | 24.3%                  | 246  | 0.3%                   | 10,639                                | 24.1%                 | 114                                  | 0.3                  |

**Obvious stuff**

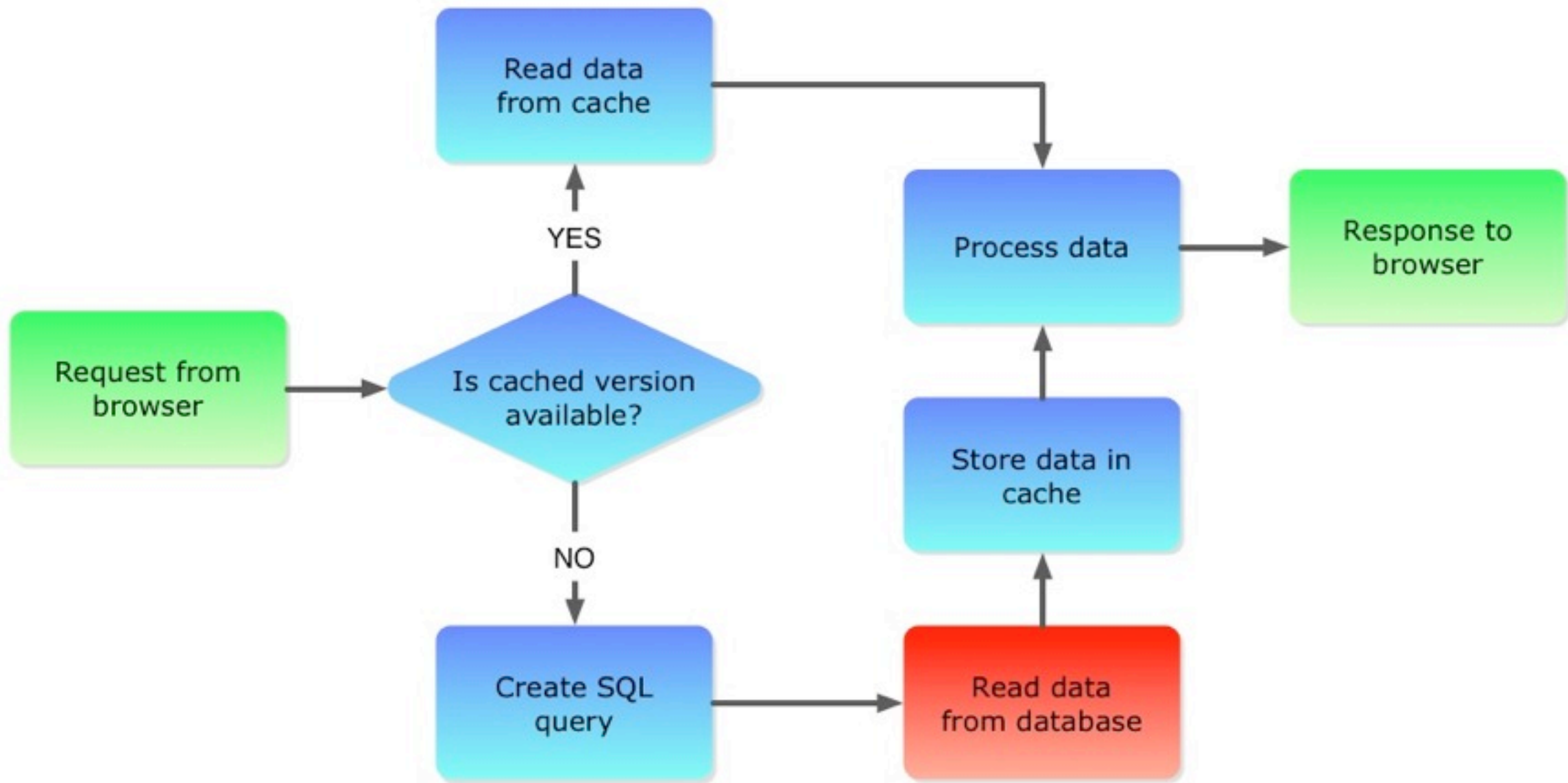
# APC

If you don't have a byte code cache installed, you don't care about performance

**Before:** 79.38 trans/sec

**After:** 311.17 trans/sec

# Cache!





# Page-level caching

Why stop at just the database?

Let's cache the entire page!

**ZF specific**

**Loading files**

# Include path

- Use absolute paths to include files
- Place ZF **first** on include\_path

```
define( 'APPLICATION_PATH',  
    realpath(dirname(__FILE__) . '/../application' ));  
  
set_include_path(implode(PATH_SEPARATOR, array(  
    realpath(APPLICATION_PATH . '/../library'),  
    get_include_path(),  
)));
```

# Use autoload only

- strip all `require_once` calls from the Zend Framework code

```
$ cd library/Zend
$ find . -name '*.php' -not -wholename \
  '*/Loader/Autoloader.php' \
  -not -wholename '*/Application.php' -print0 | \
  xargs -0 sed --regexp-extended --in-place \
  's/(require_once)/\/// \1/g'
```

or for Mac:

```
$ find . -name '*.php' -not -wholename \
  '*/Loader/Autoloader.php' -not -wholename \
  '*/Application.php' -print0 | \
  xargs -0 sed -E -i '.bak' 's/(require_once)/\/// \1/g'
```

# Baseline ZF app

webgrind of /www/phpuk11/zf/public/index.php

http://localhost/webgrind/ Google

webgrind<sup>v1.0</sup>  
profiling in the browser

Show 100% of phpuk11/zf/public/index.php in milliseconds update  
 Hide PHP functions

/www/phpuk11/zf/public/index.php  
cachegrind.out.96486 @ 2011-02-15 06:16:10

453 different functions called in 71 milliseconds (1 runs, 453 shown)

| Function   | Invocation Count | Total Self Cost | Total Inclusive Cost |
|--|------------------|-----------------|----------------------|
| ▶ Zend_Loader::loadFile  | 9                | 7               | 16                   |
| ▶ Zend_Controller_Front->dispatch  | 1                | 5               | 33                   |
| ▶ require_once::/www/phpuk11/zf/library/Zend/Controller/Dispatcher/Interface.php | 1                | 2               | 2                    |
| ▶ Zend_Config_Ini->_processKey   | 26               | 2               | 3                    |
| ▶ require_once::/www/phpuk11/zf/library/Zend/View.php                            | 1                | 2               | 2                    |
| ▶ Zend_Controller_Action_Helper_ViewRenderer->getInflector                       | 3                | 2               | 8                    |
| ▶ {main}   | 1                | 2               | 71                   |
| ▶ Zend_Loader_PluginLoader->load   | 7                | 2               | 4                    |
| ▶ php::call_user_func  | 18               | 1               | 37                   |
| ▶ Zend_Loader_Autoloader::autoload   | 9                | 1               | 21                   |
| ▶ Zend_Loader::loadClass   | 9                | 1               | 18                   |
| ▶ Zend_Filter_Inflector->filter  | 2                | 1               | 3                    |

# Zend\_Loader::loadFile

```
public static function loadFile($filename,
    $dirs = null, $once = false)
{
    self::_securityCheck($filename);
    $incPath = false;
    if (!empty($dirs)&&(is_array($dirs) || is_string($dirs))) {
        if (is_array($dirs)) {
            $dirs = implode(PATH_SEPARATOR, $dirs);
        }
        $incPath = get_include_path();
        set_include_path($dirs . PATH_SEPARATOR . $incPath);
    }
    if ($once) {
        include_once $filename;
    } else {
        include $filename;
    }
    if ($incPath) { set_include_path($incPath); }
    return true;
}
```

# Zend\_Loader::loadFile

Change loadFile() to:

```
public static function loadFile($filename,  
    $dirs = null, $once = false)  
{  
    if (APPLICATION_ENV == 'live') {  
        include $filename;  
        return true;  
    }  
    // continue...
```



# Zend\_Application

# Config file handling

```
// Create application, bootstrap, and run
$application = new Zend_Application(
    APPLICATION_ENV,
    APPLICATION_PATH . '/configs/application.ini'
);
```

# Cache with APC

```
$config = apc_fetch('my_config');
if (!$config instanceof Zend_Config) {
    require_once 'Zend/Config/Ini.php';
    $section = APPLICATION_ENV;
    $filename = APPLICATION_PATH . '/configs/application.ini';
    $config = new Zend_Config_Ini($filename, $section);
    $config = $config->toArray();
    apc_store('my_config', $config, 600);
}

// Create application, bootstrap, and run
$application = new Zend_Application(
    APPLICATION_ENV,
    $config
);
```

**Zend\_Db\_Table**

# Metadata

```
// Fetch metadata from the adapter's describeTable() method
$metadata = $this->_db->describeTable($this->_name,
    $this->_schema);
```

# Cache the Metadata

application.ini:

```
; set up the cache
resources.cachemanager.db.frontend.name = Core
resources.cachemanager.db.frontend
    .options.lifetime = 7200
resources.cachemanager.db.frontend
    .options.automatic_serialization = true
resources.cachemanager.db.backend.name = File
resources.cachemanager.db.backend
    .options.cache_dir = APPLICATION_PATH "../cache/db"

; set up database resource
resources.db.defaultMetadataCache = "db"
```

**Zend\_View**

# View helpers

Zend\_View's helper system:

```
public function __call($name, $args)
{
    // is the helper already loaded?
    $helper = $this->getHelper($name);

    // call the helper method
    return call_user_func_array(
        array($helper, $name),
        $args
    );
}
```



# View helpers

Move common view helpers to your own extension of `Zend_View`

```
// library/App/View.php
class App_View extends Zend_View
{
}
```

```
// configs/application.ini
autoloadernamespaces[] = "App_"
```

```
// application/Bootstrap.php
class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
    protected function _initView()
    {
        $resources = $this->getOption( 'resources' );
        $options = array();
        if (isset($resources[ 'view' ])) { $options = $resources[ 'view' ]; }
        $view = new App_View($options);
        if (isset($options[ 'doctype' ])) {
            $view->doctype()->setDoctype(strtoupper($options[ 'doctype' ]));
            if (isset($options[ 'charset' ]) && $view->doctype()->isHtml5()) {
                $view->headMeta()->setCharset($options[ 'charset' ]);
            }
        }
        if (isset($options[ 'contentType' ])) {
            $view->headMeta()->appendHttpEquiv( 'Content-Type',
                $options[ 'contentType' ] );
        }

        $viewRenderer = new Zend_Controller_Action_Helper_ViewRenderer();
        $viewRenderer->setView($view);
        Zend_Controller_Action_HelperBroker::addHelper($viewRenderer);
        return $view;
    }
}
```

# \$this->url()

```
public function url($urlOptions = array(), $name = null,  
    $reset = false, $encode = true)  
{  
    $fc = Zend_Controller_Front::getInstance()  
    $router = $fc->getRouter();  
  
    return $router->assemble($urlOptions, $name, $reset,  
        $encode);  
}
```

# \$this->escape()

```
public function escape($var)
{
    if (in_array($this->_escape,
        array('htmlspecialchars', 'htmlentities'))) {
        return call_user_func($this->_escape, $var,
            ENT_COMPAT, $this->_encoding);
    }

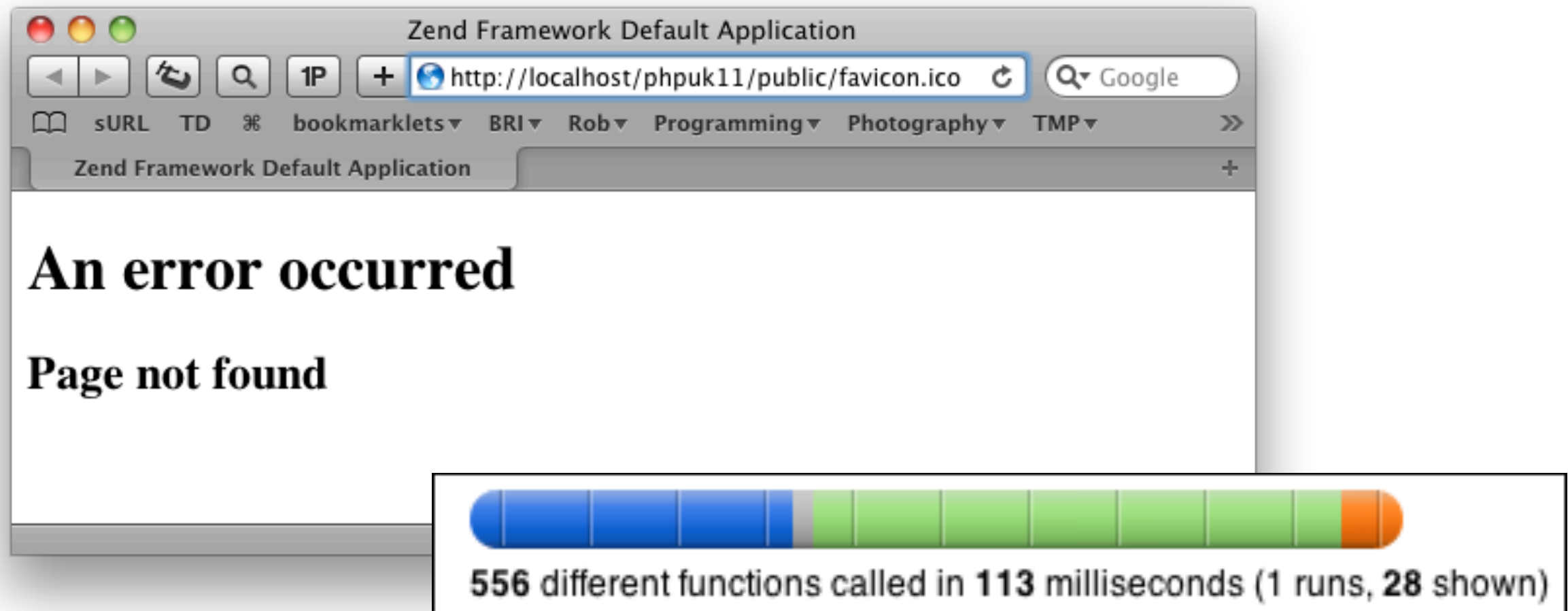
    if (1 == func_num_args()) {
        return call_user_func($this->_escape, $var);
    }
    $args = func_get_args();
    return call_user_func_array($this->_escape, $args);
}
```

# Rewrite

```
public function escape($var)
{
    return htmlspecialchars($var, ENT_COMPAT,
        $this->_encoding);
}
```

Other things to  
consider

# Add a favicon.ico



# The manual

Read the  
Zend Framework Performance Guide

<http://framework.zend.com/manual/en/performance.html>



# Summary

- Use APC (or another byte-code cache)
- Cache the slow stuff!
- Simplify Zend\_Loader in production
- Use Zend\_Db\_Table's metacache
- Move some view helpers to App\_View

**Questions?**

# Thank you

feedback: <http://joind.in/2518>

email: [rob@akrabat.com](mailto:rob@akrabat.com)

twitter: @akrabat