

Zend Framework's Little Known Gems

Matthew Weier O'Phinney
Software Architect
Zend Framework

What is Zend Framework?

Full Stack Framework?

Component Library?

Both.



MVC

Controller

View

Component

Rewrite Router

JSON

Form

Data

PDF

Database

Search

XML Database

Internationalization

Locale

Translate

Measure

Date

Calendar

Currency

Web Services

Feed

RSS

ATOM

Google Data

Client

XML-RPC

REST

HTTP

Server

XML-RPC

REST

SOAP

Services

Amazon

Simpy

Flickr

Yahoo!

del.icio.us

...

Core

Registry

Log

Mail

URI

Authentication

Config

Filter

Cache

Session

ACL

**Why loosely
coupled matters**

PHP has been around for > 10 years

PHP has a low barrier to entry

Many successful PHP applications have
been written in that time frame

- Many of these applications suck
- from an architectural standpoint:
- Many PHP developers never learn good programming skills
- Many PHP developers never learn design patterns
- Many of these applications do not follow MVC paradigms

Refactoring sucks

Loosely coupled libraries allow you
to refactor incrementally

Little Known Gems

Zend_Acl

Basic ACLs

```
$acl = new Zend_Acl();
$acl->addRole(new Zend_Acl_Role('guest'))
    ->addRole(new Zend_Acl_Role('admin'), 'guest')
    ->add(new Zend_Acl_Resource('wiki'))
    ->allow('guest', 'wiki',
          array('view', 'list', 'feed'))
    ->allow('admin', 'wiki');

if ($acl->isAllowed($role, 'wiki', 'list')) {
    // woohoo!
}
```

Custom ACL role

```
class Wiki_Role_Guest
    implements Zend_Acl_Role_Interface
{
    public function getRoleId()
    {
        return 'guest';
    }
}
```

Mixing ACL interfaces into models

```
class Wiki_Guest
    implements Zend_Acl_Role_Interface
{
    public function getRoleId()
    {
        return 'guest';
    }

    public function save()
    {
        $this->dataSource->save (
            $this->toArray()
        );
    }

    // ...
}
```

Zend_Cache

Creating a cache object

```
$cache = Zend_Cache::factory(  
    'Core', // cache object frontend  
    'File', // caching backend  
    array( // frontend options  
        'lifetime' => 3600, // seconds  
        'automatic_serialization' => true,  
    ),  
    array( // backend options  
        'cache_dir' => APPLICATION_PATH . '/cache',  
    )  
);
```

Using a cache

```
if (!$data = $cache->load('foo')) {  
    $data = $someLongRunning->process();  
    $cache->save($data, 'foo');  
}  
$view->data = $data;
```

Using tags with caching

```
if (!$data = $cache->load('foo')) {  
    $data = $someLongRunning->process();  
  
    // Use tags to group records for later cleaning  
    $cache->save($data, 'foo', array('bar', 'baz'));  
}  
  
// Clean all records tagged 'bar':  
$cache->clean(  
    Zend_Cache::CLEANING_MODE_MATCHING_TAG,  
    array('bar')  
);
```

Zend_Config

Several configuration backends

```
// Load 'testing' section of INI configuration:
$config = new Zend_Config_Ini($fileName, 'testing');

// Load 'testing' section of XML configuration:
$config = new Zend_Config_Xml($fileName, 'testing');

// Load 'testing' configuration via PHP array:
$config = new
Zend_Config(file_get_contents('testing.conf.php'));
```

Sample INI configuration

[production]

```
app.name = "Foo!"  
db.adapter = "Pdo_Mysql"  
db.params.username = "foo"  
db.params.password = "bar"  
db.params.dbname = "foodb"  
db.params.host = "127.0.0.1"
```

[testing : production]

```
db.adapter = "Pdo_Sqlite"  
db.params.dbname = APPLICATION_PATH "/data/test.db"
```

Sample XML configuration

```
<?xml version="1.0" ?>
<config>
  <production>
    <app><name>Foo!</name></app>
    <db>
      <adapter>Pdo_Mysql</adapter>
      <params>
        <username>foo</username>
        <password>bar</password>
        <dbname>foodb</dbname>
        <host>127.0.0.1</host>
      </params>
    </db>
  </production>
  <testing extends="production">
    <db>
      <adapter>Pdo_Sqlite</adapter>
      <params>
        <dbname>/data/test.db</dbname>
      </params>
    </db>
  </testing>
</config>
```

Sample PHP configuration

```
$production =  
file_get_contents('production.conf.php');  
  
$config = array(  
    'db' => array(  
        'adapter' => 'Pdo_Sqlite',  
        'params' => array(  
            'dbname' => APP_PATH . '/data/test.db',  
        ),  
    ),  
);  
$config = $production + $config;  
return $config;
```

Zend_Dom_Query

Simple link screen scraper

```
$dom = new Zend_Dom_Query($html);  
$nodes = $dom->query('#content a');  
if (0 > count($nodes)) {  
    foreach ($nodes as $node) {  
        // do something with link DOM node  
    }  
}
```

Zend_Dom_Query powers Zend_Test_PHPUnit assertions

```
class FooControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    public function testHomePageHasAtLeast3Links ()
    {
        $this->dispatch ('/');
        $this->assertQueryCountMin (
            '#content a', 3
        );
    }
}
```

Zend_Json_Server and Zend_XmlRpc_Server

JSON-RPC server with Service Mapping Description

```
$server = new Zend_Json_Server();
$server->setClass('My_FormHandler');

// Return the Service Mapping Description, if requested
if ('GET' == strtoupper($_SERVER['REQUEST_METHOD'])) {
    $server->setTarget('/jsonrpc')
        ->setEnvelope(Zend_Json_Server_Smd::ENV_JSONRPC_2);

    // Grab the SMD
    $smd = $server->getServiceMap();

    // Return the SMD to the client
    header('Content-Type: application/json');
    echo $smd;
    return;
}

// Handle the request
$server->handle();
```

XML-RPC server

```
$server = new Zend_XmlRpc_Server();  
$server->setClass('My_FormHandler');  
echo $server->handle();
```

Zend_Loader_PluginLoader

Basic plugin loading

```
$loader = new Zend_Loader_PluginLoader(array(  
    'Foo_Bar' => 'path/to/foo/',  
    'Bar_Baz' => 'path/to/bar/',  
));
```

```
// Load 'Bat' plugin:  
$class = $loader->load('bat');  
$bat    = new $class(); // Foo_Bar_Bat or  
                        // Bar_Baz_Bat
```

Use inflection for better plugin loading

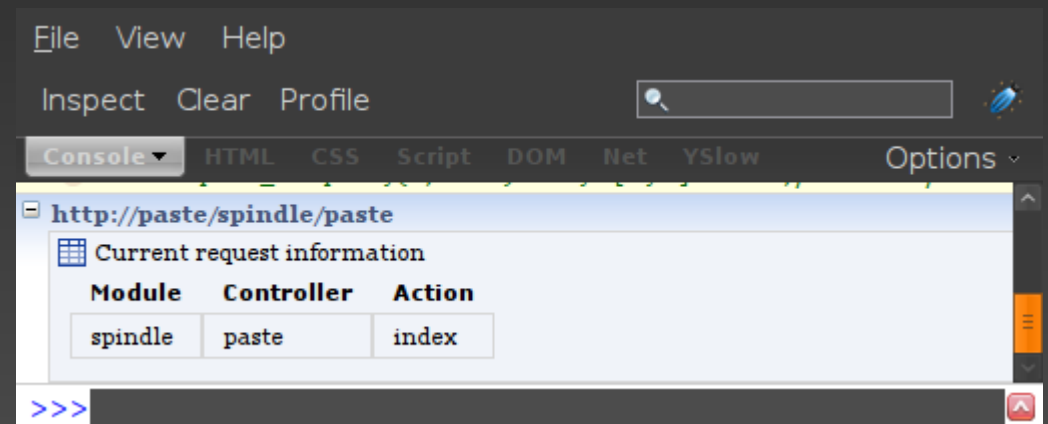
```
class My_Loader extends Zend_Loader_PluginLoader
{
    public function inflectName($name)
    {
        $inflector = new Zend_Filter_Inflector(':name');
        $inflector->setRules(array(
            'name' => array(
                'StringToLower',
                'Word_DashToCamelCase'
            ),
        ));
        return $inflector->filter(array(
            'name' => $name
        ));
    }

    public function load($name)
    {
        $name = $this->inflectName($name);
        return parent::load($name);
    }
}
```

Zend_Log_Writer_Firebug

Log to FireBug

```
$writer = new Zend_Log_Writer_Firebug();  
$log     = new Zend_Log($writer);  
  
$channel = Zend_Wildfire_Channel_Headers::getInstance();  
$channel->setRequest(new Zend_Controller_Request_Http())  
         ->setResponse(new Zend_Controller_Response_Http());  
  
ob_start();  
  
// ...  
$log->info('Firebug integration!');  
  
// flush log and send headers...  
$channel->flush();  
$channel->getResponse()->sendHeaders();
```



Akismet, ReCaptcha,
and Captcha, oh my!

Akismet

```
$akismet = new Zend_Service_Akismet($apiKey, $url);  
if (!$akismet->verifyKey()) {  
    throw new Exception('Invalid Akismet key');  
}  
  
$data = array(  
    'user_ip'           => $_SERVER['REMOTE_ADDR'],  
    'user_agent'       => $_SERVER['HTTP_USER_AGENT'],  
    'comment_content'  => $submittedContent,  
    'comment_type'     => 'email',  
    'comment_author_email' => $submittedEmail,  
);  
  
if ($akismet->isSpam($data)) {  
    $log->info('Spam trapped: '  
        . var_export($data, 1));  
}
```

ReCaptcha

```
$captcha = new Zend_Service_ReCaptcha(  
    $pubKey, $privKey  
);  
  
// Render it:  
$captcha->getHTML();  
  
// Verify it:  
$result = $captcha->verify($challenge,  
$response);  
if (!$result->isValid()) {  
    // slap their wrist  
}
```



ought mockup

Type the two words:

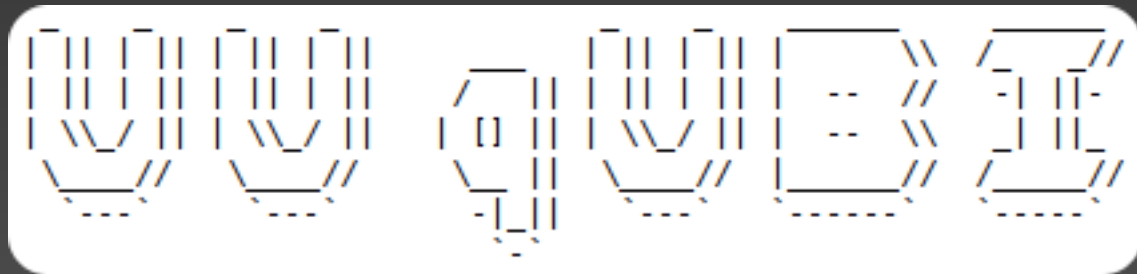
reCAPTCHA™
stop spam.
read books.

Submit

The words above come from scanned books.
By typing them, you help to digitize old texts.

Captcha

```
$captcha = new Zend_Captcha_Figlet(array(  
    'name'      => 'foo',  
    'wordLen'   => 6,      // Length of captcha  
    'timeout'   => 300,   // Validation dur.  
));  
$id = $captcha->generate();  
  
// Render it:  
echo $captcha->render($view);  
  
// Verify it:  
if (!$captcha->isValid($data['foo'], $data))  
{  
    // slap their wrist  
}
```



Questions?

Thank you.