

**Matthew Weier O'Phinney**

Project Lead

**Ralph Schindler**

Software Engineer

# Introducing Zend Framework 2.0

3 November 2010

- **Summer 2005:**

- *Coding begins on Zend Framework, with a handful of cherry-picked partners.*

- **Fall 2005:**

- *First annual **ZendCon***
- *Andi and Zeev announce the PHP Community Process, which includes involvement in the Eclipse Foundation, a relaunch of Zend's Developer Zone, and **Zend Framework**.*

- **March 2006:**

- *Zend Framework **0.1.0** released*
- *Project opened up to the public for contributions; contributions require a Contributor License Agreement.*

## • July 2007:

- *Zend Framework 1.0.0 released*
- *Project includes MVC stack, Table and Row Data Gateways, loads of Service APIs, Authentication and Authorization layers, service providers, and more.*
- *Still largely deemed a work in progress.*

- **March 2008:**

- *Zend Framework 1.5.0 released*
- *Includes Zend\_Form and Zend\_Layout, and many Zend\_View enhancements.*

- **September 2008:**

- *Zend Framework 1.6.0 released*
- *Includes Dojo Toolkit integration, functional test utilities.*

- **November 2008:**
  - *Zend Framework 1.7.0 released*
  - *Includes AMF support, performance improvements.*



## • April 2009:

- *Zend Framework 1.8.0 released*
- *Includes Zend\_Application, Zend\_Tool.*
- *First release widely seen as providing a full stack.*

- **July 2009:**

- *Zend Framework 1.9.0 released*
- *Includes reworked feed component, Zend\_Feed\_Reader, and tons of community-driven enhancements.*

- **October 2009:**

- *First monthly community bug hunts launched.*

- **January 2010:**

- *Zend Framework 1.10.0 released*
- *Includes Zend\_Feed\_Writer, re-organized documentation, and community-driven improvements.*

- **February 2010:**
  - *Development on Zend Framework **2.0** begins*

- **June 2010:**

- *Formation of the Community Review Team*

- **November 2010:**
  - *Zend Framework 1.11.0 released*
  - *Includes **mobile** support and **SimpleCloud API**.*

- **The future?**

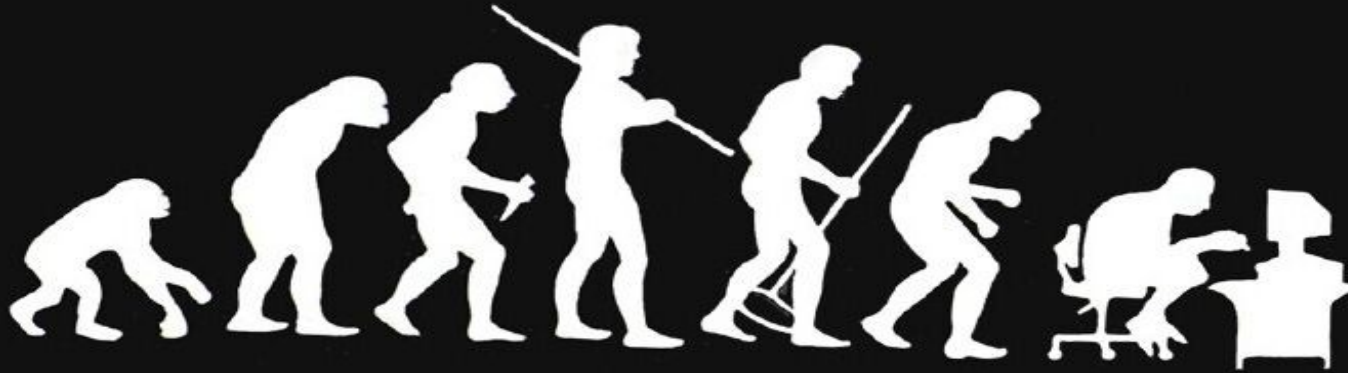


# Revolution?



JOIN, or DIE.

# Evolution.



Something, somewhere went terribly wrong

# Incremental Improvements



- Autoload only (strip `require_once` calls)
- Conversion to PHP namespaces
- Refactor and standardize exceptions usage
- Refactor and consistently implement a single plugin loading strategy
- Refactor to use PHP 5.3-specific paradigms where they fit

**Rewrite only  
where it makes  
sense**



- Difficult to get the underlying connection and share it between instances or different classes.
- Difficult to get schema metadata in a consistent fashion.
- Difficult to extend.
- Difficult to add pre/post tasks.

- Black-box design != testable
- Namespace storage incompatible with `$_SESSION`
- Many incompatibilities with `ext/session`



```
use Zend\Session\SessionManager,  
    Zend\Session\Container as SessionContainer;  
  
$manager = new SessionManager(array(  
    'class' => 'My\Custom\SessionConfiguration',  
    'storage' => 'My\Custom\SessionStorage',  
));  
$container = new SessionContainer('Foo', $manager);  
$container['somekey'] = 'somevalue';  
$container->setExpirationHops(2);
```

- Static access and chain usage were mixed in the same object
- Did not use common plugin loading methodology



```
namespace Zend\Validator;

if (StaticValidator::execute($value, 'int')) {
    //passed validation
}

$chain = new ValidatorChain();
$chain->addValidator(new Int(), true)
    ->addValidator(new GreaterThan(10));

if ($chain->isValid($value)) {
    //passed validation
}
```

P A R E N T A L  
A D V I S O R Y  
E X P L I C I T C O N T E N T

**Favor the Explicit**

Okay, not *that* kind of explicit...



**Magic is  
sometimes  
too arcane**

```

echo $this->headLink()->appendStylesheet('foo.css');
/**
 * Hits Zend_View::__call()
 * Calls Zend_View::getHelper()
 * Calls Zend_View::_getPlugin()
 * Calls Zend_Loader_PluginLoader::load()
 * Calls Zend_Loader::isReadable()
 * Calls call_user_func (hits autoloader...)
 * which calls Zend_Loader::loadClass
 * which calls Zend_Loader::loadFile
 * which calls include_once
 * Instantiates helper
 * Calls method on helper via call_user_func_array()
 * Returns helper instance
 * Call method on instance (hits __call...))
 */

```



**(hidden) automation  
makes learning hard**

```
class FooController
    extends Zend_Controller_Action
{
    public function someAction()
    {
        $this->view->foo = 'bar';
    }
}
```

**Where is  
this defined?**

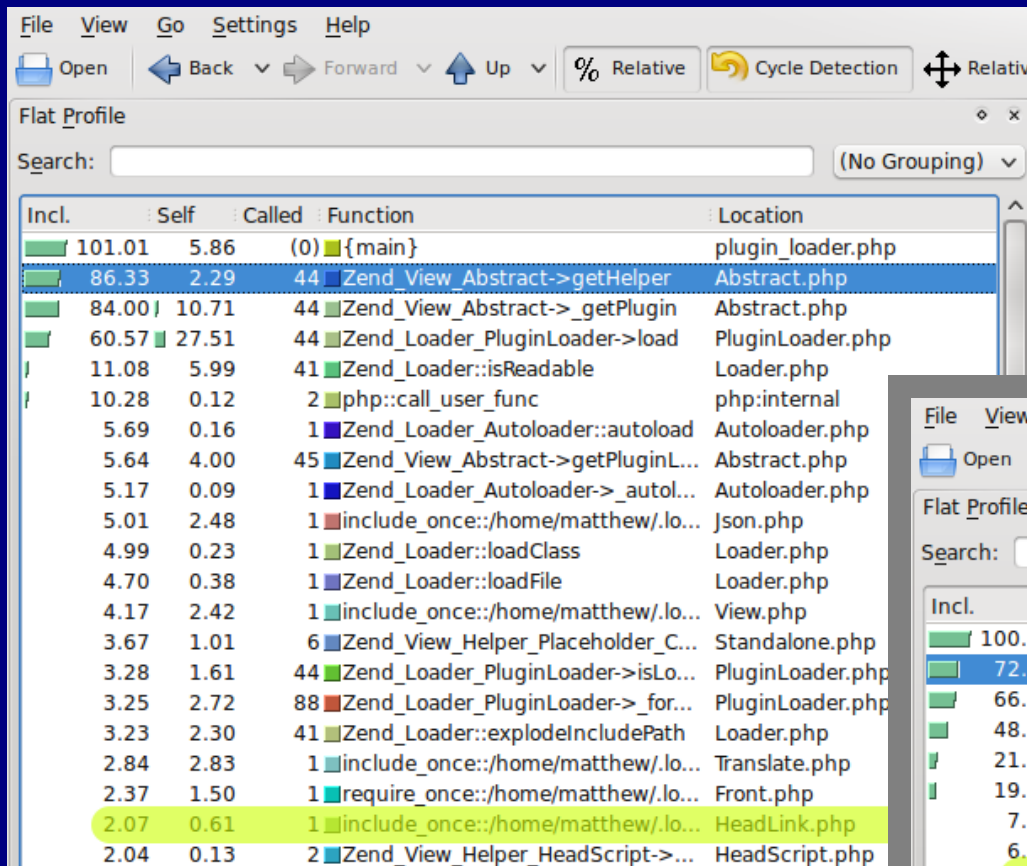
**When is  
it rendered?**

**What about  
layouts?**

# Explicit code is... easier to *understand*

```
$this->broker('head_link')  
->appendStylesheet('foo.css');  
/**  
 * Hits PhpRenderer::broker()  
 * Calls HelperBroker::load()  
 * Calls HelperLoader::load()  
 * Hits autoloader  
 * which simply does an include_once  
 * Instantiates helper  
 * Calls method on helper  
 */
```

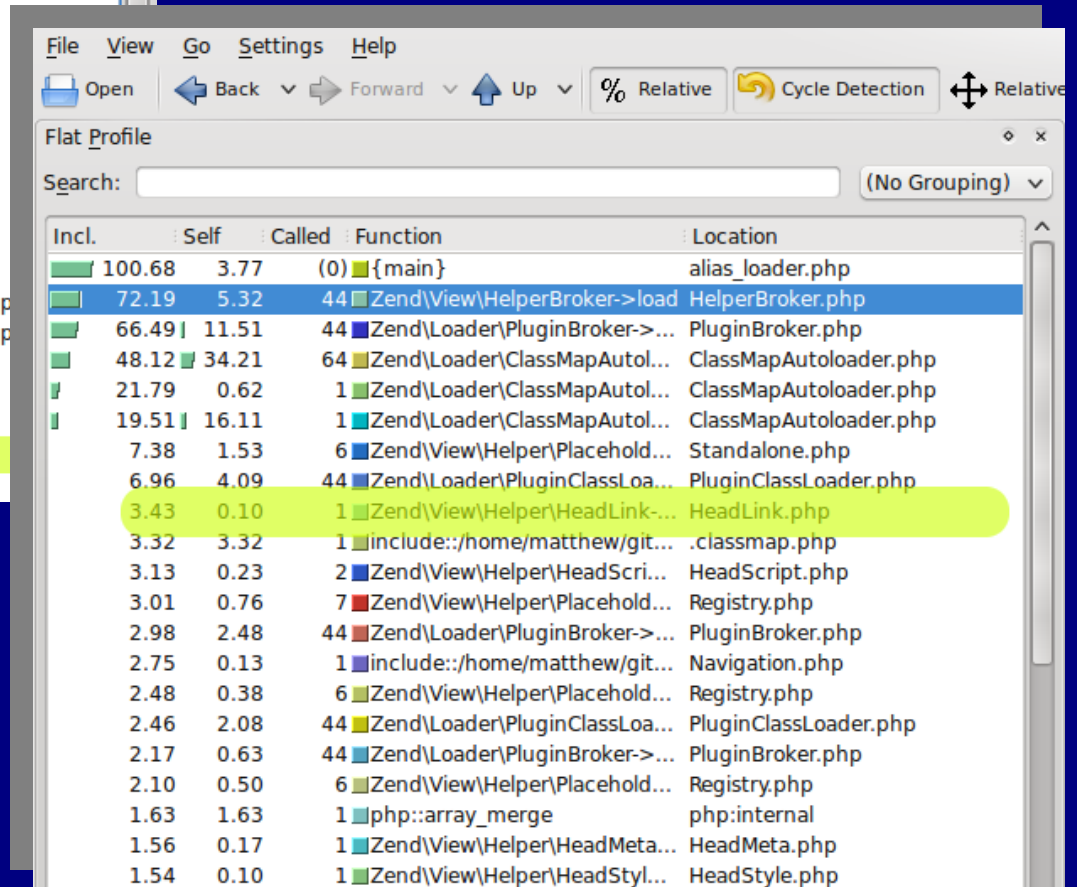
# Explicit code is... easier to *profile*



Flat Profile

Search: (No Grouping)

Incl.	Self	Called	Function	Location
101.01	5.86	(0)	{main}	plugin_loader.php
86.33	2.29	44	Zend_View_Abstract->getHelper	Abstract.php
84.00	10.71	44	Zend_View_Abstract->getPlugin	Abstract.php
60.57	27.51	44	Zend_Loader_PluginLoader->load	PluginLoader.php
11.08	5.99	41	Zend_Loader::isReadable	Loader.php
10.28	0.12	2	php::call_user_func	php:internal
5.69	0.16	1	Zend_Loader_Autoloader::autoload	Autoloader.php
5.64	4.00	45	Zend_View_Abstract->getPluginL...	Abstract.php
5.17	0.09	1	Zend_Loader_Autoloader->_autol...	Autoloader.php
5.01	2.48	1	include_once::/home/matthew/.lo...	Json.php
4.99	0.23	1	Zend_Loader::loadClass	Loader.php
4.70	0.38	1	Zend_Loader::loadFile	Loader.php
4.17	2.42	1	include_once::/home/matthew/.lo...	View.php
3.67	1.01	6	Zend_View_Helper_Placeholder_C...	Standalone.php
3.28	1.61	44	Zend_Loader_PluginLoader->isLo...	PluginLoader.php
3.25	2.72	88	Zend_Loader_PluginLoader->_for...	PluginLoader.php
3.23	2.30	41	Zend_Loader::explodeIncludePath	Loader.php
2.84	2.83	1	include_once::/home/matthew/.lo...	Translate.php
2.37	1.50	1	require_once::/home/matthew/.lo...	Front.php
2.07	0.61	1	include_once::/home/matthew/.lo...	HeadLink.php
2.04	0.13	2	Zend_View_Helper_HeadScript->...	HeadScript.php



Flat Profile

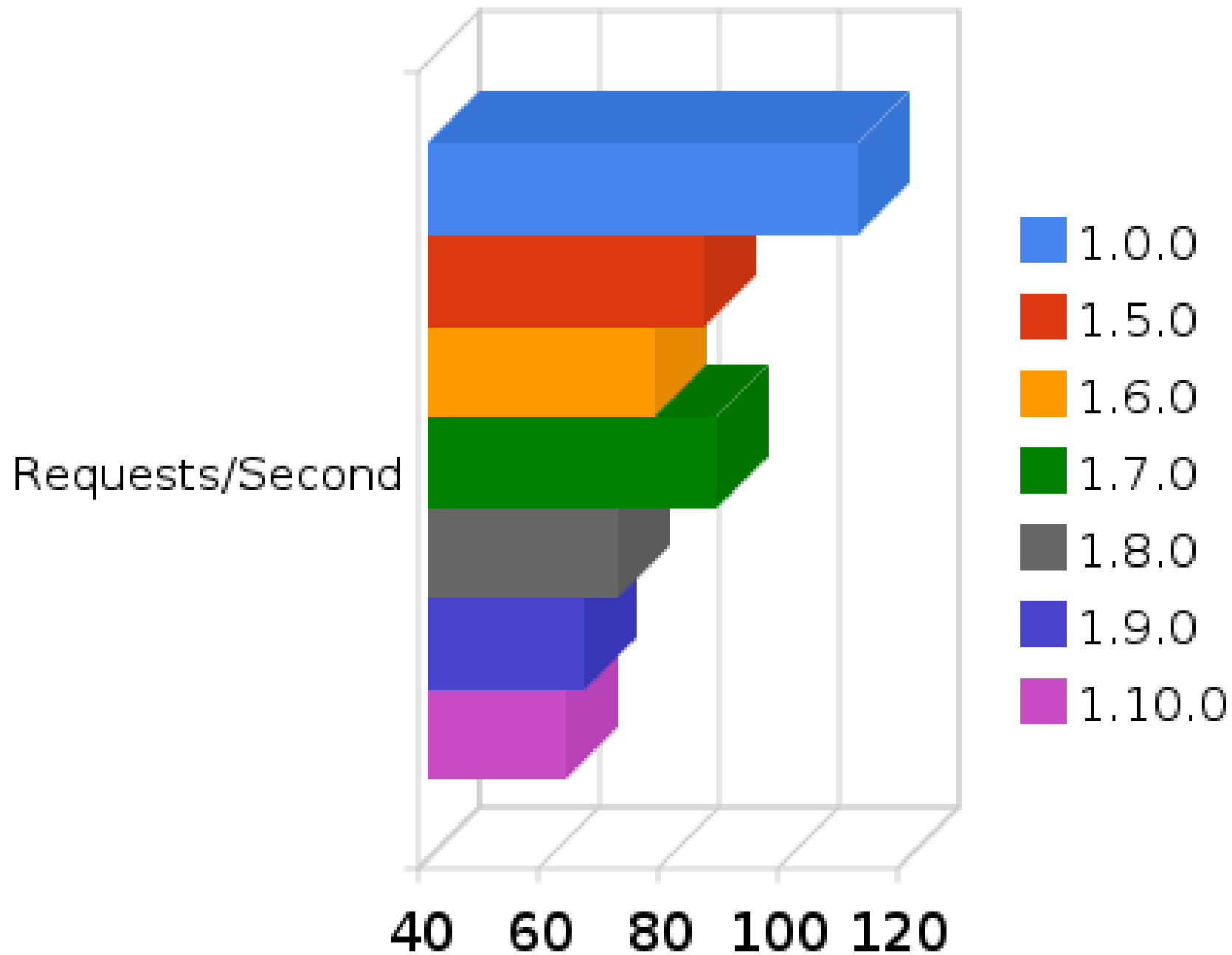
Search: (No Grouping)

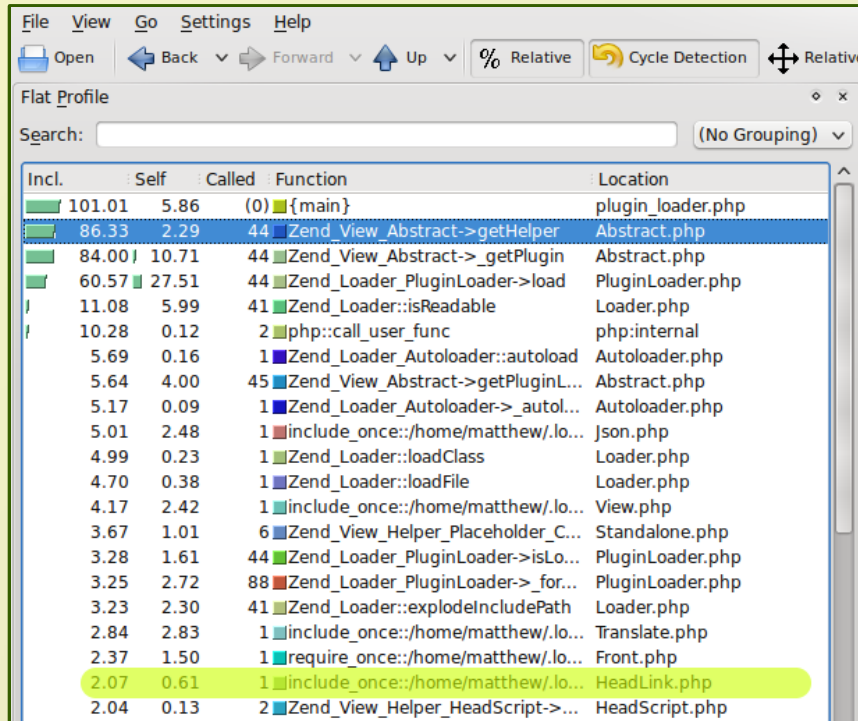
Incl.	Self	Called	Function	Location
100.68	3.77	(0)	{main}	alias_loader.php
72.19	5.32	44	Zend\View\HelperBroker->load	HelperBroker.php
66.49	11.51	44	Zend\Loader\PluginBroker->...	PluginBroker.php
48.12	34.21	64	Zend\Loader\ClassMapAutol...	ClassMapAutoloader.php
21.79	0.62	1	Zend\Loader\ClassMapAutol...	ClassMapAutoloader.php
19.51	16.11	1	Zend\Loader\ClassMapAutol...	ClassMapAutoloader.php
7.38	1.53	6	Zend\View\Helper\Placehold...	Standalone.php
6.96	4.09	44	Zend\Loader\PluginClassLoa...	PluginClassLoader.php
3.43	0.10	1	Zend\View\Helper\HeadLink-...	HeadLink.php
3.32	3.32	1	include::/home/matthew/git...	.classmap.php
3.13	0.23	2	Zend\View\Helper\HeadScri...	HeadScript.php
3.01	0.76	7	Zend\View\Helper\Placehold...	Registry.php
2.98	2.48	44	Zend\Loader\PluginBroker->...	PluginBroker.php
2.75	0.13	1	include::/home/matthew/git...	Navigation.php
2.48	0.38	6	Zend\View\Helper\Placehold...	Registry.php
2.46	2.08	44	Zend\Loader\PluginClassLoa...	PluginClassLoader.php
2.17	0.63	44	Zend\Loader\PluginBroker->...	PluginBroker.php
2.10	0.50	6	Zend\View\Helper\Placehold...	Registry.php
1.63	1.63	1	php::array_merge	php:internal
1.56	0.17	1	Zend\View\Helper\HeadMeta...	HeadMeta.php
1.54	0.10	1	Zend\View\Helper\HeadStyl...	HeadStyle.php





**Optimize for  
performance**





The screenshot shows the Zend Profiler's Flat Profile view. The table lists functions with their inclusion, self, and called execution times, along with the function name and location. The 'plugin\_loader.php' file is highlighted in blue, indicating it is the selected profile.

Incl.	Self	Called	Function	Location
101.01	5.86	(0)	{main}	plugin_loader.php
86.33	2.29	44	Zend_View_Abstract->getHelper	Abstract.php
84.00	10.71	44	Zend_View_Abstract->_getPlugin	Abstract.php
60.57	27.51	44	Zend_Loader_PluginLoader->load	PluginLoader.php
11.08	5.99	41	Zend_Loader::isReadable	Loader.php
10.28	0.12	2	php::call_user_func	php:internal
5.69	0.16	1	Zend_Loader_Autoloader::autoload	Autoloader.php
5.64	4.00	45	Zend_View_Abstract->getPluginL...	Abstract.php
5.17	0.09	1	Zend_Loader_Autoloader->_autol...	Autoloader.php
5.01	2.48	1	include_once::/home/matthew/.lo...	Json.php
4.99	0.23	1	Zend_Loader::loadClass	Loader.php
4.70	0.38	1	Zend_Loader::loadFile	Loader.php
4.17	2.42	1	include_once::/home/matthew/.lo...	View.php
3.67	1.01	6	Zend_View_Helper_Placeholder_C...	Standalone.php
3.28	1.61	44	Zend_Loader_PluginLoader->isLo...	PluginLoader.php
3.25	2.72	88	Zend_Loader_PluginLoader->_for...	PluginLoader.php
3.23	2.30	41	Zend_Loader::explodeIncludePath	Loader.php
2.84	2.83	1	include_once::/home/matthew/.lo...	Translate.php
2.37	1.50	1	require_once::/home/matthew/.lo...	Front.php
2.07	0.61	1	include_once::/home/matthew/.lo...	HeadLink.php
2.04	0.13	2	Zend_View_Helper_HeadScript->...	HeadScript.php

- Profile to determine where the pain points are
- Look for big gains primarily
- Fix them

- Class Maps are fastest
  - But they require maintenance
- Using the `include_path` is slow
  - Not using it requires maintenance
- Using the `include_path` is most flexible
  - But slower than alternatives

- Ship a ClassMapAutoloader by default
  - Class maps for full ZF library, and per-component
  - Tools for generating class maps
  - Ultimate in performance
- StandardAutoloader requires namespace/path pairs
  - Flexibility *and* performance during development
- StandardAutoloader can act as a fallback autoloader
  - Flexibility at the cost of performance

```
// .classmap.php
```

```
return array(  
    'Foo\SomeController' => __DIR__ .  
        '/foo/controllers/SomeController.php',  
    'Foo\Model\Bar' => __DIR__ . '/foo/models/Bar.php',  
);
```

```
// ClassMapAutoloader
```

```
require_once 'Zend/Loader/ClassMapAutoloader.php';
```

```
$loader = new Zend\Loader\ClassMapAutoloader(  
    './.classmap.php');
```

```
$loader->register();
```

```
$bar = new Foo\Model\Bar();
```

```
// StandardAutoloader
// StandardAutoloader
require_once 'Zend/Loader/StandardAutoloader.php';
$loader = new Zend\Loader\StandardAutoloader(array(
    'namespaces' => array(
        'Foo' => __DIR__ . '/library/Foo'),
    ));
$loader->register();

$bar = new Foo\Model\Bar();
```

# Plugin Loading: Problems



- Path stack-based autoloading is *sloooooow*
- Prefix paths are hard to grasp
  - Particularly when coupled with stacks
- Overriding the paths without propagating paths is hard
- Case sensitivity becomes an issue



# Plugin Loading: Problems



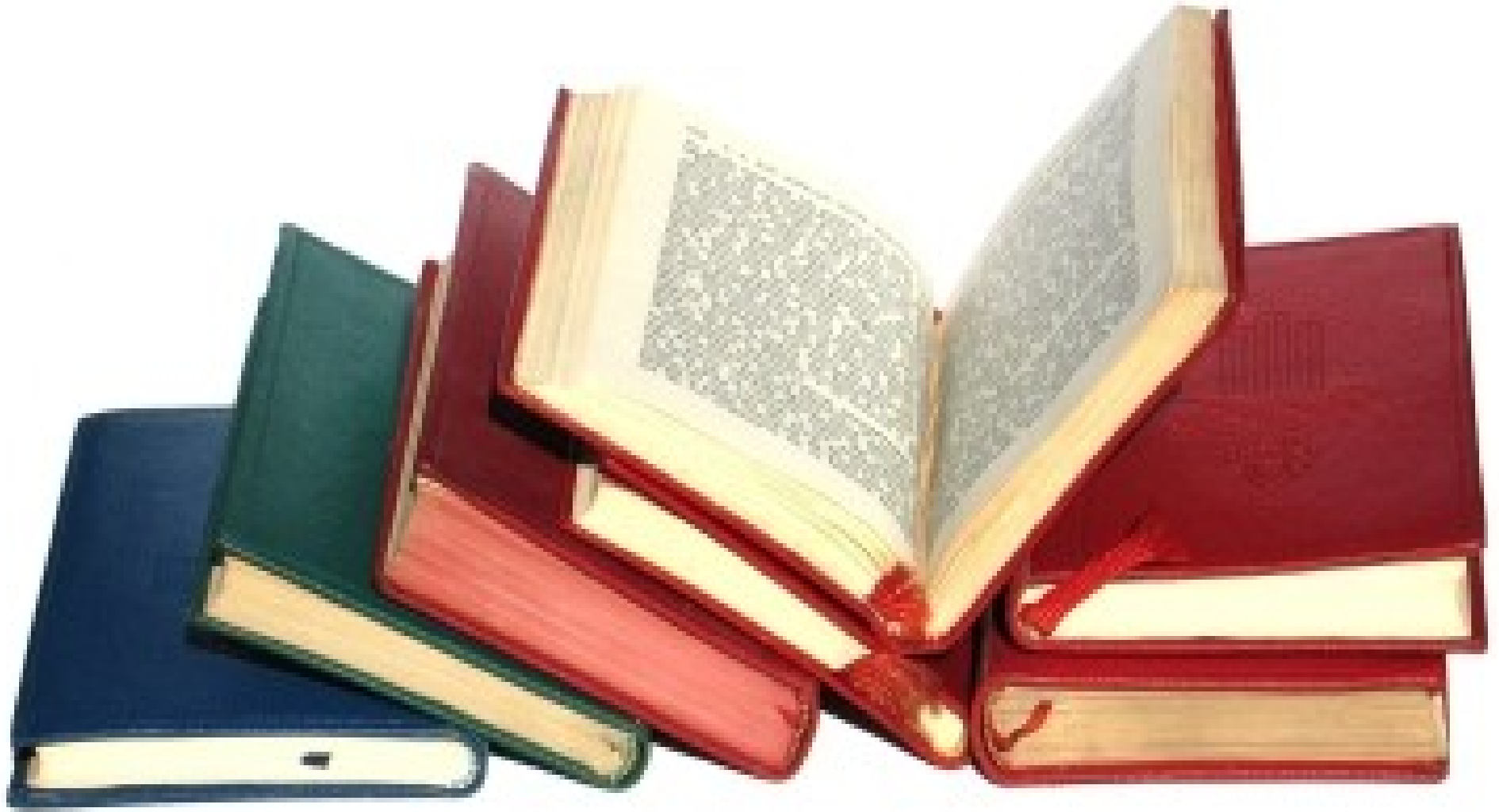
- Current solution only solves the class loading aspect of plugin loading
- Instantiation happens differently per component
- Persistence happens differently per component

- Class alias based maps by default
- Loaders are coupled with **brokers**
  - Handle instantiation, including arguments
  - Act as registry
- Allows attaching single broker to many objects

```
/* class loader */  
namespace My\Component;  
use Zend\Loader\PluginClassLoader,  
    Zend\Loader\PluginBroker;  
  
class ComponentLoader extends PluginClassLoader  
{  
    protected $plugins = array(  
        'foo' => 'My\Component\Foo',  
        'foo_bar' => 'My\Component\FooBar',  
    );  
}
```

```
/* class broker */  
namespace My\Component;  
use Zend\Loader\PluginClassLoader,  
    Zend\Loader\PluginBroker;  
  
class ComponentBroker extends PluginBroker  
{  
    protected $defaultClassLoader =  
        'My\Component\PluginClassLoader';  
  
    protected function validatePlugin($plugin)  
    {  
        if (!$plugin instanceof Adapter) {  
            throw new Exception\RuntimeException();  
        }  
        return true;  
    }  
}
```

```
/* factory */  
namespace My\Component;  
  
class Factory  
{  
    /*Not shown: setBroker() and broker() methods */  
  
    public function get($adapter, array $options)  
    {  
        return $this->broker()  
            ->load($adapter, $options);  
    }  
}
```



**Ease the learning curve**

# Common Documentation Complaints



- Options are (often) not documented.
- Available functionality (typically, methods) is not presented.
- Inconsistent structure between documentation of different components.
- Examples do not show common use cases, only using the component individually.
- No examples showing complete application development.

# Common Learning Problems



- Magic (“\_\_” methods) is hard to learn (*just ask Harry Potter*).
- Uncertainty where and when to extend or implement extension points – *and how to get ZF to use them*.
- Some patterns are non-obvious in usage (*Zend\_Form decorators...*).

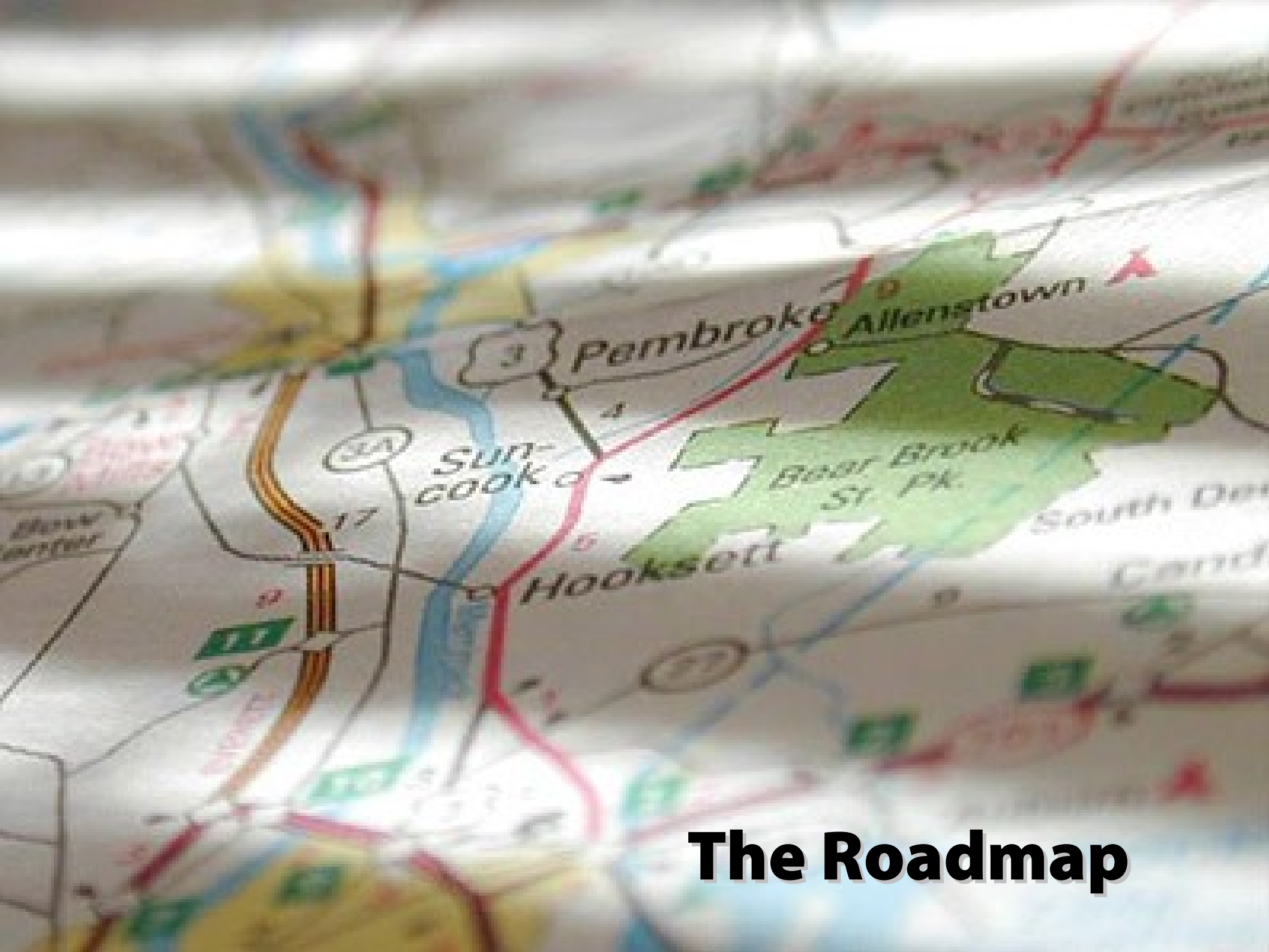


## • Coding:

- Refactor where usage patterns differ from design
- Reduce number of magic calls
- Refactor for consistency
- Refactor unclear APIs

- **Documentation Standards:**

- Introduction
- Quick Start
- Options
- Methods
- Examples



# The Roadmap

# Completed Milestones



- Migration to Git for ZF2 development
- Stripping of `require_once` calls
- Migration to PHP 5.3 namespaces
  - *Including SPL additions, Session rewrite, and addition of SignalSlot*
- Autoloading and plugin loading/brokering
  - *Including View rewrite*
- Exceptions
- *In fact, we've just released a new development milestone snapshot!*

- **Zend Framework 2.0.0dev2:**
  - <http://bit.ly/zf2dev2>

# Remaining Milestones



- MVC Refactoring/Rewrite
- Internationalization and Localization
- Testing
- Documentation
- Packaging
- Migration tools

**How  
YOU  
can  
help**



- **ZF2 wiki:**  
<http://bit.ly/zf2wiki>
- **zf-contributors mailing list:**  
[zf-contributors-subscribe@lists.zend.com](mailto:zf-contributors-subscribe@lists.zend.com)
- **IRC:**  
#zftalk.dev on Freenode



- **Git guide:**  
<http://bit.ly/zf2gitguide>
- **GitHub:**  
<http://github.com/zendframework/zf2>
- **Official repo:**  
<git://git.zendframework.com/zf.git>  
<http://git.zendframework.com/>
- *You still need to sign a CLA!*



**<http://framework.zend.com>**

**Feedback: <http://joind.in/2287>**

**Twitter: [weierophinney](#),  
[ralphschindler](#)**

**Thank you!**