



# Zend\_Layout & Zend\_View Enhancements

**Ralph Schindler**  
**Software Engineer,**  
**Zend Technologies**

*Zend Framework includes a powerful set of components that facilitate best practices in the area of keeping a consistent look and feel within an application.*



# Overview

- **The problem Zend\_Layout/ZVE solves**
- **Simple MVC Usage**
- **Benefits of Zend\_Layout & ZVE**
- **Advanced Usage: Ajax Autocomplete**
- **Q & A**

Exploring the problem area

# THE PROBLEM

# The Problem

## What are layouts?

- **Consistent look and feel across application**
- **Independent of dispatched application code**
- **Common page items such as:**
  - Navigation
  - Headers
  - Footers
  - Tag cloud

# The Problem

## Previous attempts (PHP4 until now):

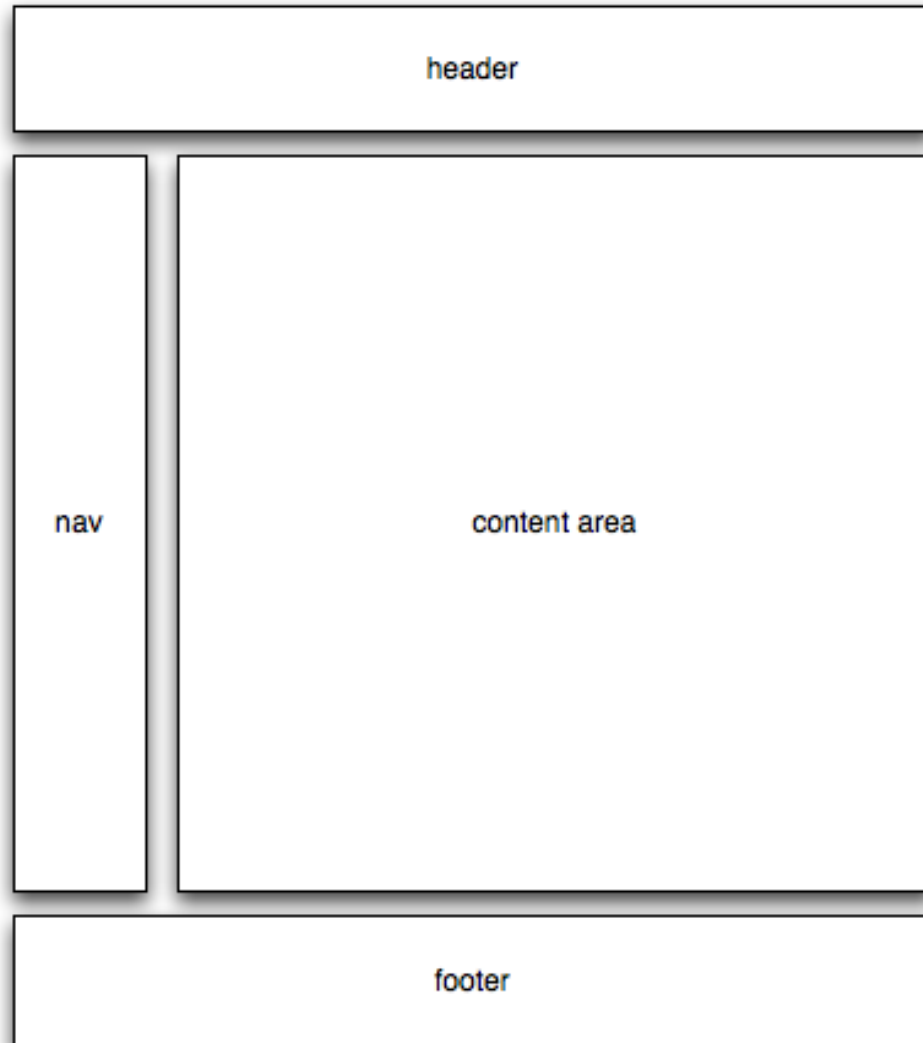
- **Smarty (separation of business & presentation logic)**
  - First divergence from Model 1 programming
- **Common solution (included in every template):**
  - Header {include file='header.tpl'}
  - Footer files {include file='footer.tpl'}
  - Navigation {include file='common/nav.tpl'}

Ref: [http://en.wikipedia.org/wiki/Model\\_1](http://en.wikipedia.org/wiki/Model_1)

# The Problem

## Processed Inline

```
{include file="header.tpl"}  
{include file="nav.tpl"}  
<!-- page content -->  
{include file="footer.tpl"}
```



# The Problem

## Previous attempts in ZF community:

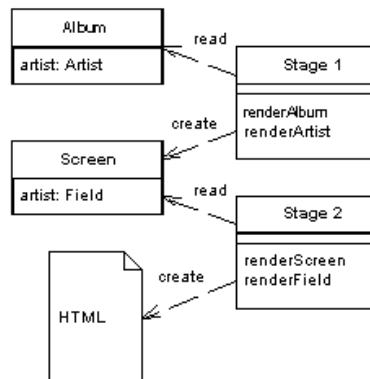
- **Controller Plugin**
  - `dispatchLoopStartup()` / `preDispatch()`
  - `dispatchLoopShutdown()` / `postDispatch()`
- **ViewRenderer Extension**
  - `postDispatch()`
- **View Extension**

# The Problem

## Does a best practices pattern exist?

- Yes, in PoEAA, M. Fowler describes the **Two-Step-View Pattern**. ([link](#))

For a full description see [P of EAA](#) page 365



If you have a Web application with many pages, you often want a consistent look and organization to the site. If every page looks different, you end up with a site that users find confusing. You may also want to make global changes to the appearance of the site easily, but common approaches using Template View (350) or Transform View (361) make this difficult because presentation decisions are often duplicated across multiple pages or transform modules. A global change can force you to change several files.

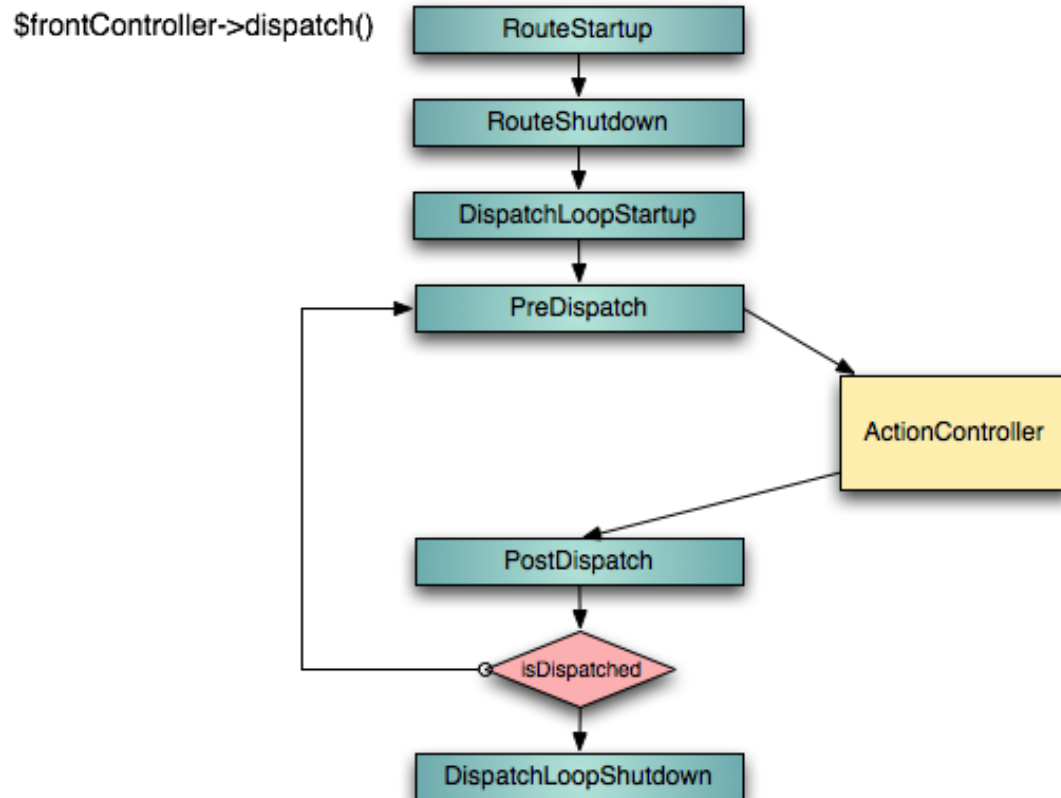
Two Step View deals with this problem by splitting the transformation into two stages. The first transforms the model data into a logical presentation without any specific formatting; the second converts that logical presentation with the actual formatting needed. This way you can make a global change by altering the second stage, or you can support multiple output looks and feels with one second stage each.

# The Problem

**Can ZF implement a Two-Step-View solution?**

- **Yes, if we leverage both the controller and the view layer, a two-step-view is possible**
- **Lets look at the controller dispatch process:**
- **(image on next screen)**

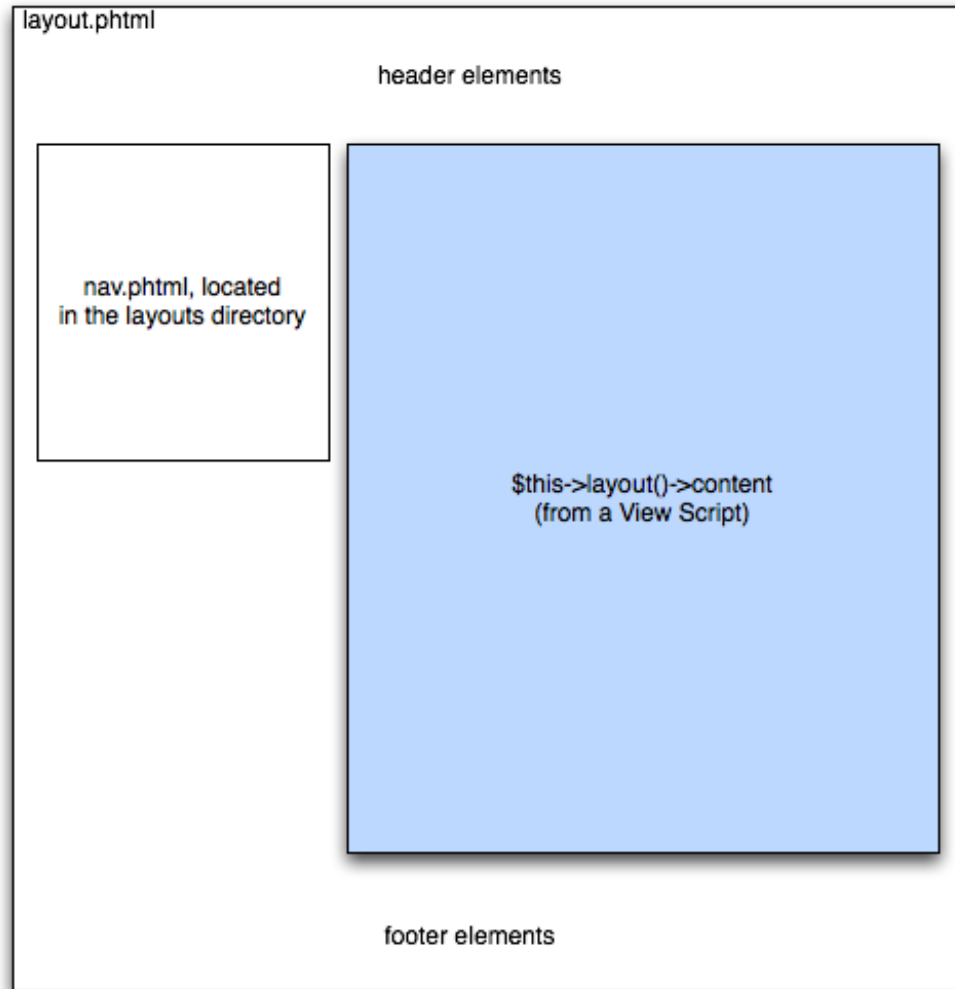
# The Problem



# The Problem

- **Zend\_Layout is the solution!**
- **Zend\_Layout by itself is simply a view decorator**
- **When used in conjunction with the MVC elements it's so much more:**
  - A Controller Plugin for detecting when to render a layout
  - An Action Helper to facilitate communication between Action Controllers and Layouts
  - A View Helper to facilitate communication between View Scripts and Layouts

# The Problem



Header and footer elements exist in the same file. This approach allows for placing of view script content inside of a layout script.

# The Problem

**The Two-Step-View and Zend\_Layout introduce new concerns:**

- **How can view scripts know the content type of the current layout?**
- **View Scripts might use code that implies a requirement at the layout layer:**
  - Setting page title
  - Inject JS in the HEAD block
  - Inject Style requirements in the HEAD block
  - Etc.

# The Problem

**With new problems come new solutions!**

- **Zend View Enhancements**

- Doctype helper for setting/getting content type
- Head Helpers:
  - headScript()
  - headMeta()
  - headStyle()
  - headTitle()

# The Problem

In addition to solving problems, there are a few other `Zend_View` Enhancements to simplify life:

- `Partial()`, `PartialLoop()`, and `Placeholder()` exist to aid developers in DRYing up their code
- `Action()` View Helper exists to facilitate the dispatching of an Action Controller when a task requires that views attempt to gain new information from the model layer

# The Problem

## **Zend\_Layout and Zend\_View Enhancements are the Solution!**

- **DRY up code**
- **Best Practices**
- **Better code organization, both application and display logic**
- **Ability to add new features and requirements to a project without having to retrofit.**

Using Zend\_Layout & Zend\_View enhancements within a ZF  
MVC application

# BASIC MVC USAGE

# Basic MVC Usage

## File structure

- ▼ layout-website
  - ▼ application
    - ▼ controllers
      - ▶ DemoController.php
      - ▶ ErrorController.php
      - ▶ IndexController.php
    - ▼ layouts
      - ▶ main.phtml
      - ▶ secondary.phtml
      - ▶ secondary-extended.phtml
    - ▼ models
      - ▶ Player.php
    - ▼ views
      - ▶ filters
      - ▶ helpers
      - ▼ scripts
        - ▼ \_partials
          - ▶ player-info.phtml
        - ▼ demo
          - ▶ ajaxy-autocomplete.phtml
          - ▶ index.phtml
          - ▶ player-random.phtml
          - ▶ players.phtml
        - ▼ index
          - ▶ contact-us.phtml
          - ▶ index.phtml
    - ▶ bootstrap.php
  - ▼ library
    - ▶ Zend
  - ▼ public
    - ▶ js
    - ▶ .htaccess
    - ▶ index.php

# Basic MVC Usage

## bootstrap.php

```
bootstrap.php ✕
ork/layout-website/
rap.php
3 define('APPLICATION_DIRECTORY', dirname(__FILE__));
4 set_include_path(dirname(APPLICATION_DIRECTORY) . '/library');
5
6 require_once 'Zend/Loader.php';
7 Zend_Loader::registerAutoload();
8
9 // set Model path in include path
10 set_include_path(get_include_path() . PATH_SEPARATOR . APPLICATION_DIRECTORY . '/models');
11
12 // bootstrap layouts
13 Zend_Layout::startMvc(array(
14     'layoutPath' => APPLICATION_DIRECTORY . '/layouts',
15     'layout' => 'main'
16 ));
17
18 Zend_Controller_Front::run(APPLICATION_DIRECTORY . '/controllers');
19
```

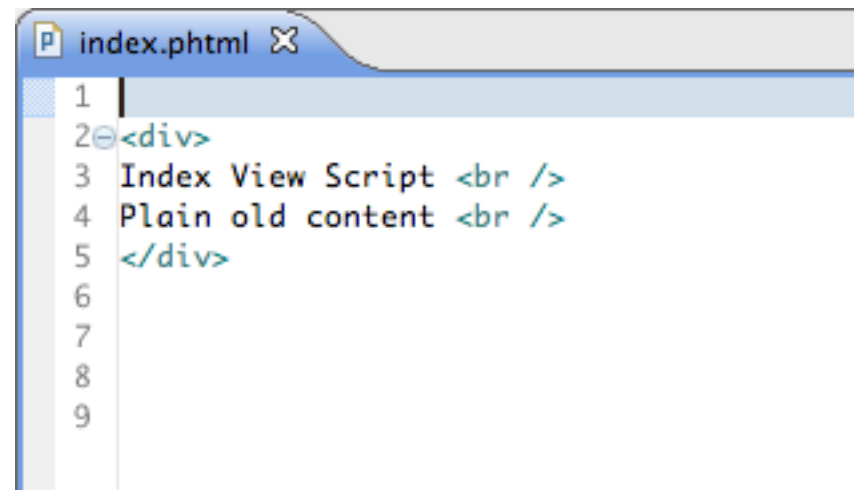
# Basic MVC Usage

## Controller Script

```
IndexController.php X
1 <?php
2
3 class IndexController extends Zend_Controller_Action
4 {
5
6     public function indexAction()
7     {
8         // auto render
9     }
10
11    public function contactUsAction()
12    {
13        $this->_helper->layout()->setLayout('secondary');
14    }
15
16
17 }
18
```

# Basic MVC Usage

## View Script



```
index.phtml
1
2 <div>
3   Index View Script <br />
4   Plain old content <br />
5 </div>
6
7
8
9
```

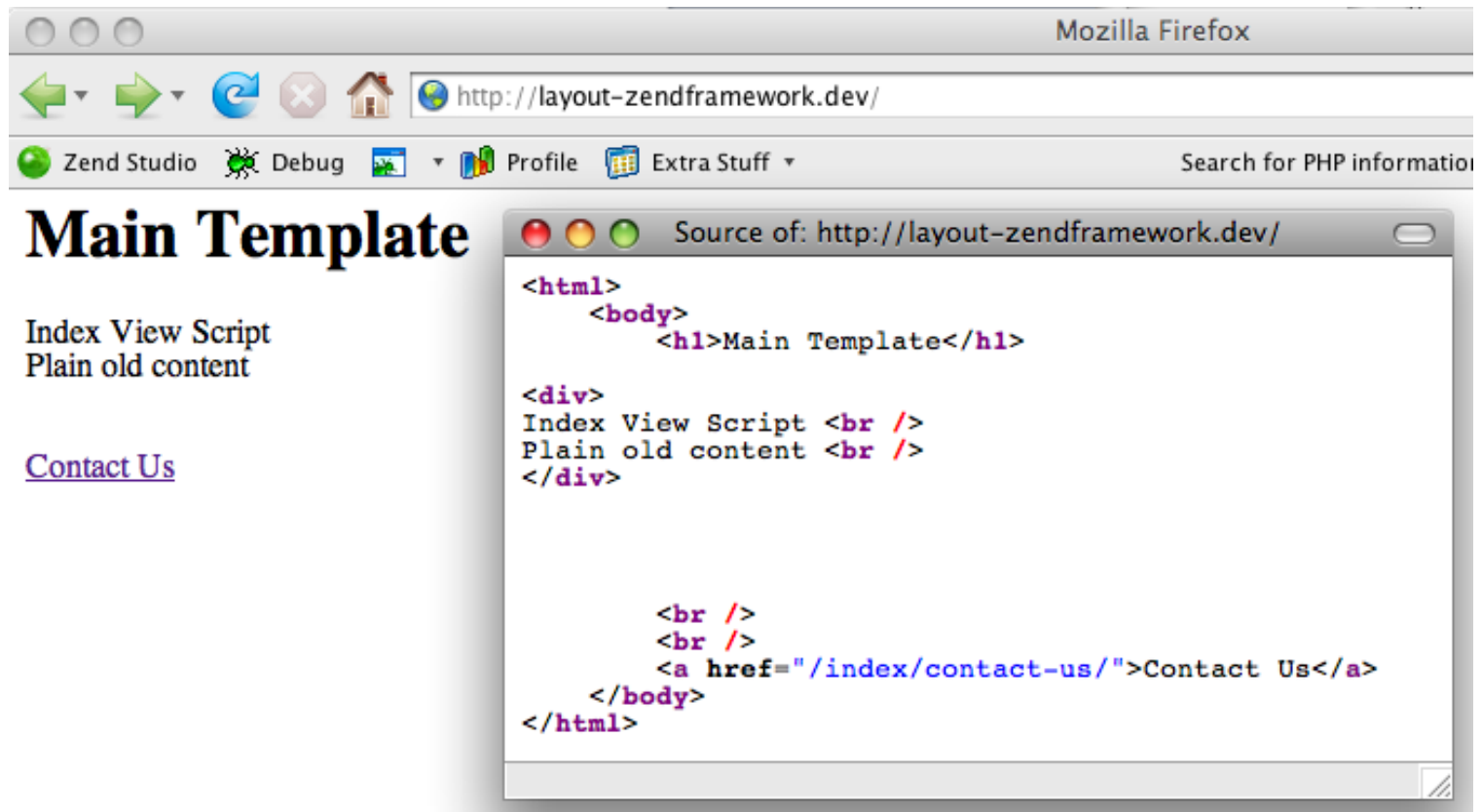
# Basic MVC Usage

- **Layout Script**

```
main.phtml ✖
1 <html>
2   <body>
3     <h1>Main Template</h1>
4     <?=$this->layout()->content ?>
5
6     <br />
7     <br />
8     <a href="/index/contact-us/">Contact Us</a>
9   </body>
10 </html>
11
```

# Basic MVC Usage

- Output HTML & Display



The screenshot shows a Mozilla Firefox browser window displaying a web page titled "Main Template". The browser's address bar shows the URL "http://layout-zendframework.dev/". The page content includes the text "Index View Script" and "Plain old content", followed by a link labeled "Contact Us". An inset window titled "Source of: http://layout-zendframework.dev/" displays the HTML source code for the page.

```
<html>
  <body>
    <h1>Main Template</h1>

    <div>
      Index View Script <br />
      Plain old content <br />
    </div>

    <br />
    <br />
    <a href="/index/contact-us/">Contact Us</a>
  </body>
</html>
```

# Basic MVC Usage

- **HeadTitle Usage**

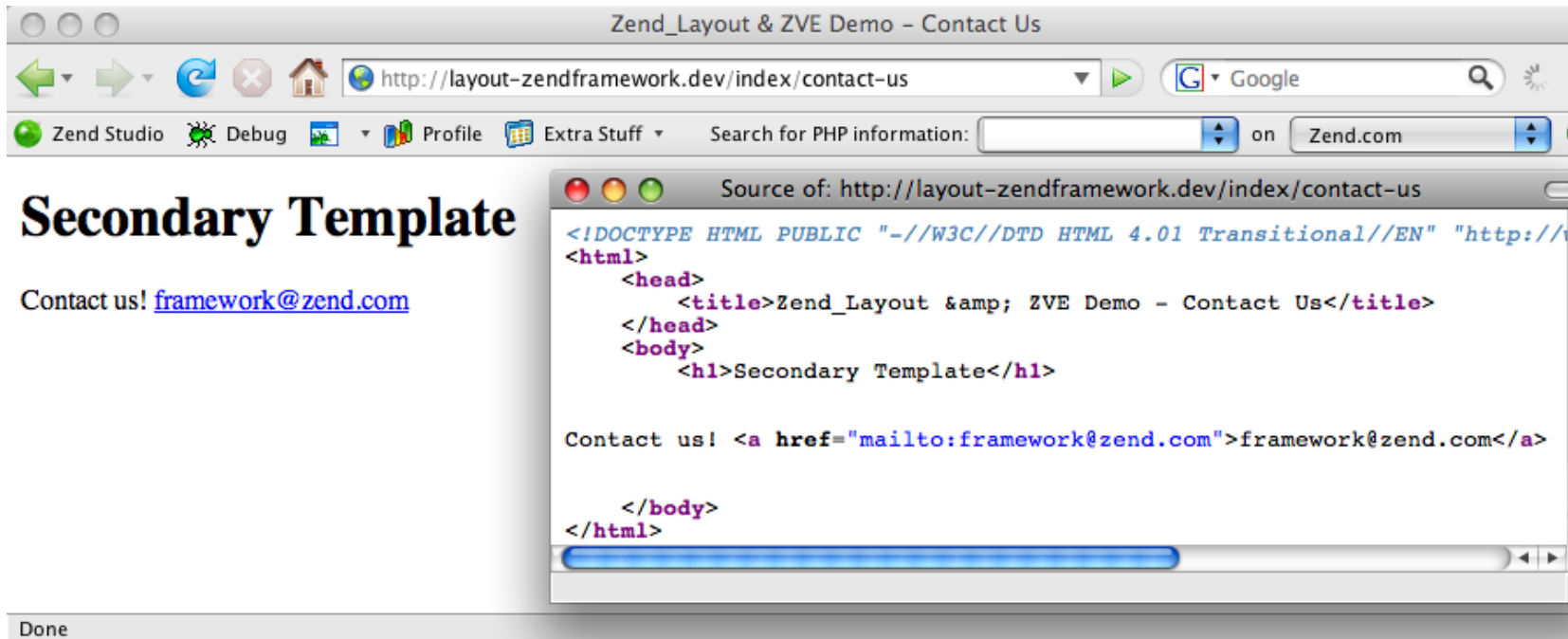
```
IndexController.php
11 public function contactUsAction()
12 {
13     $this->_helper->layout()->setLayout('secondary');
14 }
```

```
contact-us.phtml
1 <?php
2
3 // set the title
4 $this->headTitle('Contact Us');
5
6 ?>
7
8
9 Contact us! <a href="mailto:framework@zend.com">framework@zend.com</a>
10
```

```
secondary.phtml
1 <?= $this->doctype() ?>
2
3 <html>
4     <head>
5         <?
6         $this->headTitle()->prepend('Zend_Layout & ZVE Demo');
7         $this->headTitle()->setSeparator(' - ');
8         echo $this->headTitle();
9         ?>
10
11     </head>
12     <body>
13         <h1>Secondary Template</h1>
14         <?= $this->layout()->content ?>
15
16     </body>
17 </html>
18
```

# Basic MVC Usage

## Output HTML & View



The screenshot shows a web browser window titled "Zend\_Layout & ZVE Demo - Contact Us". The address bar contains the URL "http://layout-zendframework.dev/index/contact-us". The browser's toolbar includes navigation buttons (back, forward, refresh, home), a search bar with "Google", and a search for PHP information field. The main content area displays the text "Secondary Template" in a large, bold font, followed by "Contact us! [framework@zend.com](mailto:framework@zend.com)".

An inset window titled "Source of: http://layout-zendframework.dev/index/contact-us" shows the following HTML code:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://...>
<html>
  <head>
    <title>Zend_Layout & ZVE Demo - Contact Us</title>
  </head>
  <body>
    <h1>Secondary Template</h1>

    Contact us! <a href="mailto:framework@zend.com">framework@zend.com</a>

  </body>
</html>
```

The browser's status bar at the bottom left shows "Done".

# Basic MVC Usage

- **Partial & PartialLoop**
- `$this->partial($script, $model)`
- `$this->partialLoop($script, $arrayOfModels)`
- **Assumes “model” is**
  - An array
  - Implements `toArray()`
  - Or is an object that can return properties via `object_get_vars()`
  - An object, when `partial()` is configured as such

# Basic MVC Usage

- PartialLoop

```
Player.php
83 public function toArray()
84 {
85     return array(
86         'name' => $this->_name,
87         'wins' => $this->_wins,
88         'losses' => $this->_losses
89     );
90 }
```

```
DemoController.php
20 public function playersAction()
21 {
22     $this->view->players = Player::getAll();
23 }
```

```
players.phtml
2
3 Players
4 <dl>
5 <?=$this->partialLoop('_partials/player-info.phtml', $this->players); ?>
6 </dl>
```

```
player-info.phtml
1
2
3 <dt><?=$this->name; ?></dt>
4 <dd><?=$this->wins; ?> wins</dd>
5 <dd><?=$this->losses; ?> losses</dd>
6
7
```

# Basic MVC Usage

## PartialLoop Model

```
Player.php
83 public function __get($name)
84 {
85     $propertyName = '_' . $name;
86     if (property_exists($this, $propertyName)) {
87         return $this->{$propertyName};
88     }
89
90     throw new Exception('Property ' . $name . ' does not exist.');
```

```
DemoController.php
25 public function playersAsModelAction()
26 {
27     $this->view->players = Player::getAll();
28 }
```

```
players-as-model.phtml
1
2 Players
3 <dl>
4 <?=$this->getHelper('partialLoop')->setObjectKey('player')->partialLoop('_partials/player-info-model.phtml', $this->players); ?>
5 </dl>
```

```
player-info-model.phtml
1
2
3 <dt><?=$this->player->name; ?></dt>
4 <dd><?=$this->player->wins; ?> wins</dd>
5 <dd><?=$this->player->losses; ?> losses</dd>
6
7
```

# Basic MVC Usage

## The action() helper

```
another-player-random.phtml X
1
2
3 I want to dispatch a request from here
4
5 <?= $this->action('player-random', 'demo')
```

```
Source of: http://layout-zendframework.dev/demo/another-player-random.phtml
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Zend_Layout & ZVE Demo</title>

  </head>
  <body>
    <h1>Secondary Template</h1>

    I want to dispatch a request from here

    <dl>

      <dt>Cody</dt>
      <dd>4 wins</dd>
      <dd>4 losses</dd>

    </dl>

  </body>
</html>
```

**Just a little review on the benefits**

# **BENEFITS**

## DRY up code

- **When every page template use to have:**
  - `{include file="header.tpl"}`
  - `{include file="footer.tpl"}`
- **View Scripts no longer contain these elements.**

# Benefits

## **Ability to scale and grow code without retrofitting**

- **Since we have DRYed up our code, we no longer have any layout dependencies in view scripts**
- **This allows any number of new view scripts to be included into an existing system seamlessly and without changing anything at the layout layer**

**Zend\_Layout & Zend\_View enhancements are a supported best practice**

- **Developers can find the code locations for improvements and fixes faster**
- **Developers can get up to speed quicker with the detailed documentation and best practices these components promote**

**Brief discussion of the advanced usage possibilities**

# **ADVANCED USAGE**

# Advanced Usage

## AJAX Support

- **Pulling all the components together to make an Ajaxy Autocompleter**
  - headScript
  - headStyle
  - Dojo Toolkit (<http://www.dojotoolkit.org/> )
  - Zend\_Layout

# Advanced Usage

## The Model

```
Player.php
4 class Player
5 {
6
7     protected static $_players = array(
8         'Pedram' => array('wins' => 5, 'losses' => 3),
9         'Ralph'  => array('wins' => 8, 'losses' => 0),
10        'Cody'   => array('wins' => 4, 'losses' => 4),
11        'Cameron' => array('wins' => 2, 'losses' => 6)
12    );
13
14    protected $_name = null;
15    protected $_wins = null;
16    protected $_losses = null;
17
18    public static function getAll()
19    {
20        $players = array();
21        foreach (self::$_players as $playerName => $playerInfo) {
22            $players[] = new self($playerName, $playerInfo['wins'], $playerInfo['losses']);
23        }
24
25        return $players;
26    }
27
```

# Advanced Usage

## Controller

```
DemoController.php X
1 <?php
2
3 class DemoController extends Zend_Controller_Action
4 {
5
6     public function init()
7     {
8         $this->getHelper('layout')->setLayout('secondaryExtended');
9     }
10
11    public function indexAction() {}
12
13
14
15    * this action demonstrates partialLoop usage.
16
17    public function playersAction() {}
18
19
20
21
22
23
24    * this action demonstrates partial usage.
25
26    public function playerRandomAction() {}
27
28
29
30
31
32
33
34    public function ajaxyAutocompleteAction()
35    {
36
37    }
38
39    public function ajaxyAutocompleteDataAction()
40    {
41        $this->getHelper('layout')->disableLayout();
42        $this->getHelper('ViewRenderer')->setNoRender();
43        $json = new Zend_Json();
44        foreach (Player::getAll() as $player) {
45            $players[] = $player->toArray();
46        }
47
48        $this->getResponse()->setBody($json->encode(array('identifier' => 'name', 'items' => $players)));
49    }
50
51 }
52
```

# Advanced Usage

## View

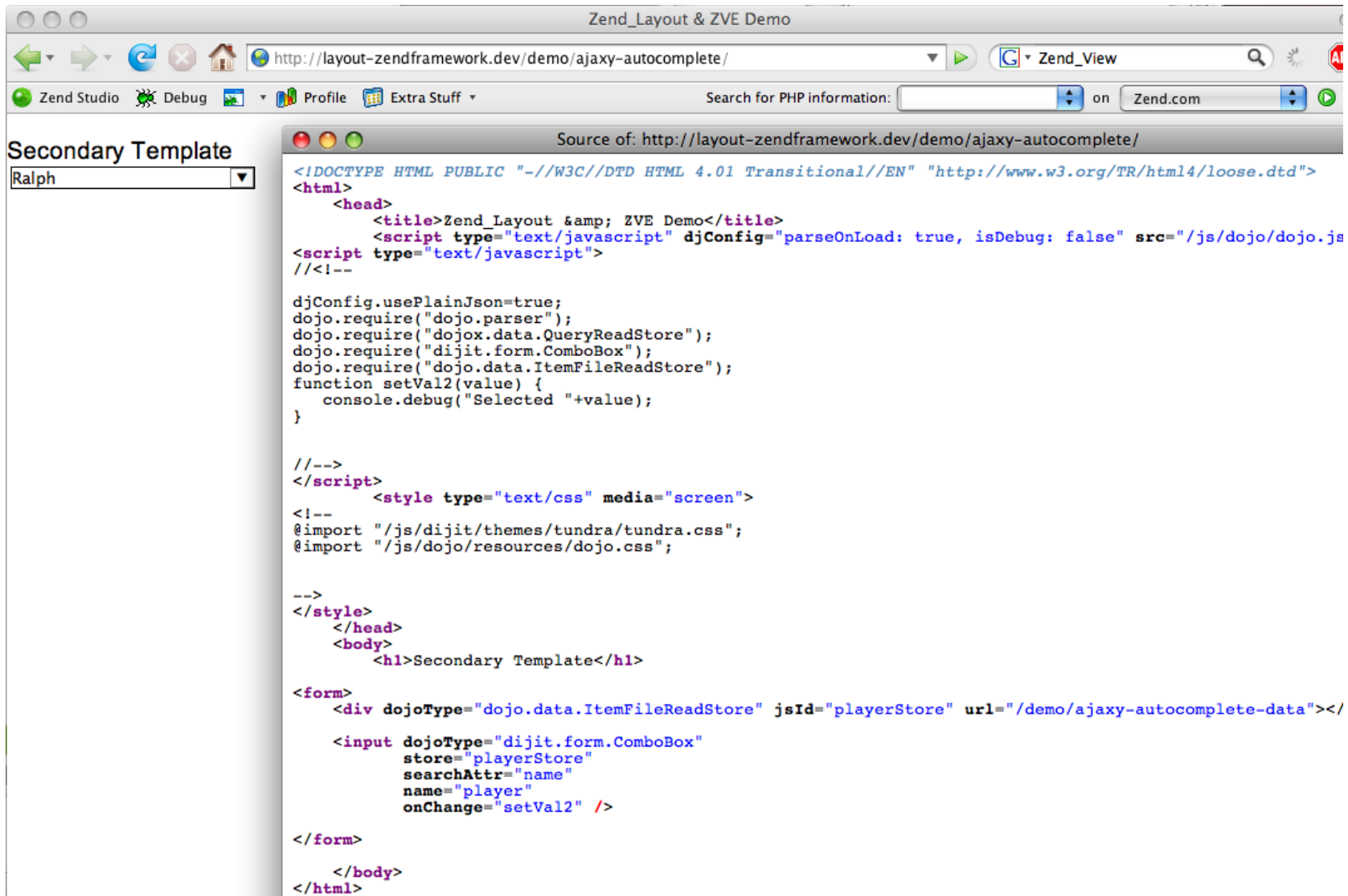
```
ajaxy-autocomplete.phtml x
1 <?php
2 // start the capturing for headStyle
3 $this->headStyle()->captureStart();
4 ?>
5 @import "/js/dijit/themes/tundra/tundra.css";
6 @import "/js/dojo/resources/dojo.css";
7
8 <?
9 // capture previous output
10 $this->headStyle()->captureEnd();
11
12 // configure the headScript
13 $this->headScript()
14     ->setAllowArbitraryAttributes(true)
15     ->appendFile('/js/dojo/dojo.js', 'text/javascript', array('djConfig' => 'parseOnLoad: true, isDebug: false'));
16
17 // now that its configured, start capturing
18 $this->headScript()->captureStart();
19
20 ?>
21
22 djConfig.usePlainJson=true;
23 dojo.require("dojo.parser");
24 dojo.require("dojox.data.QueryReadStore");
25 dojo.require("dijit.form.ComboBox");
26 dojo.require("dojo.data.ItemFileReadStore");
27 function setVal2(value) {
28     console.debug("Selected "+value);
29 }
30
31 <?
32 // finish captureing
33 $this->headScript()->captureEnd();
34
35 ?>
36
37 <form>
38     <div dojoType="dojo.data.ItemFileReadStore" jsId="playerStore" url="/demo/ajaxy-autocomplete-data"></div>
39
40     <input dojoType="dijit.form.ComboBox"
41         store="playerStore"
42         searchAttr="name"
43         name="player"
44         onChange="setVal2" />
45
46 </form>
```

# Advanced Usage

## Layout

```
secondary-extended.phtml x
1 <?=$this->doctype() ?>
2
3 <html>
4   <head>
5     <?
6     $this->headTitle()->setSeparator(' - ');
7     $this->headTitle()->prepend('Zend_Layout & ZVE Demo');
8     echo $this->headTitle();
9     ?>
10
11     <?=$this->headScript(); ?>
12
13     <?=$this->headStyle(); ?>
14
15   </head>
16   <body>
17     <h1>Secondary Template</h1>
18     <?=$this->layout()->content ?>
19
20   </body>
21 </html>
```

# Advanced Usage



Zend Studio Debug Profile Extra Stuff Search for PHP information: on Zend.com

## Secondary Template

Ralph

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Zend_Layout & ZVE Demo</title>
    <script type="text/javascript" djConfig="parseOnLoad: true, isDebug: false" src="/js/dojo/dojo.js">
    </script>
    <script type="text/javascript">
    //<!--

    djConfig.usePlainJson=true;
    dojo.require("dojo.parser");
    dojo.require("dojox.data.QueryReadStore");
    dojo.require("dijit.form.ComboBox");
    dojo.require("dojo.data.ItemFileReadStore");
    function setVal2(value) {
      console.debug("Selected "+value);
    }

    //-->
    </script>
    <style type="text/css" media="screen">
    <!--
    @import "/js/dijit/themes/tundra/tundra.css";
    @import "/js/dojo/resources/dojo.css";

    -->
    </style>
  </head>
  <body>
    <h1>Secondary Template</h1>

    <form>
      <div dojoType="dojo.data.ItemFileReadStore" jsId="playerStore" url="/demo/ajaxy-autocomplete-data"></div>

      <input dojoType="dijit.form.ComboBox"
        store="playerStore"
        searchAttr="name"
        name="player"
        onChange="setVal2" />

    </form>

  </body>
</html>
```

# Advanced Usage

- JSON Output



The screenshot shows a code editor window titled 'DemoController.php' with the following PHP code:

```
39 public function ajaxyAutocompleteDataAction()
40 {
41     $this->getHelper('layout')->disableLayout();
42     $this->getHelper('ViewRenderer')->setNoRender();
43     $json = new Zend_Json();
44     foreach (Player::getAll() as $player) {
45         $players[] = $player->toArray();
46     }
47
48     $this->getResponse()->setBody($json->encode(array('identifier' => 'name', 'items' => $players)));
49 }
50
51 }
52
```

Below the code editor is a Mozilla Firefox browser window. The address bar shows the URL: `http://layout-zendframework.dev/demo/ajaxy-autocomplete-data`. The browser's developer tools are open, showing the JSON output of the request:

```
{"identifier": "name", "items": [{"name": "Pedram", "wins": 5, "losses": 3}, {"name": "Ralph", "wins": 8, "losses": 3}]}
```

# Advanced Usage

Or use the built in autoComplete Helper

- (Dojo and Sciptaculous helpers exist)

```
DemoController.php ✕
76 public function ajaxyAutocompleteDataHelperAction()
77 {
78     foreach (Player::getAll() as $player) {
79         $players[] = $player->getName();
80     }
81     $this->getHelper('autoCompleteDojo')->direct($players);
82 }
83
84 }
```

Stump the chump! 😊

**Q&A TIME**

Slides & Sample code will be provided following the webinar

# WEBINAR & DEMO

**Matthew W. O'Phinney's Blog:**

[http://weierophinney.net/matthew/archives/163-Using-Zend\\_View-Placeholders-to-Your-Advantage.html](http://weierophinney.net/matthew/archives/163-Using-Zend_View-Placeholders-to-Your-Advantage.html)

**Zend\_Layout & Zend\_View Manual:**

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

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

**Mailing List & #zftalk**

# RESOURCES



Thank You!

<http://framework.zend.com>  
[ralph.schindler@zend.com](mailto:ralph.schindler@zend.com)