Zend_Layout

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Layout Component Proposal

Proposed Component Name	Zend_Layout
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Layout
Proposers	Ralph Schindler Matthew Weier O'Phinney
Revision	1.0 - 5 July 2007: Updated from community comments. 1.1 - 10 Aug 2007: Took out of construction and placed under review 1.2 - 30 Aug 2007: Updated proposal with new class skeletons 1.3 - 21 Sep 2007: Updated with layout view helper and Zend_View_Inflection 1.4 - 27 Sep 2007: Updated with layout variable placeholder implementation (wiki revision: 10)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
Bootstrap
Action controller accessing layout
View accessing layout
Layout script
9. Class Skeletons
Zend_Layout
Zend_Layout_Controller_Plugin
Zend_Layout_Action_Helper
Zend_View_Helper_Layout
Zend_View_Inflection

1. Overview

Zend_Layout is a composite component that joins the Controller, View and ViewRenderer together to implement a two-step-view pattern for solving the problem of "consistent look and feel".

2. References

- Rails Layouts
- Solar Views and Layouts
- Cake Views
- Symfony Views

- Two Step View Pattern

3. Component Requirements, Constraints, and Acceptance Criteria

- **Must work with a dispatch loop and named response segments.** Since we can forward between actions, or create a loop of actions to dispatch, and because the response content can exist in multiple segments, we have special needs and capabilities other frameworks don't. Rendering must be done **after** the dispatch loop has finished, and should use all content segments.
- **Must be able to work *without* MVC.** While the above should be true, Layouts should be available to those not wanting to use the MVC, or using an alternate MVC.
- **Must be simple to invoke.** A controller should be able to specify a layout at will, and this should be easy to do.
- **Must be configurable**
    - Must be able to specify which layout to use, but allow for a 'default' layout
    - Must be able to specify location of layouts
    - Must be able to specify names of layout variables
- **Must be able to pick and choose which placeholders to render**
- ~~**Must have a simple API for specifying items to use in the <head> section**~~ In discussion with Ralph, we've decided that view variables and the collection helper are the proper solution for this.
- **Must follow same naming conventions followed in ViewRenderer**
    - naming
    - suffixes
- **Must inherit from view set in view renderer (if in use).** This allows view scripts as well as action controllers to set variables to use in the layout.

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_View_Interface
- Zend_View_Helper_Placeholder (potentially; for variable storage and retrieval)
- Zend_Controller_Action_Helper_ViewRenderer (will likely push inflection into a new class, and ViewRenderer and Zend_Layout will utilize that)
- Zend_Controller_Plugin_Broker (optional)
- Zend_Controller_Action_Helper_Abstract (optional)
- Zend_Controller_Action_Helper_ViewRenderer (optional; for shared view object)
- Zend_Controller_Action_HelperBroker (optional)

5. Theory of Operation

Zend_Layout attempts to solve a problem that, since rearing its ugly head over and over, has gone by many names: Composite Views, Layouts, Templates, Partial Views, and/or Complex Views. They all basically attempt to describe a common problem - that of being able to maintain a consistent look and feel throughout a site or web application while maintaining the "Don't Repeat Yourself" principals.

Zend_Layout addresses the problem with a common pattern, the two-step-view pattern (http://www.martinfowler.com/eaaCatalog/twoStepView.html). The Two Step View pattern breaks the process up into two distinct stages. The primary stage allows the user requested action as well as layout requested actions to dispatch and execute their respective views saving them to the controllers response object. The second stage of the process takes the already dispatched actions responses and directs them to the layout where final placement of content is made (in a Layout specific view) before being sent back to the user.

Zend_Layout requires no changes to the existing codebase to accomplish its job. To maintain control over the dispatch process, Zend_Layout attaches a controller plugin - this is the essense of the Two-Step-View and a common way to implement the 2SV as we have seen already. Zend_Layout also registers its own action helper, so that the developer can directly interact with Zend_Layout from within their actions. Depending on the state of the layout subsystem (defined by either the config file, bootstrap declarations, or interactions with the action helper), Zend_Layout will engage and act as directed by the developer by either dispatching a 2SV layout, or simply not engaging at all (if so desired).

A major benefit the Two-Step-View implementation is that user requested (and layout requested) action controllers and views can inject dependencies into the final layout. For example, if the user has requested a url that would generate a form, the form view can request that the section of the overall layout include a specific form.js file, or even a form.css file be linked in the head section.

Finally, Zend_Layout can be used with the MVC, if so desired. If used in this way, it simply acts as a wrapper on Zend_View, allowing the developer to inject layout-specific variables.

6. Milestones / Tasks

- Milestone 1: [DONE] design notes will be published here
- Milestone 2: [DONE] Working prototype @ http://svn.ralphschindler.com/repo/ZendFramework/Zend_Layout/
- Milestone 3: [DONE] Working prototype checked into the incubator supporting use cases
- Milestone 4: [DONE] Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

If a milestone is already done, begin the description with "[DONE]", like this:

- Milestone #: [DONE] Unit tests ...

7. Class Index

- Zend_Layout
- Zend_Layout_Action_Helper_Layout
- Zend_Layout_Controller_Plugin
- Zend_Layout_Exception
- Zend_View_Helper_Layout
- Zend_View_Inflection (Used by both ViewRenderer and Zend_Layout for mapping script names to actual scripts)

8. Use Cases

UC-01

Bootstrap

```
// SETUP
$layout = new Zend_Layout(APPLICATION_PATH . 'user/layouts/');
// - or $layout = new Zend_Layout(array('path' => APPLICATION_PATH . 'user/layouts/',
'view' => $view));

// SET DEFAULT LAYOUT - this will set
"APPLICATION_PATH/user/layouts/scripts/sub-page.phtml"
// (layouts/scripts) assumes that you are using a view object of type Zend_View.
$layout->setLayout('SubPage');
```

Action controller accessing layout

```
class IndexController extends Zend_Controller_Action
{
    public function indexAction()
    {
        // Default Layout
    }

    public function fileAction() {
        // Layout from alternative.phtml
        $this->_helper->layout()->setLayout('alternative');
    }

    public function namedAction()
    {
        // Layout from bootstrap configured alternative
        $this->_helper->layout()->setLayout('alternative');
    }

    public function jpegAction()
    {
        $this->_helper->layout()->disableLayout();
    }
}
```

View accessing layout

```
<?
// assigning vars to layout:
$this->layout()->myLayoutVar = $this->something

// changing layout script:
$this->layout()->setLayout('foobar');

// disabling layout:
$this->layout()->disableLayout();
?>
```

Layout script

```
<?= $this->docType('xhmt1-transitional') ?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title><?= $this->headTitle() ?></title>
</head>
<body>
<? // 'default' content ?>
<?= $this->layout()->content?>
</body>
</html>
```

9. Class Skeletons

Zend_Layout

```
class Zend_Layout
{
    /**
     * Set layout script to use
     *
     * If set after disableLayout() called, implicitly re-enables layout.
     *
     * @param  string $name
     * @return Zend_Layout
     */
    public function setLayout($name)
    {}

    /**
     * Get current layout script
     *
     * @return string
     */
    public function getLayout()
    {}

    /**
     * Disable layout
     *
     * @return void
     */
    public function disableLayout()
    {}

    /**
     * Set layout script path
     *
     * @param  string $path
     * @return Zend_Layout
     */
    public function setLayoutPath($path)
    {}

    /**
     * Get current layout script path
     *
     * @return string
     */
    public function getLayoutPath()
    {}

    /**
     * Set view object
     *
     * @param  Zend_View_Interface $view
     * @return Zend_Layout
```

```
     */
    public function setView(Zend_View_Interface $view)
    {}

    /**
     * Get current view object
     *
     * @return Zend_View_Interface
     */
    public function getView()
    {}

    /**
     * Set layout variable
     *
     * @param  string $key
     * @param  mixed $value
     * @return void
     */
    public function __set($key, $value)
    {}

    /**
     * Get layout variable
     *
     * @param  string $key
     * @return mixed
     */
    public function __get($key)
    {}

    /**
     * Is a layout variable set?
     *
     * @param  string $key
     * @return bool
     */
    public function __isset($key)
    {}

    /**
     * Unset a layout variable?
     *
     * @param  string $key
     * @return void
     */
    public function __unset($key)
    {}

    /**
     * Assign one or more layout variables
     *
     * @param  mixed $spec Assoc array or string key; if assoc array, sets each
     * key as a layout variable
     * @param  mixed $value Value if $spec is a key
     * @return Zend_Layout
     */
    public function assign($spec, $value = null)
    {}
```

```
    /**
     * Render layout
     *
     * Sets internal script path as last path on script path stack, assigns
     * layout variables to view, determines layout name using inflector, and
     * renders layout view script.
     *
     * @param  mixed $name
     * @return mixed
     */
    public function render($name = null)
    {
    }

    /**
     * Constructor
     *
     * Accepts either:
     * - A string path to layouts
     * - An array of options
     * - A Zend_Config object with options
     *
     * Layout script path, either as argument or as key in options, is
     * required.
     *
     * If useMvc flag is false from options, simply sets layout script path.
     * Otherwise, also instantiates and registers action helper and controller
     * plugin.
     *
     * @param mixed $options
     * @return void
     */
    public function __construct($options)
    {
```

```
    }
}
```

Zend_Layout_Controller_Plugin

```
class Zend_Layout_Controller_Plugin extends Zend_Controller_Plugin_Abstract
{
    public function __construct(Zend_Layout $layout = null)
    {
    }

    public function setLayout(Zend_Layout $layout)
    {
    }

    public function getLayout()
    {
    }

    /**
     * Renders layout
     *
     * @todo Check for XHR request, check if layout is null, etc.
     * @return void
     */
    public function dispatchLoopShutdown()
    {
        $layout   = $this->getLayout();
        $response = $this->getResponse();
        $layout()->assign($response->getBody(true));
        $response->setBody($layout->render());
    }
}
```

Zend_Layout_Action_Helper

```
class Zend_Layout_Action_Helper extends Zend_Controller_Action_Helper_Abstract
{
    public function setLayout(Zend_Layout $layout)
    {
    }

    public function getLayout()
    {
    }

    public function setLayoutName($name)
    {
        $this->getLayout()->setLayout($name);
    }

    public function direct($name)
    {
        return $this->setLayoutName($name);
    }
}
```

Zend_View_Helper_Layout

```
/**
 * Layout view helper
 *
 * @package    Zend_View
 * @subpackage Helpers
 * @license    New BSD {@link http://framework.zend.com/license/new-bsd}
 */
class Zend_View_Helper_Layout
{
    protected $_front;

    public $pluginClass = 'Zend_Layout_Controller_Plugin';

    /**
     * Constructor
     *
     * Get front controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->_front = Zend_Controller_Front::getInstance();
    }

    /**
     * Retrieve Zend_Layout object
     *
     * @return Zend_Layout|null
     */
    public function layout()
```

```
    {
        return $this->getLayout();
    }

    /**
     * Retrieve Zend_Layout instance
     *
     * @return Zend_Layout
     */
    public function getLayout()
    {
        if (null === $this->_layout) {
            if ($this->_front->hasPlugin($this->pluginClass)) {
                $this->_layout =
$this->_front->getPlugin($this->pluginClass)->getLayout();
            }
        }

        return $this->_layout;
    }

    /**
     * Set layout object
     *
     * @param  Zend_Layout $layout
     * @return Zend_View_Helper_Layout
     */
    public function setLayout(Zend_Layout $layout)
    {
        $this->_layout = $layout;
        return $this;
```

```
    }
}
```

Zend_View_Inflection

```
/**
 * Zend_View_Inflection
 *
 * Inflection rules for view scripts.
 *
 * @package Zend_View
 * @license New BSD {@link http://framework.zend.com/license/new-bsd}
 */
class Zend_View_Inflection
{
    /**
     * Set base path specification for view object
     *
     * @param  string $path
     * @return Zend_View_Inflection
     */
    public function setViewBasePathSpec($path)
    {}

    /**
     * Retrieve base path specification
     *
     * @return string
     */
    public function getViewBasePathSpec()
    {}

    /**
     * Set view script path specification
     *
     * @param  string $path
     * @return Zend_View_Inflection
     */
    public function setViewScriptPathSpec($path)
    {}

    /**
     * Get view script path specification
     *
     * @return string
     */
    public function getViewScriptPathSpec()
    {}

    /**
     * Retrieve view script location based on specification
     *
     * @param  string $action
     * @param  array $vars
     * @return string
```

```
     */
    public function getViewScript($action, array $vars = array())
    {}

    /**
     * Set view script suffix
     *
     * @param  string $suffix
     * @return Zend_View_Inflection
     */
    public function setViewSuffix($suffix)
    {}

    /**
     * Retrieve view script suffix
     *
     * @return string
     */
    public function getViewSuffix()
```

```
        {}
    }
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>