

Zend_Navigation

By Matthew Weier O'Phinney



Who I am

- **ZF Contributor since January 2006**
- **Assigned to the ZF team in July 2007**
- **Promoted to Software Architect in April 2008**
- **Project Lead since April 2009**

Who wrote Zend_Navigation

- **Originated in Zym Framework, and authored by Robin Skoglund**
- **Ported to Zend Framework with changes based on community feedback**

Use Cases

“

Zend_Navigation is a component for managing trees of pointers to web pages.

Use Cases

- **Search Engine Optimization (SEO)**
 - Sitemap and links enhance SEO for a site
- **Site Usability**
 - Navigation elements such as breadcrumbs and menus make traversing a site easier for end-users

Features

“

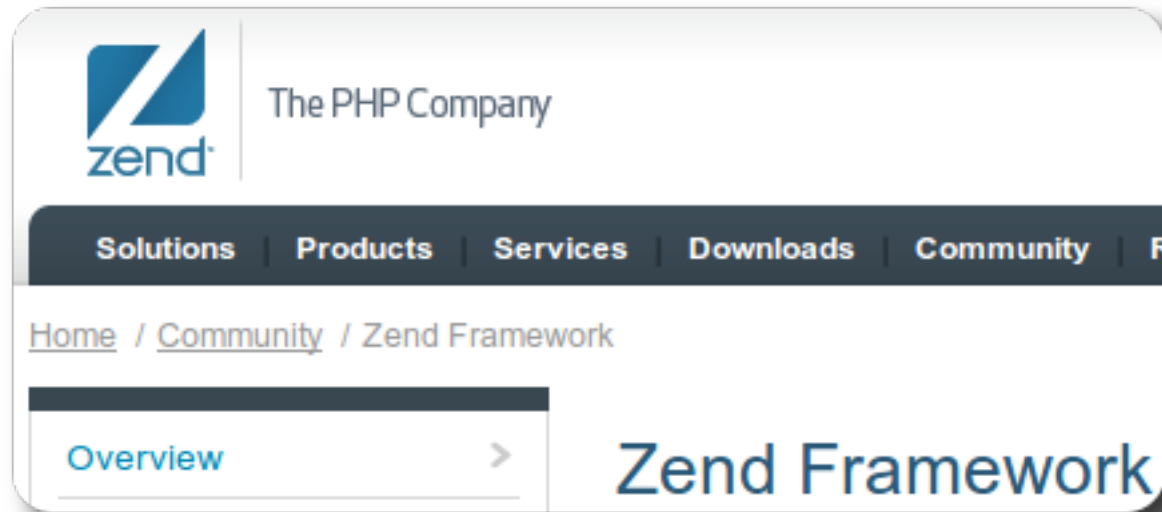
Menus, breadcrumbs,
and sitemaps, oh my!

Menus

The screenshot shows the Zend Framework website with a navigation menu open. The menu items are: OVERVIEW, QUICKSTART, APIs, REFERENCE GUIDE, TRANSLATIONS, and MULTIMEDIA. The main content area is divided into three columns:

- Make the Choice**: Standardize your PHP practices. Includes links for 'Zend Framework 1.9.0 Released' and 'Zend Framework 1.9.0 RELEASE CANDIDATE 1 Now Available'. A 'Why ZF?' section states 'You've got questions, we've got solutions'.
- Get Started**: What you need to get up to speed. Includes links for 'Jeroen Keppens on: Creating a modular application with Zend Framework' and 'Benjamin Eberlei's Blog: Using a Dependency Injection Container with Zend_Application'. A 'QuickStart' section says 'Take our 30-minute tour'.
- Give Back**: Because it just feels good. Includes links for 'PHP Performance Tips from Stas Malyshev' and 'Preview of Windows Azure Support in Zend Framework Released'. A 'Contributors Guide' section says 'Contributing is easier than you think'.

Breadcrumbs



Header Links

- **Provide hints to page regarding pages relative to current page**
 - "Start" or "Home" page
 - "Next" or "Previous" pages in same depth
 - "Chapters" or "Sub-Sections" in this area
 - Alternate formats for this content

Sitemaps

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
</urlset>
```

Body Links

- **Render links to related content**
 - previous/next page
 - parent
 - children
- **Translate link text**
- **Conditionally display links based on ACLs**

Working with Pages and Containers

“

The building blocks of
navigation

Pages

“

An object that holds a pointer to a web page.

Zend_Navigation_Page metadata

- **label**
- **title**
- **target**
- **rel**
- **order**
- **active**
- **visible**
- **ACL info (resource, privilege)**

Page Types

- **MVC**
 - Integration with the router for URL generation
- **URI**
 - Specify specific URI for the page

Creating “MVC” Pages

```
$page = new Zend_Navigation_Page_Mvc();  
$page->title      = "Foo";  
$page->controller = "foo";  
$page->action     = "index";  
$page->route      = "default";
```

Creating “URI” Pages

```
$page          = new Zend_Navigation_Page_Uri();  
$page->title   = "Foo";  
$page->uri     = "/foo";
```

Using the Page Factory

- **Zend_Navigation_Page::factory()**
- **Accepts array or Zend_Config object**
- **Rules**
 - if 'uri' provided, and no MVC options, URI page created
 - if any MVC options (action, controller, module, route), MVC page created
 - if 'type' provided, assumes this is a Page class to use
 - can be "mvc" or "uri"

Containers

“

A container for pages.

Zend_Navigation_Container

- **Implements RecursiveIterator (iterate entire tree)**
- **Implements Countable (get total count of pages)**
- **Zend_Navigation_Page is a container**
 - Allows having trees of pages

Adding a page to a container (1 of 2)

```
// addPage()  
$container->addPage($page);  
  
// addPage() using factory  
$container->addPage(  
    Zend_Navigation_Page::factory(array(  
        'uri' => 'http://www.example.com/',  
    )  
));
```

Adding a page to a container (2 of 2)

```
// addPage() using an array:  
$container->addPage(array(  
    'uri' => 'http://www.example.com/',  
));  
  
// addPage() using a Zend_Config object:  
$container->addPage($config->page);
```

Adding multiple pages to a container at once

```
// addPages() using an array:  
$container->addPages ($array) ;  
  
// addPages() using a Zend_Config object:  
$container->addPages ($config) ;
```

Removing pages from a container

```
// removePage():  
// Accepts a page instance, or an integer  
// indicating value of order metadatum within  
// container  
$container->removePage($page);  
$container->removePage(0);  
$container->removePage(20); // where order => 20  
  
// Remove all pages:  
$container->removePages();
```

Finding Pages in a Container

- **All finder methods return containers**
 - So you can iterate, count, etc.

Finding Pages in a Container

```
// findOneBy($property, $value)
$page = $container->findOneBy(
    'label', 'Zend Framework');

// findAllBy($property, $value)
$pages = $container->findAllBy(
    'tag', 'zendframework');

// Combine property name with method:
$page = $container->findOneByLabel(
    'Zend Framework');
$pages = $container->findAllByTag(
    'zendframework');
```

Iteration of Pages in a Container (Flat)

```
foreach ($container as $page) {  
    echo $page->label, "<br />\n";  
}
```

Iteration of Pages in a Container (Recursive)

```
$it = new RecursiveIteratorIterator(  
    $container,  
    RecursiveIteratorIterator::SELF_FIRST  
);  
foreach ($it as $page) {  
    echo $page->label, "<br />\n";  
}
```

Container Utility Methods

- **hasPage(Zend_Navigation_Page \$page)**
 - Does the container have the given page?
 - `if ($container->hasPage($page)) { /* do something */ }`
- **hasPages()**
 - Does the container have *any* pages?
 - Equivalent to `count($container) > 1`
- **toArray()**
 - Serialize container to array structure

Defining Pages

“

Options and strategies.

Methods for defining pages

- **Programmatically**
- **Using arrays**
- **Using Zend_Config values**

Strategies for defining pages

- **Cache the definitions**
 - First time through:
 - Create programmatically or from `Zend_Config`
 - Cache to array using `toArray()`
 - Subsequent times:
 - Load from cached array
- **Use only when needed**
 - Reduces performance hit
 - Disable on XHR calls, service calls, etc.

Navigation View Helper

“

Your gateway to using
Zend_Navigation in
the View layer.

Zend_View_Helper_Navigation

- **Holds a container**
- **Proxies to other Navigation view helpers**
- **Holds Translator and Acl objects**

Adding a Container to the Navigation helper

```
// Directly
$view->getHelper('navigation')->setContainer(
    $container
);

// Via "method call"
$view->navigation($container);

// Via registry
Zend_Registry::set(
    'Zend_Navigation', $container
);
```

Adding Internationalization (I18N)

- **Translatable metadata**
 - Labels
 - Titles

Injecting I18N into the Navigation helper

```
// Directly into view helper
$view->navigation()->setTranslator(
    $translator
);

// Using registry
// Has benefit of being used for all
// translation-aware ZF components
Zend_Registry::set(
    'Zend_Translate', $translator
);
```

Injecting ACLs into the Navigation helper (Directly)

```
// Inject ACL
$view->navigation()->setAcl($acl);

// Inject role
// $role may be either a string role name,
// or an object implementing
// Zend_Acl_Role_Interface
$view->navigation()->setRole($role);
```

Injecting ACLs into the Navigation helper (Registry)

```
// ACL:  
Zend_View_Helper_Navigation_HelperAbstract::setDefaultAcl(  
    $acl  
);  
  
// Role:  
Zend_View_Helper_Navigation_HelperAbstract::setDefaultRole(  
    $role  
);
```

Helpers that are ACL-aware...

- **can check for ACLs with hasAcl()**
- **can check for roles with hasRole()**
- **check against the “resource” and “privilege” registered with a page**
 - If either is missing, the check is skipped
 - If the ACL denies access, all sub-pages will be omitted as well

View Operations

“

How to use the various Zend_Navigation view helpers within your application.

Rendering links to pages (Basic usage)

```
// Find page, and render link
$page = $this->navigation()->findOneByLabel(
    'Foo'
);
echo $this->navigation()->htmlify($page);
```

Rendering conditional links to pages based on ACLs

```
// Conditionally render based on ACLs
echo (($this->navigation()->accept($page))
      ? $this->navigation()->htmlify($page)
      : '');
```

sitemap.xml

“

Generates output compatible with the Sitemaps XML Format, used by search engines to determine what pages on a site to crawl and index, and when to update this information.

Rendering a sitemap

```
echo $this->navigation()->sitemap();
```

Configuring the sitemap view helper

- **Indicate that output should be "formatted" (adds whitespace for readability):**
setFormatOutput(\$flag) (false)
- **Indicate whether or not to include the XML declaration:**
setUseXmlDeclaration(\$flag) (true)
- **Indicate a custom URL to prepend to links:**
setServerUrl(\$string)
- **Indicate the maximum depth to render:**
setMaxDepth(\$depth)

Sitemap-specific Page Metadata

- **lastmod**
 - Date of last modification of the file
 - Outputs using W3C Datetime format; time may be omitted
- **changefreq**
 - How frequently the page is likely to change
 - Values: always, hourly, daily, weekly, monthly, yearly, never
- **priority**
 - Priority of this URL relative to others on the site
 - Scale of 0.0 to 1.0 (1.0 being highest)

Sitemap – other behavior

- **Translation integration**
 - none; i18n elements are not present
- **ACL integration**
 - Any pages for which the user does not have permissions will not be included, nor their children

Breadcrumbs

“

Breadcrumbs indicate the ancestry of the current page within the sitemap.

Rendering breadcrumbs

```
echo $this->navigation()->breadcrumbs();
```

Configuring the breadcrumbs view helper

- **indentation: `setIndent($spaces)`**
- **set minimum breadcrumb depth: `setMinDepth($depth)`**
 - set to 1 (default) to omit root page
 - set to 0 to include root page
- **set maximum breadcrumb depth: `setMaxDepth($depth)`**
- **set breadcrumbs separator: `setSeparator($separator)`**
- **indicate whether to link last crumb in trail: `setLinkLast($flag)`**
- **indicate a view partial to use when rendering: `setPartial($script)`**

Breadcrumbs – other behavior

- **Translation integration**
 - Link text (labels) will be translated
- **ACL integration**
 - Any pages in the trail for which the user does not have permissions will not be linked, nor its children

Menus

“

Renders trees of links for use as navigational menus. Typically these will be unordered lists (UL elements) that you will then style with CSS to create a menu.

Configuring the menu view helper

- **Set the class for the UL element:
setUIClass(\$class)**
- **Indicate whether to render only pages and branches marked "active":
setOnlyActiveBranch(\$flag)**
- **Indicate whether to render parents when only rendering active branches:
setRenderParents(\$flag)**
- **Indicate a partial script to use to render the menu:
setPartial(\$script)**

Configuring the menu view helper: renderMenu()

- **Takes options array as second argument**
- **Options (in addition to setters):**
 - **indent:** either a string to indent with, or an integer number of spaces
 - **minDepth:** minimum depth to render; null indicates no minimum
 - **maxDepth:** maximum depth to render; null indicates no maximum

Rendering menus

- **By default, render the entire menu**
- **renderMenu() and renderSubMenu() allow specifying trees**
 - "null" argument will grab from active root node
 - otherwise pass in a Page, and that tree will be used

Rendering menus: Trees

```
// Full tree
echo $this->navigation()->menu();

// Selective tree
$subtree = $this->navigation()->findOneByLabel(
    'Zend Framework'
);
$options = array(
    'ulClass' => 'subtree',
);
echo $this->navigation()->menu()->renderMenu(
    $subtree, $options
);
```

Rendering menus: Deepest active menu

```
// Deepest active menu
echo $this->navigation()->menu()
                                ->renderSubMenu();

// Equivalent to:
echo $this->navigation()->menu()
                                ->renderMenu(
    null,
    array(
        'minDepth'           => null,
        'maxDepth'          => null,
        'onlyActiveBranch'  => true,
        'renderParents'     => false,
    )
);
```

Menus – other behavior

- **Translation integration**
 - Link text (labels) will be translated
- **ACL integration**
 - Any pages in the trail for which the user does not have permissions will not be included, nor its children

Links

“

Renders LINK elements that describe document relationships of the currently active page.

Link-specific Page Metadata

```
$page = new Zend_Navigation_Page_Mvc(array(
    'rel' => array(
        'alternate' => array(
            'label' => 'Example.org',
            'uri'    => 'http://www.example.org',
        ),
        'next' => array(
            'label' => 'Foo',
            'uri'    => '/foo',
        ),
        'prev' => array(
            'label' => 'Bar',
            'uri'    => '/bar',
        ),
    ),
    'rev' => array(
        'alternate' => 'http://www.example.net',
    ),
));
```

Rendering links

- **Typically from within your site layout**
- **All links**
 - `echo $this->navigation()->links();`

Selectively rendering links

```
$this->navigation()->links()->setRenderFlag(  
    Zend_View_Helper_Navigation_Links::RENDER_NEXT  
    | Zend_View_Helper_Navigation_Links::RENDER_PREV  
);  
echo $this->navigation()->links();
```

Links – other behavior

- **Translation integration**
 - Link labels will be translated
- **ACL integration**
 - Any linked pages for which the user does not have permissions will not be rendered

Conclusions

- **Powerful tool for SEO and user navigation**
- **I18n integration allows for flexible linking strategies**
- **ACL integration permits conditional site navigation**
- **Make it as complex or as simple as you want**
- **Cache, cache, cache**

Further Reading

- **Zend_Navigation manual:**
 - <http://framework.zend.com/manual/en/zend.navigation.html>
- **Zend_Navigation view helpers:**
 - <http://framework.zend.com/manual/en/zend.view.helpers.html#zend.view.helpers.initial.navigation>
- **Sitemaps XML Format**
 - <http://www.sitemaps.org/protocol.php>
- **Document relationships and the LINK element**
 - <http://www.w3.org/TR/html4/struct/links.html#h-12.3>

Thank You!

