Zend_Paginator - Matthew Ratzloff

<ac:macro ac:name="note"><ac:parameter ac:name="title">Under Construction</ac:parameter><ac:rich-text-body>
This proposal is being merged with
http://framework.zend.com/wiki/display/ZFPROP/Zend_Paginate+-+Jurri
ën+Stutterheim+and+Matthew+Ratzloff </ac:rich-text-body></ac:macro>

<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_Paginator - Matthew Ratzloff Component Proposal

Proposed Component Name	Zend_Paginator - Matthew Ratzloff
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Paginator - Matthew Ratzloff
Proposers	Matthew Ratzloff Darby Felton, Zend liaison
Revision	0.1 - First version (wiki revision: 8)

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
9. Class Skeletons

1. Overview

Zend_Paginator is a pagination class for both database- and view-level pagination. Users supply a page number and an optional configuration object and the rest is handled more or less seamlessly. The database strategy is the default, but view-level pagination can be selected in the options. For database pagination, users should have their models implement Zend_Paginator_Pageable_Interface.

Disclaimer: This is something I threw together over a couple of hours on Christmas Eve. I haven't coded anything yet, so I'm not sure if everything would work exactly as currently described. I really should probably finish all my other proposals and not-yet-proposals before I create another proposal, but this is something that a lot of people probably need right away.

2. References

I really wasn't impressed with any other pagination implementations I came across. In general, they were just too limiting.

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** paginate database and arbitrary result sets.
- This component **will** be flexible enough to allow users to implement any page-to-page navigation they like.
- This component **will not** be a one-size-fits-all solution. That means that there may be a little more work involved on the part of the user when initially incorporating this class into their project.

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Db
- Probably not Zend_View (but we'll see)

5. Theory of Operation

6. Milestones / Tasks

- Milestone 1: Proposal is finished.
- Milestone 2: Working prototype checked into the incubator supporting use cases #1, #2, ...
- Milestone 3: Working prototype checked into the incubator supporting use cases #3 and #4.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

7. Class Index

- Zend_Paginator
- Zend_Paginator_Exception
- Zend_Paginator_Page
- Zend_Paginator_Pageable_Interface
- Zend_Paginator_Strategy_Db
- Zend_Paginator_Strategy_View

8. Use Cases

UC-01

Database-level pagination example

Controller:

```
$pageNumber = $this->getRequest()->getParam('pageNumber');
$paginator  = new Zend_Paginator($pageNumber, $options);

$myModel = new MyModel();
$myModel->setPaginator($paginator);

$view->resultSet    = $myModel->fetchAll();
$view->pageSelector = $paginator->render('page_selector', $url);
```

Model (which implements Zend_Paginator_Pageable_Interface):

```
$select  = $db->select()->from('example');
$results = $this->_paginator->getRowset($select);
```

View (page_selector.php partial):

```
// A Google-like page selector

<?php if ($paginator->hasPages()): ?>
<div id="page-selector">
<!-- First page link -->
<?php if (!$paginator->current->isFirst()): ?>
  <a href="<?= $paginator->first->url; ?>">First</a> |
<?php else: ?>
  <span class="disabled">First</span> |
<?php endif; ?>

<!-- Previous page link -->
<?php if ($paginator->hasPrevious()): ?>
  <a href="<?= $paginator->previous->url; ?>">Previous</a> |
<?php else: ?>
  <span class="disabled">Previous</span> |
<?php endif; ?>

<!-- Numbered page links -->
<?php foreach ($paginator->getNearbyPages() as $page): ?>
  <a href="<?= $page->url; ?>"><?= $page->number; ?></a> |
<?php endforeach; ?>

<!-- Next page link -->
<?php if ($paginator->hasNext()): ?>
  <a href="<?= $paginator->next->url; ?>Next</a> |
<?php else: ?>
  <span class="disabled">Next</span> |
<?php endif; ?>

<!-- Last page link -->
<?php if (!$paginator->current->isLast()): ?>
  <a href="<?= $paginator->last->url; ?>Last</a>
<?php else: ?>
  <span class="disabled">Last</span>
<?php endif; ?>
</div>
<?php endif; ?>
```

UC-02

View-level pagination example

Controller:

```
// $data is an array-like object that comes from some external source
// $options includes a setting to use the view-level pagination strategy

$pageNumber = $this->getRequest()->getParam('pageNumber');
$paginator  = new Zend_Paginator($pageNumber, $options);

$view->results      = $paginator->getRowset($data);
$view->pageSelector = $paginator->render('page_selector', $url);
```

View (page_selector.php partial):

```
// A drop-down page selector

<?php if ($paginator->hasPages()): ?>
<div id="page-selector">
  <select size="1">
<?php foreach ($paginator->getAllPages() as $page): ?>
    <option value="<?= $page->number; ?>"><?= $page->number; ?></option>
<?php endforeach; ?>
  </select>
</div>
<?php endif; ?>
```

9. Class Skeletons

Here's a partial list. This includes the public methods.

```
class Zend_Paginator
    /** @var Zend_Paginator_Page $first First page */
    public $first = null;

    /** @var Zend_Paginator_Page $previous Previous page */
    public $previous = null;

    /** @var Zend_Paginator_Page $current Current page */
    public $current = null;

    /** @var Zend_Paginator_Page $next Next page */
    public $next = null;

    /** @var Zend_Paginator_Page $last Last page */
    public $last = null;

    /** @var string $rowRange Current row range (e.g., '51-100')
    public $rowRange = null;

    /** @var integer $totalRows Total rows
    public $totalRows = null;

    public function __construct($pageNumber, $options = null);
    public function setOptions($options);
    public function setPageNumber($pageNumber);
    public function render($view, $url = '');

    // The following are methods that may be split into another class
    public function hasPages();
    public function hasPrevious();
    public function hasNext();
    public function getNearbyPages(); // returns array
    public function getAllPages(); // returns array
```

```
class Zend_Paginator_Strategy_Db
    public function getRowset($select);
```

```
class Zend_Paginator_Strategy_View
    public function getRowset($data);
```

```
class Zend_Paginator_Page
    public $number = null;
    public $url    = null;

    public function __construct();
    public function isFirst();
    public function isLast();
```

```
interface Zend_Paginator_Pageable_Interface
    // This should probably specify the parameter type as Zend_Paginator_Interface
    public function setPaginator($paginator);
```

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