Confessions of an RPG Programmer: Why use Zend Framework?

Alan Seiden
Strategic Business Systems, Inc.

PHP/i consultant and developer
First certified ZF developer on IBM i

September 29, 2009
About Strategic Business Systems, Inc.

- **IBM partner since 1982**
  - IBM i (AS/400) hardware, software development, consulting
  - Concentration in food & beverage and automotive industries
  - HQ in northern New Jersey

- **Zend ("the PHP company") partner since 2008**
  - PHP’s been our preferred web technology for ourselves and clients since 2005
  - In addition to our consulting/development services, we offer Zend’s training and software
  - We represent Zend in the northeastern USA
We’ll be covering…

- What Zend Framework is
- Why ZF is a great match for the IBM i
- Intro to key concepts
- What ZF can do for your PHP/i projects
- How to get started!
What Zend Framework is

- A free, open source PHP framework
- A starting point for your PHP applications, providing
  - Modular design
  - Security features
- A collection of over 70 PHP components to simplify common tasks, including some for:
  - Form creation (and reuse)
  - Logging
  - Database access
- A demonstration of PHP 5 best practices
- It provides standards and great functionality but will not cramp your style. Your development is not limited in any way
Why ZF’s time is right

• **PHP is being used for critical apps on IBM i**

• **Managers, CIOs, technology architects are taking notice**

• **It’s time for professional practices**
  - Standards and consistency
  - Awareness of security
  - Reuse and easy maintenance of code
    - Leverage your software investments
  - Training and support
  - Doing it “right”

• **ZF gets you there—“Enterprise PHP”—faster—and keeps you in control**
Why I use it

• As I learn what it can do, the less boring code I write
  ▪ I can write less “plumbing” code

• Use ZF’s code however you like
  ▪ http://framework.zend.com/license
  ▪ Safe for corporate use

• It keeps up with trends and APIs
  ▪ Compatibility with diverse database systems, and APIs (authentication, web services, more)
Community

- **Contributors include individuals and companies.**
  Companies include:
  - Zend (of course)
  - IBM
  - OmniTI

- **Technology partners:**
  - Adobe, Google, IBM, Microsoft, nirvanix, Strikelron
Here’s why ZF reminds me of the i5 world

- Appreciation of standards: naming, parameter lists

- The tools you need are already integrated
  - Common components (template system, emailer, etc.) are there for you; no need to research/download/install
  - Upgrades like a “cume tape”—all components upgraded as a well tested unit

- ZF support available from Zend
  - Similar to phoning IBM about i5/OS
ZF’s birth, early years, and maturity on i5

• **2005: PHP Collaboration Project at ZendCon**
  - Started as collection of components but coalesced
  - PHP 5, object oriented (OO) from the start
  - Set example of OO design patterns and practices
    - More on OO later

• **2007-2009: Fast progress**
  - July 2007: GA version 1.0
  - Feb. 2009: version 1.70 with db2/i5 support
  - June 2009: version 1.82; minor releases every couple of weeks

• **April 2009: ZF/i application won COMMON’s “best web solution”**
### Allied Beverage Group: Wine catalog/ordering system on IBM i

**Search Results** (7 products found), order for EUROPA LIQUORS (001568)

**FRICING MONTH: June 2009**

#### Advanced Search

- **Search for item:**
  - [ ] (proof)
- **with bottle price ($):**
  - [ ] to [ ] (optional)
  - **Search**

#### Results for **Keywords: CHARD NAPA 375ML**

<table>
<thead>
<tr>
<th>Product</th>
<th>Code</th>
<th>Size</th>
<th>Pack</th>
<th>Qty</th>
<th>Csi/Bt</th>
<th>Add Items</th>
<th>Price</th>
<th>Inventory</th>
<th>Vintage</th>
<th>Info</th>
<th>$ Best Buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acacia Chardonnay</td>
<td>5607001</td>
<td>375 ML</td>
<td>12</td>
<td>3</td>
<td>cases</td>
<td></td>
<td></td>
<td>26 5</td>
<td>NV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cakebread Cellars Chardonnay Napa Valley 07</td>
<td>7433065</td>
<td>375 ML</td>
<td>12</td>
<td>1</td>
<td>bottles</td>
<td></td>
<td></td>
<td>9 5</td>
<td>2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grgich Hills Cellar Chardonnay</td>
<td>5544266</td>
<td>375 ML</td>
<td>12</td>
<td></td>
<td>cases</td>
<td></td>
<td></td>
<td>5 7</td>
<td>2006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Levendi Chardonnay Red Hen 05</td>
<td>4591060</td>
<td>375 ML</td>
<td>12</td>
<td></td>
<td>cases</td>
<td></td>
<td></td>
<td>13 2</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merryvale Chardonnay Starmont 07</td>
<td>4223069</td>
<td>375 ML</td>
<td>12</td>
<td></td>
<td>cases</td>
<td></td>
<td></td>
<td>8 8</td>
<td>2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schramsberg Blanc de Blanc</td>
<td>4056064</td>
<td>375 ML</td>
<td>12</td>
<td></td>
<td>cases</td>
<td></td>
<td></td>
<td>17 9</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Instant Intro to Object Orientation (2 slides!)
Here is an incredibly quick summary of OO, which you’ll see used throughout ZF

<table>
<thead>
<tr>
<th>OO Concept</th>
<th>Analogy in i5</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>a field in a data structure</td>
<td>$_orderNum</td>
</tr>
<tr>
<td>Method</td>
<td>function or subprocedure</td>
<td>isOrder()</td>
</tr>
<tr>
<td>Class</td>
<td>Imagine an intelligent data structure containing both data (properties) and programming logic (methods), which are both called “members” of the class</td>
<td>class Order { protected $_orderNum; function isOrder() { ... } ... }</td>
</tr>
</tbody>
</table>
• The arrow (\(\rightarrow\)) lets you access the members (methods and properties) of an object instance
  - $controller = $this\rightarrow get\_Request()\rightarrow get\_Controller\_Name();

• Sometimes you’ll also see the double colon (\(::\)), which is similar, but is used when a member is “static” (one per class)
  - echo Zend\_Registry::get('user');

• If you can read this notation, you can read ZF code. You will learn to appreciate its simplicity.
• **Autoloader**
  - PEAR convention for class/file names
    - Example: Search_Product = Search/Product.php
    - Put this in bootstrap file:
      ```php
      require_once 'Zend/Loader/Autoloader.php';
      $loader = Zend_Loader_Autoloader::getInstance()->
        setFallbackAutoloader(true);
      ```
    - Now you won’t need an “include” statement to do:
      ```php
      $prod = new Search_Product();
      ```

• **Fluent interface**
  ```php
  $select = $db->select()
    ->from( ...specify table and columns... )
    ->where( ...specify search criteria... )
    ->order( ...specify sorting criteria... );
  ```
Model-View-Controller Pattern
Model – View – Controller (MVC) design pattern

• You already know this pattern from RPG/DDS

• With green screens, IBM handles it under the covers, so you take it for granted

• On the web, you must define your application’s structure more explicitly

• Be patient…MVC seems strange at first, but you’ll soon realize that you’ve been here before…
MVC in detail

• **Model**
  - Reusable classes that access these resources:
    - Data
    - Business rules
  - Keep SQL and application details in one place

• **View**
  - Templates containing HTML or other output, with small bits of PHP
  - Plunk your HTML into a “view” without worrying about overwriting your mainline PHP code—helps web designers work with business programmers

• **Controller (action controller)**
  - Application flow
  - Connects model and view
  - Don’t confuse with “front controller,” which just initializes the MVC

• Next: MVC from an RPG perspective
RPG Model View Controller (MVC)
Confession

- For my first attempt with ZF, I put all my SQL in the controller
- It gave me a feeling of accomplishment
- The MVC police did not appear
- Later, I moved the SQL into a model class
  - Simplified the controller, which was getting complex and hard to understand
  - Made the SQL reusable
Initialize MVC
Front controller Routes “friendly” URL request

- **Default routing convention:**
  - `http://example.com/controller/action/param1/value1...`

  - **Controller maps to class name**
  - **Action maps to method name**
  - **Param/value pairs are passed to action**

  - **Bootstrap:** `index.php`

  - **Front Controller**

  - **Controller1**
    - action1()
    - action2()

  - **Controller2**
    - action1()
    - action2()
All requests routed through index.php in doc root

Document root is the only public folder.

index.php:
- initializes application
- instantiates Front Controller

On i5, I prefer to edit the Apache .conf file directly rather than add .htaccess files.
Apache configuration

Most tutorials suggest .htaccess, but I prefer to use the main PASE Apache config file (without proxy):

```
/usr/local/Zend/apache2/conf/httpd.conf
```

- **Listen 8000**
- **RewriteEngine on**
- **NameVirtualHost 10.11.12.13:8000**

```
<VirtualHost 10.11.12.13:8000>
    DocumentRoot /www/ebiz/htdocs/html
</VirtualHost>
```

```
<Directory /www/ebiz/htdocs/html/>
    # disallow .htaccess, so webserver won’t search for them
    AllowOverride None

    # funnel all requests to index.php
    # except requests for static resources
    RewriteEngine On
    RewriteRule !\.(js|ico|gif|jpg|png|css|html)$ index.php
</Directory>
```
<?php

// minimum bootstrap file (can be many variations)

// explicit, full paths save the i5 time searching for files
$paths = array(
    realpath(dirname(__FILE__) . '/../library'),
    realpath(dirname(__FILE__) . '/../application'),
    realpath(dirname(__FILE__) . '/../application/models'),
    '.');

set_include_path(implode(PATH_SEPARATOR, $paths));

// Prepare the front controller
$frontController = Zend_Controller_Front::getInstance();

// Dispatch the request using the front controller
$frontController->dispatch();
Action Controller
**Action Controller**

- **Controller classes handle groups of request URLs**
  
  http://example.com/{controller}/action
  
  Default: IndexController
  
  - Organizes and groups functionality
  
  - One class (extending Zend_Controller_Action) for each controller

- **Action methods in each controller class handle requests**
  
  http://example.com/{controller}/action
  
  Default: indexAction()
  
  - Named like `actionAction()`
    
    - Example: If `action` is “edit” then method is `editAction()`
More controller functionality

• **Several standard methods help organize and control the flow**
  - `init()` – called by the constructor
  - `preDispatch()` – called before the action’s method
  - `postDispatch()` – called after the action’s method

• **Utility methods**
  - `forward()`, `redirect()`, `getParam()`, `getRequest()`, `getResponse()`, `render()`

• **Action helpers add functionality**
  - Built-in helpers. Example: `gotoSimple()`
  - Your own helpers
  - Avoids the need to build your own base controller class
<?php

require_once 'Zend/Controller/Action.php';

class IndexController extends Zend_Controller_Action
{

    /**
     * The default action - show the home page
     */
    public function indexAction()
    {
        // Use default value of 1 if id is not set
        $id = $this->_getParam('id', 1);

        // assign id to view
        $this->view->id = $id;
    }
}
View
• **Scripts (templates)**
  - PHP-based script templates to present data
  - Should contain only display logic, not business logic
  - Default naming: “myaction.phtml”

• **Helpers**
  - Classes and methods that provide reusable view functionality
    - Examples of built in view helpers: `escape()`, `formText()`, `partial()`, `partialLoop()`, `headTitle()`
    - Write your own, too

• **Layout**

• **Placeholders**
What View means to you

• You can plunk HTML right into the view script and replace literals with PHP echo statements:
  - `<?php echo $this->productNum ?>`

• ZF provides smart defaults
  - The `$this->escape()` view helper uses PHP’s `htmlentities()` function, recommended by most security experts.
class Zend_View_Helper_Title_Case {

    public $view;

    public function titleCase($string = '')
    {
        return ucwords(strtolower(trim($string))));
    } // (public function titleCase())

    public function setView(Zend_View_Interface $view) {
        $this->view = $view;
    }
}

Usage:

    echo $this->titleCase('mozilla firefox');
    // Mozilla Firefox
Controller (again)…leads to view

```php
1<br>
2<session_once 'Zend/Controller/Action.php';
3
4class IndexController extends Zend_Controller_Action
5{
6   /**
7      * The default action - show the home page
8     */
9     public function indexAction()
10    {
11        // Use default value of 1 if id is not set
12        $id = $this->_getParam('id', 1);
13
14        // assign id to view
15        $this->view->id = $id;
16    }
17}
18```
<?php

3 echo '<?xml version="1.0" encoding="UTF-8" ?>';
4 echo $this->doctype();
5 ?>

7<html>
8   <head>
9     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
10 </head>
11 <body>
12   <h1><?php echo $this->placeholder('title') ?></h1>
13   <?php echo $this->layout()->content ?>
14 
15</body>
16 </html>
Zend_Layout

- **Two-step view pattern**
  - Uses Zend_View for rendering

- **Placeholders useful for setting javascript, titles, other variable data**

- **Layout view helper**
  - shortcut to layout placeholder
  - These are equivalent:
    ```php
    // fetch 'content' key using layout helper:
    echo $this->layout()->content;
    
    // fetch 'content' key using placeholder helper:
    echo $this->placeholder('Zend_Layout')->content;
    ```
Model
Models are abstract representations of data
- Can be extended from:
  - Zend_Db_Table_Row – For database abstraction
  - Zend_Feed_Element – For RSS abstraction
  - Or any other class that fits your needs
  - Or build your own abstract representations of your data

Model classes can contain business logic to prepare complex data for presentation

I stuff any “weird” code in models so that controllers/views are clean
// model: Busyflag.php

class Busyflag
{
    protected $name = 'SYSFLAGS'; // old-fashioned "System 36" table

    // isSiteUp: return true if up, false if down
    public function isSiteUp() {
        $sql = "select BZYFLG from {$this->name} where RECID = 'B'";
        $row = SBSDbhelp::getOneRow($sql);

        // true if Y, false otherwise.
        return $row['BZYFLG'] == 'Y';
    }

} // (public function isSiteUp())
} // (class Busyflag)

// usage (from a preDispatch front controller plugin)
$busyFlag = new Busyflag();
if (!$busyFlag->isSiteUp()) {
    // Take user to "site down" page.
} // (if (!$busyFlag->isSiteUp()))
Components
Reminder:

Zend/Db.php = Zend_Db

Zend/Db/Table.php = Zend_Db_Table
Forms
Zend_Form

- Creates the HTML for your data entry forms
  - $form = new Zend_Form();
  - $form->addElement('text', 'ordernum');
  - $form->addElement('text', 'date');

- Several ways to output form elements
  - echo $form; // (all elements) or
  - echo $form->ordernum; // (just ordernum) or
  - echo $form->getElement('ordernum');

- The HTML generated by that last echo
  - <input type="text" name="ordernum" id="ordernum">
// in a model:
class My_Form extends Zend_Form
{
    /* Create a text box that checks for non-letter characters **
    and converts text to lower case on submission */
    $form->addElement('text', 'username', array(
        'validators' => array(
            'alnum',
            array('regex', false, '/^[a-z]/i')
        ),
        'required' => true,
        'filters' => array('StringToLower'),
    ));
}

// in a controller:
$form = new My_Form();
$this->view = $form

// in a view:
echo $this->form;
Real life example of Zend_Form

Partial-description combo boxes (Dojo) on advanced search
### Search results

#### Advanced Search
- Search for item:
- with bottle price ($):
  - [ ] $0 to [ ] $ (optional)

#### Results for **Varietal: CHARDONNAY BLEND**

<table>
<thead>
<tr>
<th>Product</th>
<th>Code</th>
<th>Size</th>
<th>Pack</th>
<th>Price</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antinori Cervaro</td>
<td>6600543</td>
<td>750</td>
<td>6</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>Della Sala Chardonnay</td>
<td></td>
<td></td>
<td></td>
<td>$</td>
<td></td>
</tr>
<tr>
<td><strong>This product may be substituted with 6600544</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rodegas Julian</td>
<td>6784340</td>
<td>750</td>
<td>6</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>Chivite Navarra</td>
<td></td>
<td></td>
<td></td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>Reserva Coleccion 125</td>
<td></td>
<td></td>
<td></td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>Ca Montini Luna Di Luna Chardonnay/Pinot Grigio Gift Pack</td>
<td>4582449</td>
<td>750</td>
<td>6</td>
<td>$</td>
<td></td>
</tr>
</tbody>
</table>

**Pricing Month:** June 2009

Welcome, Angel Wong #105

---

Alan Seiden, Strategic Business Systems

RPG confessions/Zend Framework | 29-September-2009 | 47
// AdvancedSearchForm class is a model:
class AdvancedSearchForm extends Zend_Form {

    $prodId = new Zend_Form_Element_Text("prodid",
            array('size' => 7, 'maxlength' => 7, 'class' =>
                'width5'));

    $prodId->setRequired(false)
        ->addFilters(array("StripTags", "StringTrim"))
        ->addValidator(new Zend_Validate_Digits())
        ->setDescription("Partial product ID")
        ->setLabel("Code");

    $this->addElements(array($prodId));

} //(AdvancedSearchForm)
Database access
Database access with Zend_Db

- **Zend_Db** can create SQL for you. You don’t have to be an SQL expert to do everyday tasks.

- **Zend_Db** offers a lot beyond creating SQL:
  - Consistent quoting, escaping, prepared statements, profiler.

- Eventually, you should try to become proficient in SQL, both to understand what **Zend_Db** is doing, and for creating more complex queries.
• **Several classes give you a good start**
  - **Zend_Db_Adapter_Abstract**
    - Abstract class for all adapters
    - You will most likely use this or concrete implementations (such as Zend_Db_Adapter_Db2) for your database access
  - **Zend_Db_Table**
    - Gateway class for doing queries on a given table
  - **Zend_Db_Table_Row**
    - An instance of a given row
  - **Zend_Db_Statement**
Zend_Db_Table

- **Zend_Db_Table** gives you record-level access similar to what you may be used to.
  - **Insert**
    - `$products->insert(array(
        ‘prodid’ => ‘1234567’,
        ‘prodname’ => ‘sparkling water’,
    );
  - **Update**
  - **Find (like chaining with a key)**
    - `$results = $products->find(‘1234567’);
  - **Delete**
More Zend_Db examples for i5

```php
$driverOptions = array('i5_lib' => 'MYLIBRARY');
// Use 'driver_options' => array('i5_naming' => DB2_I5_NAMING_ON)) for liblists

$config = array(
    'host' => 'localhost',
    'username' => 'ALAN',
    'password' => 'secret',
    'dbname' => 'SBSDB',
    'driver_options' => $driverOptions);

$db = Zend_Db::factory('DB2', $config);

// Using "select" method to select and display records
$rows = $db->select()->from('CUSTOMERS')
    ->where('CUSTNO >= 0');

// or write your own SQL with parameters
$sql = 'SELECT * FROM CUSTOMERS WHERE CUSTNO > ? and CUSTNO < ?';
$rows = $db->fetchAll($sql, array(100, 2000));

// either way, output results
foreach ($rows as $row) {
    echo $row['CUSTNO'] . ' ' . $row['CUSTNAME'];
}
```
Config.ini lets you externalize Zend_Db settings

```
; config.ini
[dev]
db.adapter = PDO_MYSQL
db.params.username = alan
db.params.password = secret
db.params.dbname = devdb
db.params.host = 12.13.14.15

// in index.php (bootstrap file)
$config = new Zend_Config_Ini(realpath(dirname(__FILE__) . '/../application/config.ini'), 'dev');
$db = Zend_Db::factory($config->db);
```
Working with RPG
Use models to call RPG from ZF

• I always wrap RPG calls in a model class to simplify my code. Here’s why:
  ▪ If the RPG program’s name changes, or we call a different program (e.g. CL instead of RPG), I only need to change the model class, not every place it’s used
  ▪ Implement consistent error handling (e.g. level check)
  ▪ The model bridges the worlds of RPG and PHP
    • From PHP to RPG, zero-pad numbers
    • From RPG to PHP (return), interpret the RPG’s results
      • Convert ‘Y’ to ‘true’. Boolean values are well understood by PHP, can be evaluated by \texttt{if}($flag)…
Example of calling RPG from ZF

<table>
<thead>
<tr>
<th>Dlt</th>
<th>Qty</th>
<th>Cs/Bt</th>
<th>Code</th>
<th>Msg</th>
<th>Description</th>
<th>$ List</th>
<th>$ Disc</th>
<th>$ Net</th>
<th>$ Ext</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1</td>
<td>cases</td>
<td>3817220</td>
<td></td>
<td>JOSE CUERVO SILVER (1.75L)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>3</td>
<td>cases</td>
<td>5607061</td>
<td></td>
<td>A BY ACACIA CHD0712P (375ML/12)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>bottles</td>
<td>7433065</td>
<td></td>
<td>CAKEBREAD CHRD07 12P (375ML/12)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add 1 or more cases to qualify for the 2 case $20 RIM.

NOTE: Product is a half case qualifier.
class wer104
{
    public function__construct($sequence = 0) {
        // lots of code in here, conversions, error handling, etc.
        $parmsIn = array('PWEBID'=>$sessionKey);
        I5_command .. .

        $this->_isValidData = (($returnValues['PRTN'] == 'Y') ? true : false);
        // be very explicit, true or false

    }

    final public function isValidData()
    {
        return $this->_isValidData;
    }

} //(class wer104)
See how simple the controller code is

/* in controller, use model ‘wer104’
 * which wraps/calls RPG */
$validationCall = new wer104($sequence);

if (!$validationCall->isValidData()) {
    // validation failed; redirect to “edit”
    ...
}

// otherwise, we passed validation...
Paginator
Zend_Paginator

- Handles page-at-time logic, similar to subfiles, for large lists

- **Gives you:**
  - the right data records
  - Page numbering, back, next, first, last

- **For data, it’s commonly “fed” an array or db select object**
  - If database select, paginator is smart enough to read only the records to be displayed on the page
Example of Zend_Paginator code

**Controller**

```
$result = $db->select()->from("SLEMSTP");

$paginator = Zend_Paginator::factory($result);

// Set parameters for paginator
$paginator->setCurrentPageNumber($this->_getParam("page")); // URL must be something like:
          // http://example.com/orders/index/page/1  <- meaning we are currently on page one, and pass that
          // value into the "setCurrentPageNumber"
$paginator->setItemCountPerPage(20);
$paginator->setPageRange(10);

// Make paginator available in views
$this->view->paginator = $paginator;
```

**View script**

```
<?php if (count($this->paginator)): ?>
<ul><?php foreach ($this->paginator as $item): ?></li>
  <li><?= $item['LENAME1']; ?></li>
<?php endforeach; ?></ul><?php endif; ?>

<?= $this->paginationControl($this->paginator, 'Sliding', 'partials/paginationcontrol.phtml'); ?>
```
Example of Zend_Paginator code

View Partial (used in View Script on previous slide)
NOTE these view helpers: $this->url which build URL links with the prev, next, and other page numbers, and leads back to controller with the page clicked by user.

```php
<?php echo sprintf('Page %s of %s', $this->current, 'xxx'); ?>
<?php if ($this->pageCount): ?>
    <div class="paginationControl">
        <!-- Previous page link -->
        <?php if (isset($this->previous)): ?>
            <a href="<? = $this->url(array('page' => $this->previous)); ?>">&lt; Previous</a> |
        <?php else: ?>
            <span class="disabled">&lt; Previous</span> |
        <?php endif; ?>
        <!-- Numbered page links -->
        <?php foreach ($this->pagesInRange as $page): ?>
            <?php if ($page != $this->current): ?>
                <a href="<? = $this->url(array('page' => $page)); ?>"><? = $page ?> </a> |
            <?php else: ?>
                <? = $page ?> |
            <?php endif; ?>
        <?php endforeach; ?>
        <!-- Next page link -->
        <?php if (isset($this->next)): ?>
            <a href="<? = $this->url(array('page' => $this->next)); ?>">Next &gt;</a>
        <?php else: ?>
            <span class="disabled">Next &gt;</span>
        <?php endif; ?>
    </div>
<?php endif; ?>
```
Zend_Paginator display

(The appearance can be fully customized by changing the View and View Partial scripts)

- Piccolo Tuesday Men’s "D"
- Ramapo Valley White Sox
- Huntsville International League
- Streetsboro Flames
- Fontana Community Little League
- Test
- The Bandits
- Black Hat
- Brookline Mens Softball
- Intensity
- Lou Gehrig League
- Mel Ott League
- Roy Campanella League (AAA)
- Roy Campanella League (A)
- patch
- Spring League 2001
- Lehigh Valley MSBL
- Lancaster Depew Leagues
- Roy Campanella
- Spartans

url: http://example.com/leagues/index/page/5
Other components you’ll like
Other components

- Auth
- ACL
- Filter/Validate
- Log (with familiar concept of logging levels)
- Navigation (bread crumbs)
How to start a ZF project?
Start the right way with Zend Studio for Eclipse

- Creates a complete “hello world” application for you
  - Leverage the ZF development team’s best practices
Resources: online

• **Official information:**
  - framework.zend.com/docs/quickstart
  - zend.com/resources/webinars

• **Community tutorials and answers:**
  - zfforums.com
  - devzone.zend.com
Path to ZF

• **Jump in**
  - Have a pilot project in mind
  - Take a ZF training class
  - Get mentoring from someone savvy in both ZF and “i”

• **Stay connected**
  - Join a ZF community, either online or a Meetup in person
  - Subscribe to Zend’s ZF support if it’s a mission-critical app
  - Write to me for guidance: aseiden@sbsusa.com
Questions and Thanks

Alan: aseiden@sbsusa.com

Leave a comment: alanseiden.com/presentations