Zend Framework in 2009

Alan Seiden – Strategic Business Systems
alan@alanseiden.com

New York PHP
24-Feb-2009
Speaker

Alan Seiden: alan@alanseiden.com
PHP and Zend Framework Consultant
Strategic Business Systems, Inc.

Zend Certified Engineer—Zend Framework

Contributor: Mantis/400 & IBM’s Zend Core for i5/OS
Writer about PHP for IBM i5 community
Mentor for teams learning Zend Framework
Tonight we will cover:

• **Zend Framework (ZF) basics**
  ▪ What, why, how

• **ZF’s flexible model-view-controller (MVC) implementation**
  ▪ Keeps your code organized logically and handles “plumbing”

• **Components (libraries)**
  ▪ Including classes to access web services

• **What’s new and what’s next**
What is ZF?

• An open source, MVC-based PHP framework

• Is it a framework? Yes
  ▪ "Glue" to build applications
  ▪ Components developed, tested, distributed together

• But loosely coupled
  ▪ “Use at will” architecture
  ▪ Dependencies documented in manual
    • Section A.1.4. Zend Framework Dependencies
Birth and early years

• **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

• **July 2007: GA version 1.0**

• **February 17, 2009: version 1.75**

• **Frequent minor releases**
What’s it made of?

• Robust, high quality object-oriented PHP 5 libraries
  ▪ Test-driven development
    • 80+% code coverage
  ▪ Peer review of all code
  ▪ Coding standards
  ▪ Components not released until documented in manual and inline
    • phpDocumentor format: http://phpdoc.org
Flexible License

- “New BSD” license
  - http://framework.zend.com/license
  - Use code however you like

- Apache-like contributor license agreement (CLA)
  - Safety for users/corporations
  - All original code; no patent claims
Why I use it

• As I learn what it can do, the less boring code I write
  ▪ At first, I reinvented the wheel
  ▪ Realized that ZF already provided functionality
  ▪ I can write less “plumbing” code

• Can customize and extend
  ▪ Numerous integration points
  ▪ Interfaces and small methods give you control

• It keeps up with trends and APIs
  ▪ Compatibility with diverse database systems, authentication and other APIs
  ▪ Including IBM i (AS/400)’s unique version of db2
  ▪ Web services
Community

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

- **Technology partners:**
  - Adobe, Google, IBM, Microsoft, nirvanix, StrikeIron

- **Help available at zfforum.com, e-mail lists, #zftalk IRC**
Requirements for Zend Framework 1.75

- **Required:**
  - PHP 5.2.4+

- **Optional:**
  - `mod_rewrite`
    - Recommended; route URLs through bootstrap file
  - Extensions as needed. Examples:
    - `ext/session` in Zend_Session
    - `ext/gd` in Zend_Captcha
  - PHPUnit 3.3.0 if use Zend_TestPHPUnit

- **Full list of reqs and dependencies:**
How it’s developed
Issue tracker

Zend Framework

get a list of all supported db adapters

Created: 04/Jan/07 07:14 AM  Updated: 04/Dec/08 01:00 PM

Component/s:  Zend_Db
Affects Version/s:  0.8.0
Fix Version/s:  Next Minor Release

Time Tracking:

Original Estimate:  1 day
Remaining Estimate:  1 day
Time Spent:  Not Specified

Public Fields

Internal Project Management Fields

Tags:  
Resolved Date:  04/Dec/08 01:00 PM
Participants:  Bill Karwin, Marc Bennewitz, Ralph Schindler and Wil Sinclair

Description

Hi,

add static method: Zend_Db::listSupportedAdapters(void)
return array of all database adapters which supported by Zend_Framework and installed php

```php
array(
    '<adapter>' => '<full name>'
    [, '<adapter>' => '<full name>', ...]
)`
Timesaving conventions

• **Autoloader**
  - PEAR convention for class/file names
    - Example: Search_Product = Search/Product.php
    - Put this in bootstrap file:
      ```php
      require_once 'Zend/Loader.php';
      Zend_Loader::registerAutoload();
      ```
    - Now you can 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
Model – View – Controller design pattern

• **Model**
  - Classes that access your data

• **View**
  - Templates to present that data (e.g. HTML for browser)

• **Controller (action controller)**
  - Application flow
  - Connects model and view

• (Bonus: front controller!)
Front Controller
Front controller pattern

• Front controller sits in front of MVC

• All PHP requests funneled through index.php (bootstrap file)

• Front controller gets your application started
  ▪ Initializes request/response objects
  ▪ Can handle common settings and functionality
    • “Include” paths
    • Configurations
    • Location of MVC components (if necessary)
    • Logging, db (perhaps), authentication/authorization
  ▪ Converts URL to a “request” object with distinct parts
  ▪ Routes requests to appropriate action controllers

• Receives exceptions
Front controller to action controller
Routes 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
Front controller needs two files in public folder

In document root (public folder):

1. `.htaccess` redirects requests to bootstrap script (`index.php`)

2. `index.php` instantiates Front Controller
Front controller file #1: .htaccess

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

// bootstrap file

setincludepath('.\PATHSEPARATOR\..\library\PATHSEPARATOR\..\application\default\default\models\PATHSEPARATOR\includepath());

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

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

```php
<?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;
    }
}
```
Action helpers

- They extend Zend_Controller_Action_Helper_Abstract

- Built-in action helpers
  - ActionStack
  - AjaxContext
  - AutoComplete: Dojo, Scriptaculous
  - ContextSwitch
  - FlashMessenger
  - Json
  - Redirector
  - Url
  - ViewRenderer

- Create your own
  - Place in the "My/Helper/" directory of your library (or any directory on your includepath)
Action helper example: Redirector gotoSimple()

```php
class Forward_Controller extends Zend_Controller_Action
{
    protected $redirector = null;

    public function init()
    {
        $this->redirector = $this->helper->getHelper('Redirector');
    }

    public function myAction()
    {
        /* do some stuff */

        // Redirect to 'my-action' of 'my-controller' in the current module, using the params param1 => test and param2 => test2
        $this->redirector->gotoSimple('my-action',
            'my-controller',
            null,
            array('param1' => 'test',
                   'param2' => 'test2'
            );
    }
}
```
View
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

• **Output filters**

• **Layout**

• **Placeholders**
Controller (again)...leads to view

```php
<?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 script automatically rendered

```php
<?php

/**
 * Home page view
 */
@version 1.00

$this->pageTitle('Zend Framework Demo');
$this->placeholder('title')->set('Welcome');

Welcome to the home page's simple view script.<BR>
Your id is <?php echo $this->escape($this->id) ?>.
```

Welcome

Welcome the home page's simple view script.
Your id is 1.

Welcome

Welcome to the home page's simple view script.
Your id is 39.
Zend_Layout

```php
<?php

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

7 <html>

   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <?php
       echo $this->headTitle();
       echo $this->headScript();
       echo $this->headStyle();
      ?>
   </head>

   <body>
      <h1><?php echo $this->placeholder('title') ?></h1>
      <?php echo $this->layout()->content ?>
      <br />
      <br />
   </body>

26 </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;
    ```

- **Enable in bootstrap**
  ```php
  // accept default path
  Zend_Layout::startMvc();
  
  // or specify path
  $layoutPath = realpath(dirname(__FILE__) . '../application/layouts');
  Zend_Layout::startMvc(array("layoutPath" => $layoutPath));
  ```
<?php
Require_once 'Zend/View/Interface.php';

/**
 * TitleCase helper
 */
class Zend_View_Helper_Title_Case {

    public $view;

    public function titleCase($string = '')
    {
        // convert incoming string so that
        // first letter of each word is capitalized/
        // but all other letters are lowercase.
        // Also trim the string.
        return ucwords(strtolower(trim($string)));
    }

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

Usage:

    echo $this->titleCase('mozilla firefox');
    // Mozilla Firefox
Model
• **Models are abstract representations of data**
  
  - Can be extended from:
    - `Zend_Db_Table_Row` – For database abstraction
    - `Zend_Feed_Element` – For RSS abstraction
    - `Yahoo_Result` – For Yahoo abstraction
    - Or any other class that fits your needs
    - Or build your own own abstract representations of your data
Models (more)

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

- I stuff any “weird” code in models so that controllers/views are clean
<?php

// model: Busyflag.php

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

    // isSiteUp: return true if up, false if down
    public function isSiteUp() {

        // one record, with key "B"
        $sql = "select BZYFLG from {$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.
    $request->setControllerName("Down");
    $request->setActionName("index");
} // (if (!$busyFlag->isSiteUp()))
Controller + Model + View: traditional

```php
// Controllers/SearchController.php
require_once 'Zend/Controller/Action.php';
class SearchController extends Zend_Controller_Action
{
    public function indexAction()
    {
        $prodid = $this->_getParam('prodid');

        // look up product data in model
        $products = new Search_Products();
        $results = $products->find($prodid);

        // assign results to view
        $this->view->results = $results;
    }
}
```

```php
// views/scripts/search/index.phtml

<?php
Products found:<ul>
<?php
foreach ($this->results as $result) {
    echo "<li>{$this->escape($result->prodname)}</b>{$this->escape($result->available)}</li>";
}
?>
</ul>
```
MVC your way: custom routing rules

• Can declare custom routing rules with Zend_Controller_Router_Route classes
  - Not limited to “controller/action/param” format
  - Example: “year/month/day/title” as in a blog posting:
  - URL: http://myblog.com/2008/02/24/i-like-routers
  - $route = new Zend_Controller_Router_Route(
      ':year/:month/:day/:title',
      array(
        'year' => 2009,
        'controller' => 'articles',
        'action' => 'show'
      ),
      array(
        'year' => '\d+',
        'month' => '\d+',
        'day' => '\d+',
      )
    );
  $router->addRoute('posts', $route);
MVC your way: multi-MVC

- Can separate app sections into modules (Mitch P.’s “atomic MVC”)
  
  ```php
  $front->setControllerDirectory(array(
    'default' => '/demo/application/controllers',
    'blog' => '/demo/application/blog/controllers'
  ));
  ```

- **Blog_IndexController** is class name of index action controller within blog app/module
  - URL: example.com/blog or /blog/index or /blog/index/index

- **IndexController** is still class name within default app/module
  - URL: / or /index or /index/index
Tools can help get you started

- Zend Studio for Eclipse creates default directory structures and classes for you
Library of Zend components

Reminder:
Zend/Db.php = Zend_Db
Zend/Db/Table.php = Zend_Db_Table
Components vs. straight PHP

- **Additional functionality**
  - Zend_Session vs. straight $_SESSION
  - Zend_Debug::dump() vs. vardump

- **Thread-safety**
  - Zend_Translate vs. gettext
  - Zend_Locale vs. setlocale

- **OO**
  - OO Syntax
  - May be extended with custom functionality
  - ZF components reduce coding when used together

- **They’ve been tested and integrated for you**
## Components in 1.75

<table>
<thead>
<tr>
<th>Zend_Acl</th>
<th>Zend_Amf</th>
<th>Zend_Flickr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zend_Auth</td>
<td>Zend_Amf</td>
<td>Zend_Nirvanix</td>
</tr>
<tr>
<td>Zend_Cache</td>
<td>Zend_Amf</td>
<td>Zend_ReCaptcha</td>
</tr>
<tr>
<td>Zend_Captcha</td>
<td>Zend_Amf</td>
<td>Zend_Simpy</td>
</tr>
<tr>
<td>Zend_Config</td>
<td>Zend_Amf</td>
<td>Zend_SlideShare</td>
</tr>
<tr>
<td>Zend_Config_Writer</td>
<td>Zend_Amf</td>
<td>Zend_Strikelron</td>
</tr>
<tr>
<td>Zend_Console_Getopt</td>
<td>Zend_Amf</td>
<td>Zend_Technorati</td>
</tr>
<tr>
<td>Zend_Controller</td>
<td>Zend_Amf</td>
<td>Zend_Twitter</td>
</tr>
<tr>
<td>Zend_Currency</td>
<td>Zend_Amf</td>
<td>Zend_Yahoo</td>
</tr>
<tr>
<td>Zend_Date</td>
<td>Zend_Amf</td>
<td>Zend_Session</td>
</tr>
<tr>
<td>Zend_Db</td>
<td>Zend_Amf</td>
<td>Zend_Soap</td>
</tr>
<tr>
<td>Zend_Debug</td>
<td>Zend_Amf</td>
<td>Zend_Test</td>
</tr>
<tr>
<td>Zend_Dojo</td>
<td>Zend_Amf</td>
<td>Zend_Text</td>
</tr>
<tr>
<td>Zend_Dom</td>
<td>Zend_Amf</td>
<td>Zend_Timesync</td>
</tr>
<tr>
<td>Zend_Exception</td>
<td>Zend_Amf</td>
<td>Zend_Translate</td>
</tr>
<tr>
<td>Zend_Feed</td>
<td>Zend_Amf</td>
<td>Zend_Uri</td>
</tr>
<tr>
<td>Zend_File</td>
<td>Zend_Amf</td>
<td>Zend.Validate</td>
</tr>
<tr>
<td>Zend_Filter</td>
<td>Zend_Amf</td>
<td>Zend_Version</td>
</tr>
<tr>
<td>Zend_FilterInput</td>
<td>Zend_Amf</td>
<td>Zend_View</td>
</tr>
<tr>
<td>Zend_Form</td>
<td>Zend_Amf</td>
<td>Zend_Wildfire</td>
</tr>
<tr>
<td>Zend_Gdata</td>
<td>Zend_Amf</td>
<td>Zend_XmlRpc</td>
</tr>
<tr>
<td>Zend_Http</td>
<td>Zend_Amf</td>
<td>Zend_XConsoleProcessUnix</td>
</tr>
<tr>
<td></td>
<td>Zend_Amf</td>
<td>Zend_XJQuery</td>
</tr>
</tbody>
</table>
Component categories

- MVC
- Formatting
- Ajax
- Identity, Authentication, and Authorization
- Forms
- Datastore
- Web Services
- Enterprise
- Debugging, Logging, and Testing
- I18n and L10n
- Mail
- Infrastructure
MVC components (you already know these)

- **Zend_Controller**
- **Zend_Layout**
- **Zend_View**
Formatting

- Zend_Text
- Zend_Paginator
Ajax

- **Zend_Dojo**
- **Zend_Json**
Identity, authentication, and authorization

- Zend_Acl
- Zend_Auth
- Zend_Infocard
- Zend_OpenId
- Zend_Session
Form and input validation

- Zend_Captcha
- Zend_Form
- Zend_Validate
- Zend_Filter
Database access

- **Zend_Db**
- **Zend_Db_Table**
- **Zend_Db_Profiler**
Web services

- Zend_Feed
- Zend_Gdata
- Zend_Http
- Zend_Rest

- Zend_Server_Reflection
- Zend_Soap
- Zend_Uri
- Zend_XmlRpc
Zend_Service_* clients

- Akismet
- Amazon
- Audioscrobbler
- Delicious
- Flickr
- Nirvanix
- ReCaptcha
- Simpy
- SlideShare
- Strikelon
- Technorati
- Twitter
- Yahoo
Additional categories

- Zend_Ldap
- Zend_Search_Lucene
- Zend_Pdf
Debugging, logging, and testing

- Zend_Debug
- Zend_Log
- Zend_Test
- Zend_Wildfire
I18n and L10n

- Zend_Currency
- Zend_Calendar
- Zend_Date
- Zend_Locale
- Zend_Measure
- Zend_TimeSync
- Zend_Translate
Mail

- Zend_Mail
- Zend_Mime
Infrastructure

- Zend_Cache
- Zend_Config
- Zend_Console
- Zend_File
- Zend_Loader
- Zend_Memory
- Zend_Registry
- Zend_Version
Registry
Registry

• Store global data without polluting the global scope
  ▪ `Zend_Registry::set('key', $value);`
  ▪ `Zend_Registry::get('key');`
  ▪ Useful for storing sessions, configuration data or any data that could potentially be important on a per-request basis
Logging
Logging

- **Structured, flexible logging class**

- **Zend_Log** has several backends
  - Stream (write to a file)
  - Firebug
  - Mail
  - Db

- **Example of logging to file system**
  ```php
  $writer = new Zend_Log_Writer_Stream('/path/to/logfile');
  // Only email warning level entries and higher.
  $writer->addFilter(Zend_Log::WARN);
  $logger = new Zend_Log($writer);

  // later...
  $logger->info('Informational message');
  $logger->err('Error message');

  // “Error message”
  ```
Logging: Firebug backend

- **Zend_Log_Writer_Firebug, Zend_Db_Profiler_Firebug**

- **Install Firebug and FirePHP in Firefox**

  ```php
  // in bootstrap
  $logger = new Zend_Log();
  $writer = new Zend_Log_Writer_Firebug();
  $logger->addWriter($writer);
  Zend_Registry::set('logger',$logger);

  // in action controller or anywhere
  $logger = Zend_Registry::get('logger');
  $logger->log('Log message', Zend_Log::DEBUG);
  $logger->log('Info message', Zend_Log::INFO);
  $logger->log('Warn message', Zend_Log::WARN);
  $logger->log('Error message', Zend_Log::ERR);
  ```
Config
Zend_Config

- OO syntax for reading/writing config files
- Internally it’s an array
- Multiple backends
  -INI
  -XML

- Divide sections with [ ], good for dev/prod
  - And hierarchy with periods after that
    - db.params.username = xxxx

- Many components accept Zend_Config object as well as array
  -Db
  -Forms
Config.ini:
[dev]
db.adapter = PDO_MYSQL
db.params.username = admin
db.params.password = xxxx
db.params.dbname = appdb
db.params.host = 192.168.9.1

log.filename = /appdev/logs/app.log

$config = new Zend_Config_Ini(realpath(dirname(__FILE__) . '/..//application/config.ini'), 'dev');
$registry = Zend_Registry::getInstance();
$registry->set('config', $config);

$db = Zend_Db::factory($config->db);
Zend_Db_Table::setDefaultAdapter($db);
$registry->set('db', $db);

// create logger; store it in registry
$logFile = $registry->get('config')->log->filename;
$writer = new Zend_Log_Writer_Stream($logFile);
$logger = new Zend_Log($writer);
$registry->set('logger', $logger);
Sessions
Sessions

- **Zend_Session does more than PHP’s ext/session**
  - Namespaces let multiple applications (or sections of application) run in the same $_SESSION without risking key name collision
  - Used by Zend_Auth by default
  - Can set data to expire based on time or clicks/requests

```php
$s = new Zend_Session_Namespace('myNamespace');
$s->a = 'apple';
$s->p = 'pear';
$s->o = 'orange';

$s->setExpirationSeconds(5, 'a'); // expire only the key "a" in 5 seconds

// expire entire namespace in 5 "hops"
$s->setExpirationHops(5);

$s->setExpirationSeconds(60); // The "expireAll" namespace will be marked "expired" on
// the first request received after 60 seconds have elapsed,
// or in 5 hops, whichever happens first.
```
Authentication
Authentication

• **Built over an adapter that implements** `Zend_Auth_Adapter_Interface`
  - Must have an `authenticate()` method that returns a `Zend_Auth_Result` object

• **Several pre-defined adapter classes can authenticate against common data stores**
  - `Zend_Auth_Adapter_Db_Table`, `Digest`, `Http`, `Ldap`, `Openid`

• **`Zend_Auth_Storage_Session` uses a session namespace of "Zend_Auth" by default**
Authentication example

// created db adapter earlier
$dbAdapter = Zend_Registry::get('db');
$authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
$authAdapter
    ->setTableName('users')
    ->setIdentityColumn('username')
    ->setCredentialColumn('password');

// Perform the authentication query, saving the result in $_SESSION
$result = $authAdapter->authenticate();
if ($result->isValid) {
    echo 'logged in';
} else {
    echo 'failed';
}

// Print the result row
print_r($authAdapter->getResultRowObject());

/* Output: 
  my_username
  
  Array
  ( 
    [id] => 1
    [username] => my_username
    [password] => my_password
    [real_name] => My Real Name
  ) */
Authorization
Authorization

- **Classes used**
  - Zend_Acl_Role
  - Zend_Acl_Resource

- **Three concepts**
  - **Role**
    - Assigns particular groups of access to an individual
  - **Resource**
    - An object to which access is controlled
    - Resources can have privileges (e.g., "create", "read", "update", "delete")
  - **Access Control List (ACL)**
    - A hierarchical list that connects role/resource permissions
Authorization

```php
$acl = new Zend_Acl();

$acl->addRole(new Zend_Acl_Role('guest'))
    ->addRole(new Zend_Acl_Role('member'))
    ->addRole(new Zend_Acl_Role('admin'));

$parents = array('guest', 'member', 'admin');
$acl->addRole(new Zend_Acl_Role('someUser'), $parents);

$acl->add(new Zend_Acl_Resource('someResource'));

$acl->deny('guest', 'someResource');
$acl->allow('member', 'someResource');

echo $acl->isAllowed('someUser', 'someResource') ?
    'allowed' : 'denied';
```
Forms
Zend_Form

- Flexible solution for building forms

- Create in code
  - Put it in a model class

- Or create in config file

- Factory pattern lets you determine element type at runtime

- Pass $form to view script, where it’s output
  - echo $form; or
  - echo $form->ordernum; or
  - echo $form->getElement('ordernum');

- Decorators
class My_Form extends Zend_Form
{
    $form->addElement('text', 'username', array(
        'validators' => array(
            'alnum',
            array('regex', false, '/^[a-z]/i')
        ),
        'required' => true,
        'filters' => array('StringToLower'),
    ));
}

// in your controller...
$form = new My_Form();
$this->view = $form

// in your view...
cho $this->form;
AJAX/Dojo
Since May 2008

Integration points:
• JSON-RPC Server
• dojo.data Envelopes
• Dojo View Helper
• Dijit integration with Zend_Form & Zend_View
• Dojo Library Re-distribution

• Also JQuery in extras folder
class My_Form extends Zend_Dojo_Form
{
    protected $_selectOptions = array(
        'red'    => 'Rouge',
        'blue'   => 'Bleu',
        'white'  => 'Blanc',
        'orange' => 'Orange',
        'black'  => 'Noir',
        'green'  => 'Vert',
    );

    $this->addElement(
        'FilteringSelect',
        'filterselect',
        array(
            'label' => 'FilteringSelect (select)',
            'value' => 'blue',
            'autocomplete' => false,
            'multiOptions' => $this->_selectOptions,
        )
    );
}
Databases
Databases

• **Adapters for various database drivers**
  - IBM DB2/informix (PDO)
  - IBM DB2 and DB2/i5 (IBM i) (non-PDO)
  - Firebird/Interbase (non-PDO)
  - MS SQL (PDO)
  - MySQL (PDO) and mysqli (non-PDO)
  - Oracle (PDO and non-PDO)
  - PostgreSQL (PDO)
  - SQLite (PDO)
  - or build your own by extending Zend_Db_Adapter_Abstract
```php
$db = new Zend_Db_Adapter_Pdo_Mysql(array(
    'host'    => '127.0.0.1',
    'username' => 'webuser',
    'password' => 'xxxxxxxxx',
    'dbname'   => 'test'
));
```
Databases

• **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_Pdo_Mysql) 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 examples

// 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'];
}
class Orders extends Zend_Db_Table_Abstract
{
    protected $_name = 'orders';
}

// instantiating your class
$db = Zend_Db::factory('PDO_MYSQL', $options);

// don’t need to specify db if used setDefaultAdapter method earlier
// Zend_Db_Table_Abstract::setDefaultAdapter($db);

$table = new Orders(array('db' => $db));

$data = array(
    'custid' => '12345',
    'custname' => 'Irving Jones',
);

$table->insert($data);
Caching
Caching

• **Frontend**
  - Core
  - Output
  - Function
  - File
  - Class

• **Backend**
  - Apc
  - File
  - Memcached
  - Sqlite
  - ZendPlatform
Caching

```php
$frontendOptions = array(
    'lifetime' => 7200
);
$backendOptions = array(
    'cache_dir' => '../application/cache/'
);

$cache = Zend_Cache::factory('Output', 'File',
    $frontendOptions, $backendOptions);

if (!$cache->start('name-of-cached-item')) {
    // produce output
    $cache->end();
}
```
Unit Tests
Zend_Test_PHPUnit

• **Extend Zend_Test_PHPUnit_ControllerTestCase**

```php
class User_Controller_Test extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...

    public function testCallWithoutActionShouldPullFromIndexAction()
    {
        $this->dispatch('/user');
        $this->assertController('user');
        $this->assertAction('index');
    }

    public function testLoginFormShouldContainLoginAndRegistrationForms()
    {
        $this->dispatch('/user');
        $this->assertQueryCount('form', 2);
    }
}
```
More testing examples

- Valid response codes and login

```php
public function testIndexActionShouldContainLoginForm()
{
    $this->dispatch('/user');
    $this->assertResponseCode(200);
    $this->assertSelect('form#login');
}

public function testValidLoginShouldInitializeAuthSessionAndRedirectToProfilePage()
{
    $this->request
        ->setMethod('POST')
        ->setPost(array(
            'username' => 'foobar',
            'password' => 'foobar' )
    );
    $this->dispatch('/user/login');
    $this->assertTrue(Zend_Auth::getInstance()->hasIdentity());
    $this->assertRedirectTo('/user/view');
}
```
Validation
Validation

• **Uses the Zend.Validate* classes**
  
  ```php
  $check = new Zend.Validate.Alnum();
  if ($check->isValid($GET['data'])) {
    // do stuff
  }
  ```

• **Each class extends the Zend.Validate_Interface interface**
  - You can use the internal validation classes or build your own
Validation

• Pre-defined classes

  ▪ Alnum
  ▪ Alpha
  ▪ Between
  ▪ Ccnum
  ▪ Date
  ▪ Digits
  ▪ EmailAddress
  ▪ Float
  ▪ GreaterThan

  ▪ Hex
  ▪ Hostname
  ▪ InArray
  ▪ Int
  ▪ Ip
  ▪ LessThan
  ▪NotEmpty
  ▪ Regex
  ▪ StringLength
Web Services
Client for web services

- **Interfaces into web service providers**
  - Example: Google data
    - Calendar
    - Docs
    - Exif
    - Feed
    - Gapps
    - Gbase
    - Geo
    - Media
    - Photos
    - Query
    - Spreadsheets
    - YouTube
Client classes for web services

- Akismet
- Amazon
- Audioscrobbler
- Delicious
- Flickr
- Nirvanix
- ReCaptcha
- Simpy
- SlideShare
- Strikelron
- Technorati
- Twitter
- Yahoo
Zend_Service_Yahoo

• **Search the web with Yahoo**
  - Get your application ID from http://developer.yahoo.com/wsregapp/
  - Class uses Zend_Rest_Client under the covers
  - Returns Zend_Service_Yahoo_WebResultSet containing instances of Zend_Service_Yahoo_WebResult

```php
$yahoo = new Zend_Service_Yahoo("YAHOO_APPLICATION_ID");
$results = $yahoo->webSearch('IBM PHP',
                              array('results' => '10',
                                    'start' => 1));

foreach ($results as $result) {
    echo '<b>' . $result->Title . '</b> ' . $result->Url . '<br />';
}
```
Results from $yahoo->webSearch

Informed Networker - Social News for IT Professionals. - IBM, Zend ... http://www.informednetworker.com/other/ib
platform/
:: News : PHP : Zend Core for IBM on Linux http://madpenguin.org/cms/?m=show&id=4775
Digg - IBM: PHP development within Eclipse http://digg.com/programming/IBM_PHP_development_within_Eclipse
Digg - Will IBM Buy Zend / PHP ? http://digg.com/linux_unix/Will_IBM_Buy_Zend_PHP
php, simplexml | Diigo http://www.diigo.com/tag/php+simplexml
ibm, rest | Diigo http://www.diigo.com/tag/ibm,rest
IBM poop heads say LAMP users need to "grow up" http://naeblis.cx/rtomayko/2005/05/28/ibm-poop-heads
Hypergene MediaBlog " IBM, blogging and the rise of the world's biggest ... http://www.hypergene.net/blog/print.ph
Other Yahoo search methods

- $yahoo->imageSearch
- $yahoo->videoSearch
- $yahoo->localSearch
- $yahoo->newsSearch
News Feeds
Importing news feeds

• **Usage**
  
  ```php
  $feed = Zend_Feed::import($url);
  ```
  
  ▪ Returns an instance of Zend_Feed_Abstract
    • Implements the Iterator interface

• **Understands**
  
  ▪ RSS 1.0
  ▪ RSS 2.0
  ▪ Atom
What’s New?
New in ZF 1.7x

• **Just a few of the enhancements:**
  - Performance enhancements in **Zend_Loader**, **Zend_Controller**, and server components
  - **Zend_Amf** component
  - Dojo Toolkit 1.2.1
  - **ZendX_JQuery** component
  - **Zend_Tool** in incubator
  - Google book search API in **Zend_Gdata**
  - **Zend_Db_Table_Select** support for **Zend_Paginator**
What’s Next?
Near future

- **From ZF wiki:**
  - framework.zend.com/wiki/display/ZFPROP/Home
  - In Standard Library Incubator

<table>
<thead>
<tr>
<th>From ZF wiki</th>
<th>From ZF wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zend_Action_Controller Directory Tree: Christopher Thompson</td>
<td>Zend_RememberTheMilk</td>
</tr>
<tr>
<td>Zend_Crypt: Pádraic Brady</td>
<td>Zend_Feed_Reader: Pádraic Brady &amp; Jurriën Stutterheim</td>
</tr>
<tr>
<td>Zend_Db Firebird-Interbase support</td>
<td>Zend_Image_Barcode: Mickael Perraud &amp; Julien Pauli</td>
</tr>
<tr>
<td>Zend_MailRead Proposal: Nico Edtinger</td>
<td>Twitter</td>
</tr>
<tr>
<td>Zend_Server Proposal: Davey Shafik</td>
<td>Zend_Log factory(): Martin Roest</td>
</tr>
<tr>
<td>Zend_Uri Improvements: Shahar Evron</td>
<td>Zend_LogWriterSyslog: Thomas Gelf</td>
</tr>
<tr>
<td>Zend.Validate_BarcodeSBN13: Andries Seutens</td>
<td>Zend_JsonExpr to allow Javascript Expressions (functions) to be encoded using Zend_Json</td>
</tr>
<tr>
<td>Zend_Mail_TransportQueue: Simon Mundy</td>
<td>Extended Zend_Ldap Proposal: Stefan Gehrig</td>
</tr>
<tr>
<td>Zend_Controller_Action_Helper_MultiPageForm: Jurriën Stutterheim</td>
<td>Zend_Service_Digg: Luke Crouch</td>
</tr>
<tr>
<td>Zend_Db_Table_Plugin: Simon Mundy, Jack Sleight</td>
<td>Zend_LogWriterMail</td>
</tr>
<tr>
<td>Zend_Queue: Justin Plock</td>
<td>Zend_TagCloud: Ben Scholzen</td>
</tr>
<tr>
<td>Zend_Framework Default Project Structure: Wil Sinclair</td>
<td>Zend_LoaderAutoloader: Ralph Schindler</td>
</tr>
<tr>
<td>Zend_FilterStripNewlines: Martin Hujer</td>
<td>Zend_Markup: Pieter Kokx</td>
</tr>
<tr>
<td>Zend_Ascii: Ben Scholzen</td>
<td>Zend.Validate_Db_RecordExists:</td>
</tr>
<tr>
<td>Zend_CryptRsa: Pádraic Brady</td>
<td>Zend.Validate_Db_NoRecordExists: Ryan Mauger</td>
</tr>
<tr>
<td>Zend_Yadis: Pádraic Brady</td>
<td>Zend_Controller_Router_Route_Rest: Luke Crouch</td>
</tr>
<tr>
<td>Zend_Oauth: Pádraic Brady</td>
<td>Zend_Loader_Autoloader_Resource: Matthew Weier O'Phinney</td>
</tr>
<tr>
<td>Zend_View_Helper_Cycle: Kamil Nowakowski</td>
<td>Zend_Gravatar Proposal: Wojciech Naruniec</td>
</tr>
</tbody>
</table>
Further out in the future

- Emphasis on tooling to create and deploy projects
- Look here: framework.zend.com/wiki/display/ZFPROP/Home
My goals

• **For my own apps**
  - Write more view helpers
  - Refactor pre-1.6 piecemeal Dojo code with ZF’s built-in Dojo view helpers and form elements
  - Replace homegrown database code with Zend_Db classes
    - Consistent quoting, escaping, prepared statements, profiler
    - And to provide feedback to developers
  - Use more of Zend_Form’s power with Zend_Filter and Zend_Verify.
    - My controller actions now do too much

• **ZF**
  - Contribute to Zend_Db_Adapter_Db2
In conclusion...
Zend Framework is…

A use-at-will component framework that is:

• built with OO, PHP 5
• intended to set a high standard for enterprise PHP
• flexible with MVC to put you on right track
• full of components that handle drudgery for you, and web services, too
• always improving. You can contribute
Resources: books

Books in Japanese, German, Portuguese:
Resources: online

• **On the web:**
  - framework.zend.com/docs/quickstart
  - survivethedeeper.com/pdf/survivethedeeper.pdf
  - zfforums.com

• **Send me your thoughts:**
  - alan@alanseiden.com
  - http://alanseiden.com (my blog)
Questions?

alan@alanseiden.com

See you at TGIF!