Zend Framework: Creating a CRUD Application

Matthew Weier O'Phinney
PHP Developer
matthew@zend.com

Zend The php Company

Who am I?

Zend
The *php* Company

Matthew Weier O'Phinney

- PHP Developer, Zend Technologies
  - Production site maintenance and deployment
  - Internal web services
- Open Source Contributor
  - PEAR
  - Cgiapp
  - Solar
- Zend Framework Core Contributor
  - Too many to list: MVC, Mail/MIME, JSON, XmlRpc, Rest...
- Phly PEAR Channel maintainer
  - Phly_Auth, Phly_InputFilter, and other NIH projects

Framework Overview

Zend Framework provides a high-quality open-source framework for developing Web Applications and Web Services.

By following the PHP spirit, the Zend Framework delivers easy-to-use and powerful functionality, focusing on the challenges of building robust, secure and modern Web applications.

http://framework.zend.com/

Framework Overview

“Things should be made as simple as possible, but no simpler.”
-- Albert Einstein

- Simpler is easier to use.
- Simpler is more stable, and less prone to error.
- Simpler is more compatible.
- Simpler is easier to maintain.

Framework Overview

Framework Vision: Extreme Simplicity:

- Simple, yet powerful
- Use what you need approach
- Focused on the task
- Highly productive
- Cost effective
- Scales from simple tasks to customized applications

Framework Overview

Zend The php Company

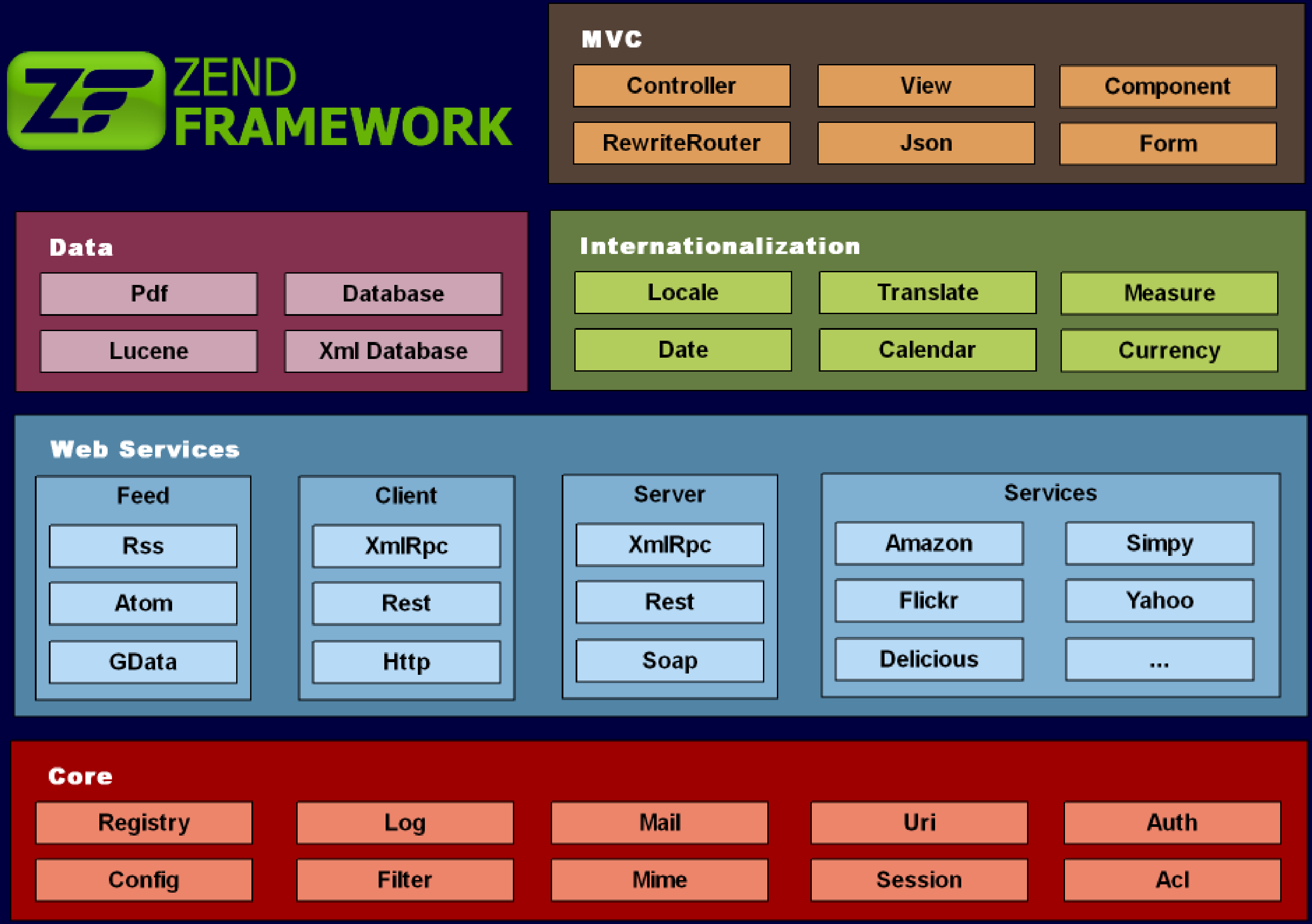
Framework Principles

- Keep it "extremely simple" - stick to 20%/80% rule and compensate by:
  - Extensibility
  - Use-at-will architecture
  - Configuration-less
- Cherry pick best-of-breed ideas
- Showcase current trends in Web development (Web Services, Ajax, Search, etc.)
- Document development with use-cases
- Only high quality and necessary components
- Friendly license (BSD license)
  - Contributors agree to contributor's license agreement

Framework Architecture

- It’s a framework, not just components
  - “Glue” to build applications
  - Tooling to increase productivity
  - Components developed and tested together
- Use only some components if you’d like, but…
  - It will always be distributed in its entirety
  - Upgrading or bundling will always be one-click
- Still plays well with others (PEAR, Horde, Solar, etc.)

Zend Framework Architecture



Development Process

- PHP 5.1.4 and later will be supported
- All errors throw exceptions (almost)
- Constants on the class-level
- Uses Zend Engine II & SPL interfaces where practical
- Use __magic() only where it makes sense

Development Process

QA Process

- Strict adherence to Zend Coding Standards
- All classes fully unit tested with PHPUnit
- Peer-review and approval of all code

MVC in Zend Framework

MVC in Zend Framework

Zend
The *php* Company

Basics

- Basic URL routes:
  - /controller/action
  - /controller/action/param/value
  - /module/controller/action
  - /module/controller/action/param/value
- controller == Zend_Controller_Action-derived class ending with 'Controller'
  - IndexController, BlogController
- action == public method ending with 'Action'
  - indexAction, listAction

MVC in Zend Framework

Workflow

- Bootstrap file
  - Instantiate Zend_Controller_Front
  - Setup front controller environment
- $front->dispatch()
  - get request (Zend_Controller_Request_Abstract)
  - route request (Zend_Controller_Router_Rewrite)
  - start dispatch loop
    - dispatch action (Zend_Controller_Dispatcher_Standard)
      - instantiate controller (Zend_Controller_Action)
      - call action method
  - send response (Zend_Controller_Response_Abstract)

MVC in Zend Framework

Plugins

- Major point of extensibility
  - Eliminates need to extend MVC classes
  - Modify actions for all controllers in site, including third-party apps
- Hooks bookend each major action in the dispatch process
- Register any number of plugins with the front controller
  - Plugins are called in the order in which they are registered
  - Each plugin can implement one or many hooks

MVC in Zend Framework

Plugins (cont.)

Defined hooks:

- routeStartup() - prior to routing
- routeShutdown() - after routing has finished
- dispatchLoopStartup() - prior to starting the dispatch loop
  - predispatch() - prior to dispatching an individual action
  - postDispatch() - after dispatching an action
- dispatchLoopShutdown() - after all actions have been dispatched

MVC in Zend Framework

Zend_View

- The 'View' in MVC
- Uses PHP as the template language
  - Set variables in a Zend_View object
  - Access them in a view script using object notation: $this->value
- Benefits: all of PHP is at your disposal
- Issues: all of PHP is at your *designer's* disposal

MVC in Zend Framework

Zend_View (cont.)

View Scripts

- Actual HTML (or other format) to return in response
- Mix in PHP to grab template variables and perform display logic
- Use PHP short tags for shorthand notation

MVC in Zend Framework

Zend_View (cont.)

Helpers

- Register helper paths with Zend_View object
- Helpers are helper classes with a single method; Zend_View then proxies via __call():

```
<dl>
    <dt>Username:</dt>
    <dd><?= $this->formText('username', null, array('width' => 20, 'maxlength' => 64)) ?></dd>
    <dt>Password:</dt>
    <dd><?= $this->formPassword('password', null, array('width' => 20, 'maxlength' => 256)) ?></dd>
</dl>
<?= $this->formSubmit("login", "Login") ?>
```

MVC in Zend Framework

Zend_View (cont.)

Filters

- Allow filtering compiled content prior to returning
- Like helpers, one class and method per filter
- Possible use cases:
  - HTML -> PDF
  - HTML -> JSON
  - HTML -> XML
  - Tidy
  - Injecting session IDs

Zend_Db

Zend_Db

Overview

- Provides abstraction to all databases PHP supports via Zend_Db_Adapter
  - Including those not supported by PDO
- Limited (select) query abstraction
- Provides optional query profiling via Zend_Db_Profiler
- Primitive ORM/Table Data Gateway via Zend_Db_Table

Zend_Db

Zend
The *php* Company

Benefits

- Code portability
  Ephemeral; typically you'll end up writing DB specific code to utilize special features or for optimization reasons
- Easily set default database adapter for all tables, and override on a per-table basis
- Nice OO syntax:

```
public function getSnippets($category)
{
    $select = $this->select()
        ->from($this->_name, array(
            'id',
            'title',
            'description',
            'developer as author',
            'entered as creationDate',
            'last_modified as lastModifiedDate',
            'code',
            'rating'
        ))
        ->where('category = ?', $category)
        ->where('status = ?', 'A')
        ->order('title', ASC);

    return $this->getAdapter()->fetchAll($select);
}
```

Zend_Db

Zend
The *php* Company

Problems

- PDO drivers are stable, but many others are not (yet)
- Zend_Db_Table by default returns Zend_Db_Row and Zend_Db_Rowset objects; not optimal for many use cases
- Zend_Db_Table is simply a Table Data Gateway - simpler ActiveRecord and more complex ORM not available
- Zend_Db_Select is convenient, but no analogues for update, insert, or delete actions
- Zend_Db_Table 'inflection' can be painful

Zend_Db

- Note: under active development; most issues illustrated should be gone by 0.9

Putting Together a CRUD Application

Putting it Together

Create a Generic Table Object

- Extend Zend_Db_Table
- Add accessors to allow grabbing Zend_Db_Table instances for each table and getting metainfo

```
class Phly_Db_Table_Generic extends Zend_Db_Table
{
    protected $_name = 'test';
    protected $_fieldInfo;

    protected static $_instances = array();
    public static function getInstance($table);
    protected function _setTable($table);
    public function getName();
    public function getPrimary();
    public function getFields();
    public function getFieldInfo();
}
```

Putting it Together

Zend
The *php* Company

Define the routing schema

- List rows in a table (paginated):
  - /crud/list/table/<TABLE>/page/<PAGE>
- Create a record in a table:
  - /crud/create/table/<TABLE>
- Read (view) a single record based on ID:
  - /crud/read/table/<TABLE>/id/<ID>
- Update an existing record:
  - /crud/update/table/<TABLE>/id/<ID>
- Delete a single record based on ID:
  - /crud/delete/table/<TABLE>/id/<ID>

Putting it Together

Bootstrap file

```
<?php
set_include_path('.:/path/to/lib/framework/library:/path/to/lib/pear');
require_once 'Zend/Db.php';
require_once 'Zend/Db/Table.php';
require_once 'Zend/Controller/Front.php';

$db = Zend_Db::factory('Pdo_Mysql', array(
    'host'     => 'localhost',
    'username' => 'matthew',
    'password' => 'password',
    'dbname'   => 'bostonphp',
    'profiler' => true
));
Zend_Db_Table::setDefaultAdapter($db);

$front = Zend_Controller_Front::getInstance();
$front->setControllerDirectory('../controllers')
      ->throwExceptions(true);

$front->dispatch();
```

Putting it Together

Create a controller class

- CrudController, in CrudController.php:

```
<?php
require_once 'Phly/Db/Table/Generic.php';

require_once 'Zend/Controller/Action.php';

class CrudController extends Zend_Controller_Action
{
    protected $_table;
    public $view;

    public function init();
    public function preDispatch();
    public function postDispatch();
    public function indexAction();
    public function getTableList();
    public function listAction();
    public function createAction();
    public function readAction();
    public function updateAction();
    public function deleteAction();
}
```

Putting it Together

Zend
The *php* Company

CrudController (cont.)

- init(): setup view
- preDispatch(): verify table is valid

```
public function init()
{
    $this->view = new Zend_View();
    $this->view->setHelperPath(dirname(__FILE__) . '/../helpers');
    $this->view->setScriptPath(dirname(__FILE__) . '/../views');
}

public function preDispatch()
{
    if ('index' != $this->_request->getActionName()) {
        $table = $this->_getParam('table', false);
        if (!$table) {
            return $this->_redirect('/crud', array('exit' => true));
        }

        try {
            $this->_table = Phly_Db_Table_Generic::getInstance($table);
        } catch (Exception $e) {
            return $this->_redirect('/crud', array('exit' => true));
        }

        $this->_tableName = $table;
    }

    $this->view->table = $table;
}
```

Putting it Together

Zend The *php* Company

CrudController (cont.)

- postDispatch(): push content into a sitewide template
- render(): make rendering output easier

```
public function postDispatch()
{
    $this->view->content = $this->_response->getBody();
    $this->render('site');
}

public function render($action = null)
{
    if (null === $action) {
        $action = $this->_request->getActionName();
    }
    $script = $action . '.phtml';

    $this->_response->appendBody($this->view->render($script));
}
```

Putting it Together

CrudController (cont.)

- indexAction(): display a list of tables

```
public function indexAction()
{
    $this->getTableList();
    $this->render();
}

public function getTableList()
{
    $result = $this->_table->getAdapter()->getConnection()->query(
        'SHOW TABLES'
    );
    $this->view->tables = $result->fetchAll();
}
```

Putting it Together

CrudController (cont.)

- listAction(): paginated list of table rows

```
public function listAction()
{
    $offset = $this->_paginate();

    $this->view->fields  = $this->_table->getFields();
    $this->view->primary = $this->_table->getPrimary();
    $this->view->rows    = $this->_table->fetchAll(null, null, 30, $offset);

    $this->render();
}
```

Putting it Together

CrudController (cont.)

- _paginate(): get number of pages and current offset

```
protected function _paginate()
{
    $total    = $this->_table->getAdapter()
              ->fetchOne('SELECT COUNT(*) FROM ' . $this->_table->getName());
    $offset   = 0;
    $numPages = ceil($total / 30);
    $curPage  = $this->_getParam('page', 1);

    $pages = array();
    for ($i = 1; $i <= $numPages; $i++) {
        $pages[$i] = ($i - 1) * 30;
    }

    if ((1 <= $curPage) && ($curPage <= $numPages)) {
        $offset = $pages[$curPage];
    } else {
        $curPage = 1;
    }

    $this->view->assign(array(
        'count'   => $total,
        'pages'   => $pages,
        'curPage' => $curPage
    ));

    return $offset;
}
```

Putting it Together

CrudController (cont.)

- createAction(): display new record form, and process new record submission

```
public function createAction()
{
    $this->view->values  = array();

    if ('post' == strtolower($_SERVER['REQUEST_METHOD'])) {
        $this->process();
    }

    $this->view->fields  = $this->_table->getFields();
    $this->render('form');
}
```

Putting it Together

CrudController (cont.)

- process(): process a new or updated record submission

```
public function process($update = false)
{
    $primary = $this->_table->getPrimary();
    $data = array();
    foreach ($this->_table->getFields() as $field) {
        if (!$update && ($field == $primary)) {
            continue;
        }
        if (isset($_POST[$field])) {
            $data[$field] = $_POST[$field];
        }
    }

    if ($update) {
        $id    = $data[$primary];
        $where = $this->_table->getAdapter()->quoteInto(
            $primary . ' = ?', $id
        );
        unset($data[$primary]);
        try {
            $this->_table->update($data, $where);
        } catch (Exception $e) {
            $this->view->error  = 'Unable to update record: '
                . $e->getMessage();
            $this->view->values = $data;
            return;
        }
    } else {
        try {
            $id = $this->_table->insert($data);
        } catch (Exception $e) {
            $this->view->error  = 'Unable to insert record: '
                . $e->getMessage();
            $this->view->values = $data;
            return;
        }
    }

    $this->_redirect(
        '/crud/view/table/' . $this->_tableName . '/id/' . $id,
        array('exit' => true)
    );
}
```

Putting it Together

Zend
The *php* Company

CrudController (cont.)

- readAction(): get a single row and display it

```
public function readAction()
{
    if (!$id = $this->_getParam('id', false)) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    if (!$this->view->row = $this->getRow($id)) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    $this->render();
}
```

Putting it Together

CrudController (cont.)

- getRow(): get a single row, and prepare it as an array

```
public function getRow($id)
{
    $row    = $this->_table->find($id);
    $fields = array_flip($this->_table->getFieldInfo());
    $data   = array();

    foreach ($fields as $inflector => $field) {
        if (!empty($row->$inflector)) {
            $data[$field] = $row->$inflector;
        }
    }

    if (empty($data)) {
        return false;
    }

    $this->view->id = $id
    return $data;
}
```

Putting it Together

Zend
The *php* Company

CrudController (cont.)

- updateAction(): display an edit form and process it

```
public function updateAction()
{
    if (!$id = $this->_getParam('id', false)) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    if (!$this->view->values = $this->getRow($id)) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    if ('post' == strtolower($_SERVER['REQUEST_METHOD'])) {
        $this->process();
    }

    $this->view->edit    = true;
    $this->view->fields  = $this->_table->getFields();
    $this->view->primary = $this->_table->getPrimary();
    $this->render('form');
}
```

Putting it Together

Zend
The *php* Company

CrudController (cont.)

- deleteAction(): delete a single record

```
public function deleteAction()
{
    if (!$id = $this->_getParam('id', false)) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    if (!$this->view->values = $this->getRow($id)) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    if ('post' != strtolower($_SERVER['REQUEST_METHOD'])) {
        return $this->_redirect('/crud/list/table/' . $this->_tableName);
    }

    $where = $this->_table->getAdapter()->quoteInto(
        'id = ?', $id
    );
    $this->_table->delete($where);

    return $this->render();
}
```

Putting it Together

Views

- index.phtml: list tables

```
<h2>Choose a table</h2>
<form id="tables" action="/crud/list" method="get"
    onSubmit="javascript:location.href=this.table.value">
    <?= $this->tableSelect($this->tables, $this) ?>
    <?= $this->formSubmit('go', 'Go') ?>
</form>
```

Putting it Together

Views (cont.)

- tableSelect helper: create a form select with URL paths for keys, and table names for values

```
class Zend_View_Helper_TableSelect
{
    public function tableSelect(array $tables, Zend_View_Abstract $view)
    {
        $select = array();
        foreach ($tables as $table) {
            $url = '/crud/list/table/' . $table;
            $select[$url] = $table;
        }

        return $view->formSelect('table', null, null, $select);
    }
}
```

Putting it Together

Views (cont.)

- list.phtml: list rows, paginated

```
<h2>Table: <?= $this->table ?></h2>
<?= $this->paginate($this) ?>
<table>
    <tr>
    <? foreach ($this->fields as $field): ?>
        <th><?= $this->escape($field) ?></th>
    <? endforeach ?>
    </tr>
    <? foreach ($this->rows as $row): ?>
    <tr>
        <? foreach ($row as $field => $value): ?>
        <td>
            <? if ($field == $this->primary): ?>
            <a href="/crud/view/table/<?= $this->table ?>/id/<?= $va
                <?= $value ?>
            </a>
            <? else: ?>
            <?= $value ?>
            <? endif ?>
        </td>
        <? endforeach ?>
    </tr>
    <? endforeach ?>
</table>
<?= $this->paginate($this) ?>
```

Putting it Together

Views (cont.)

- paginate helper: create pager for list view

```
class Zend_View_Helper_Paginate
{
    public function paginate(array $tables, Zend_View_Abstract $view)
    {
        $pages = $view->pages;
        $count = count($pages);

        if (1 == $count) {
            return;
        }

        $baseUrl   = '/crud/list/' . $view->table . '/page/';
        $curPage   = $view->curPage;
        $startPage = 1;
        $endPage   = $count;

        if ($count > 9) {
            $startPage = ($curPage < 6) ? 1 : $curPage - 4;
            $endPage   = ($curPage + 4 > $count) ? $count : $curPage + 4;
            if (($startPage < 5) && ($endPage < $count)) {
                $endPage = $startPage + 8;
            } elseif (($endPage == $count) && ($endPage - $startPage < 9)) {
                $startPage = $endPage -8;
            }
        }

        return $this->render($curPage, $startPage, $endPage, $baseUrl);
    }
```

Putting it Together

Views (cont.)

- paginate helper: render() view

```
public function render($curPage, $startPage, $endPage, $baseUrl)
{
    $pager = '<ul class="pager">';
    if (1 < $curPage) {
        $pager .= '<li><a href="' . $baseUrl . '1" title="first page">&lt;&lt;</a></li>';
    }

    for ($i = $startPage; $i <= $endPage; $i++) {
        $page = $i;
        if ($curPage == $page) {
            $pager .= '<li class="current">' . $page . '</li>';
        } else {
            $pager .= '<li><a href="' . $baseUrl . $page
                   . '" title="page ' . $page . '">'
                   .  $page . '</a></li>';
        }
    }

    if ($curPage < $count) {
        $pager .= '<li><a href="' . $baseUrl . ($curPage + 1)
               . '" title="next page">next</a></li>';
        $pager .= '<li><a href="' . $baseUrl . $count
               . '" title="last page">&gt;&gt;</a></li>';
    }

    $pager .= '</ul>';

    return $pager;
}
```

Putting it Together

Zend
The *php* Company

Views (cont.)

- read.phtml: display a single record, with links to edit or return to table view

```
<h2>View Record</h2>
<h3>Viewing record <b><?= $this->escape($this->id) ?></b></h3>
<dl>
<? foreach ($this->row as $field => $value): ?>
    <dt><?= $this->escape($field) ?></dt>
    <dt><?= $this->escape($value) ?></dt>
<? endforeach ?>
</dl>
<p>
    <a href="/crud/update/table/<?= $this->table ?>/id/<? $this->id ?>">
        Edit this record
    </a> |
    <a href="/crud/list/table/<?= $this->table ?>">
        Return to table view
    </a>
</p>
```

Putting it Together

Views (cont.)

- form.phtml: display a new or edit form

```
<?
$update = false;
$action = 'create';
if (isset($this->edit)) {
    $update = true;
    $action = 'update';
}
?>
<h2><?= ($update) ? 'Edit' : 'New' ?> Record</h2>
<form name="record" action="/crud/<?= $action ?>/table/<?= $this->table ?><?
    ($update) ? '/id/' . $this->id : '' ?>" method="post">
<fieldset>
    <legend>Details</legend>
    <? if ($update): ?><?= $this->formHidden('id', $this->id) ?><? endif ?>
    <dl>
    <? foreach ($this->fields as $field): ?>
        <dt><?= $this->escape($field) ?></dt>
        <dd><?
            $value = '';
            if (isset($this->values[$field])) {
                $vaule = $this->values[$field];
            }
            echo $this->formText($field, $value);
        ?></dd>
    <? endforeach ?>
    </dl>
    <?= $this->formSubmit('process', $action) ?>
</fieldset>
</form>
```

Putting it Together

Zend The php Company

Views (cont.)

- delete.phtml: display a deletion confirmation message

```
<h2>Record deleted</h2>
<p>
    Record <b><?= $this->id ?></b> was successfully deleted from table
    <b><?= $this->table ?></b>.
</p>
<p>
    <a href="/crud/list/table/<?= $this->table ?>">
        Return to table list
    </a>
</p>
```

Putting it Together

Zend
The *php* Company

Views (cont.)

- site.phtml: site template

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title></title>
</head>
<body>
    <h1>CRUD Application</h1>
    <?= $this->content ?>
    <p><a href="/crud">List tables</a></p>
</body>
</html>
```

Other Considerations

Other Considerations

- Authorization and ACLs
  - Zend_Auth, Zend_Acl
- Input Filtering
  - Zend_Validate, Zend_Filter
- Exposing as web services
  - XML-RPC => Zend_XmlRpc_Server
  - REST => Zend_Rest_Server
- Exposing to AJAX
  - Zend_Json
  - SimpleXML

Questions?