Zend Forms

Eric Lightbody

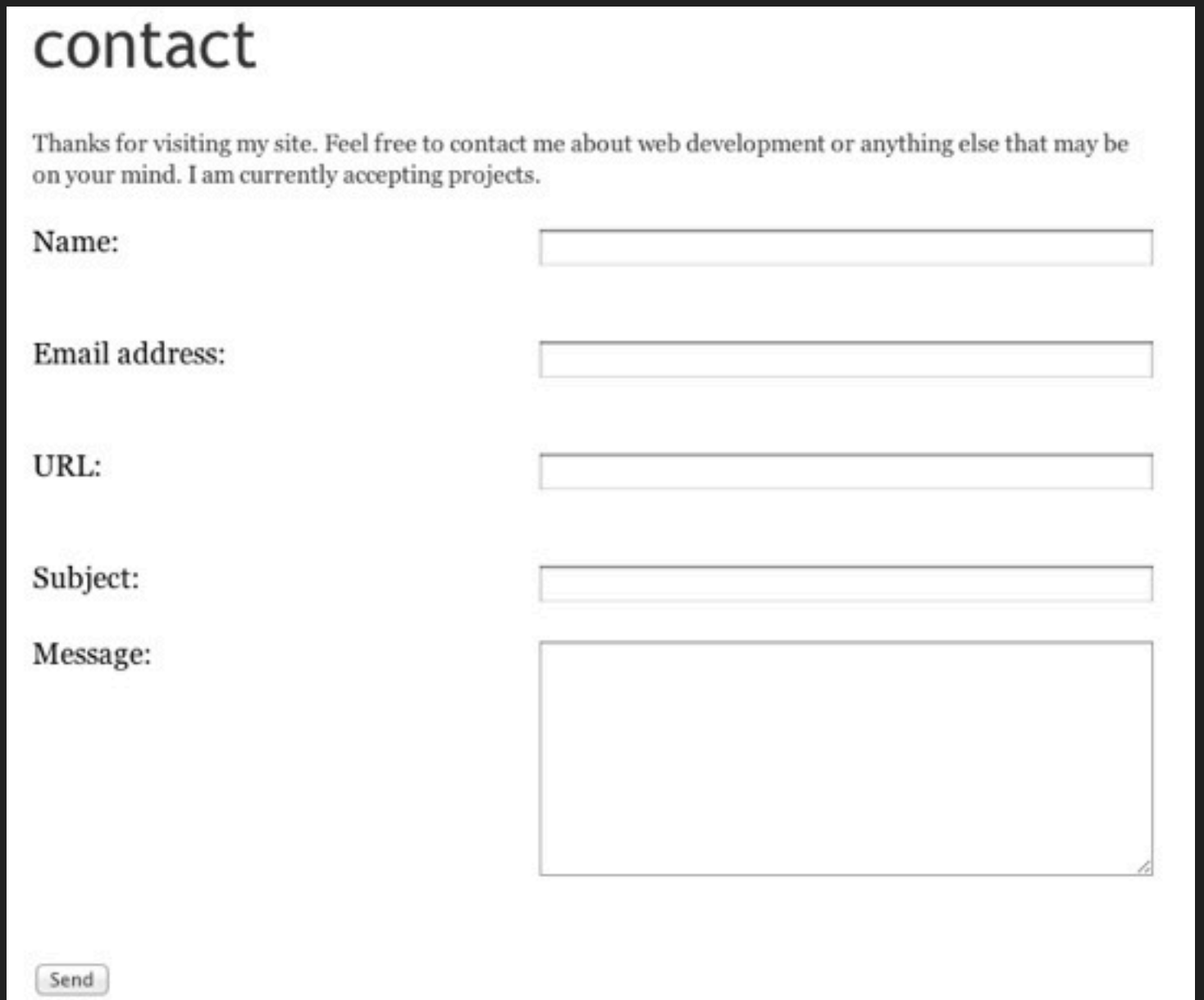
/about .me

- Web developer for almost ten years
- Computer Science from UW-Green Bay
- Nothing else

What are they?

- Displaying (rendering)
- Filtering and validating
- Grouping (fieldsets, etc.)

Old Way


contact
Thanks for visiting my site. Feel free to contact me about web development or anything else that may be on your mind. I am currently accepting projects.
Name:
Email address:
URL:
Subject:
Message:
Send

Oh Yeah

```
2  class Application_Form_Contact extends Zend_Form
3  {
4      /**
5       * initialize the form
6       */
7      public function init()
8      {
9          $this->setMethod('post');
10         $this->setAttrib('id', 'contactForm');
11
12
13         $this->addElement('text', 'name', array(
14             'label'          => 'Name:',
15             'required'       =>  true,
16             'validators'     =>  array(
17                 array('StringLength', false, array(1,250)),
18             ),
19             'class'          =>  'required'
20         ));
21
22         $this->addElement('text', 'email', array(
23             'label'          => 'Email:',
24             'required'       =>  true,
25             'validators'     =>  array(
26                 array('StringLength', false, array(1,250)),
27                 'EmailAddress'
28             ),
29             'class'          =>  'required email'
30         ));
31
32         more awesomeness follows...trust me
```

Don’t be hatin’

Model / View / Controller?

MVC what??

Model

- Color
- Temperament
- Size

- Color – brunette
- Temperament – hyper weird
- Size – perfect

- Color - blonde
- Temperament - basically a jerk
- Size - stout

A bit more on models

- Don’t care where the data is going
- Don’t care where it’s been
- Model != Database

Controller

- Traffic director
- Simplest sense takes data and passes it to the view

Our View

- What is presented to the user

Where do forms fit in?

- First, what does it do?
- Uses decorators and view helpers
- Filters and validates

So is this a form a model?

- I say no
- This Matthew jerk says pretty much
  - http://mwop.net/blog/200-Using-Zend_Form-in-Your-Models.html

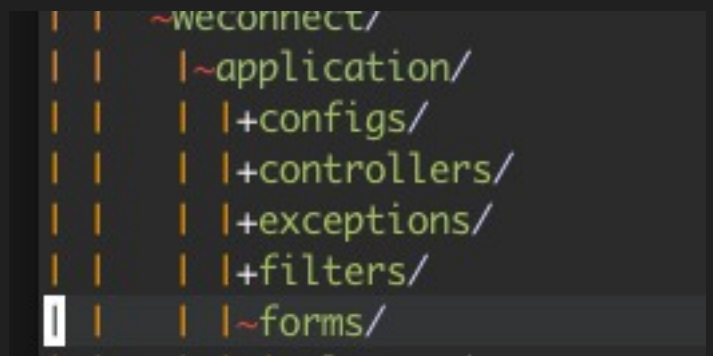
Gotta be a view then

NOPE

Let me guess it's not a controller

CORRECT!!

Where do they go?

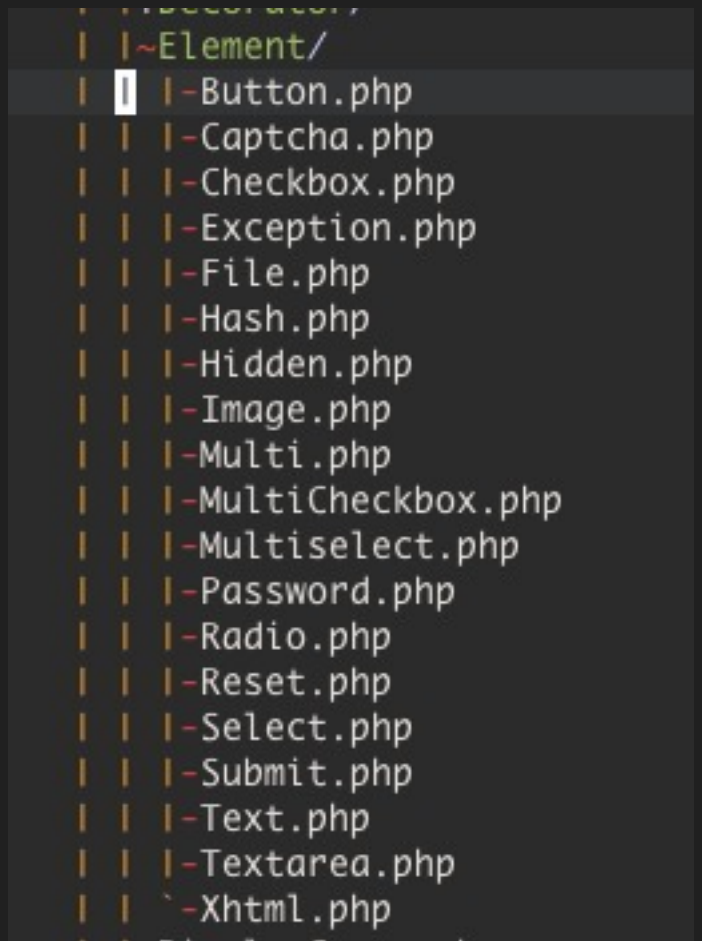


First, Creating the Form

```php
<?php
/**
 * Blog Entry form
 */
class Application_Form_Blog extends Zend_Form
{
    public function init()
    {
        $this->setMethod('post');
        $this->setAttrib('id', 'blogForm');

        $this->addElement('text', 'title', array(
            'label'         => 'Blog Title',
            'required'      =>  true,
            'class'         =>  'required'
        ));

        $this->addElement('textarea', 'body', array(
            'label'         => 'Body Content',
            'required'      =>  true,
            'class'         =>  'required',
            'maxlength'     =>  65000,
        ));

```

Form Elements

- Select
- Text
- Textarea
- Hidden
- File
- Submit
- Hash



Filters and Validators

- Validators used for making sure content coming in is safe
  - If it doesn't meet our requirements, user has to change the data
- Filters used for altering the data provided
  - Trimming strings, etc.

Filters and Validators

- Check out existing filters and validators
  - Filters in /Filter
  - Validators in /Validate

Useful Validators

- StringLength
- EmailAddress
- File Validators
  - Count
  - Extension
  - IsImage

Useful Filters

- Compress
- StringTrim
- File Filters
- Check them out!

Using filters & validators

```
...

class Application_Form_Blog extends Zend_Form
{
    /**
     * initialize the form
     */
    public function init()
    {
        $this->addElementPrefixPath('Application_Validate', APPLICATION_PATH . '/validators', 'validate');
        $this->addElementPrefixPath('Application_Filter', APPLICATION_PATH . '/filters', 'filter');

        $this->setMethod('post');
        $this->setAttrib('id', 'blogForm');

        $this->addElement('text', 'title', array(
            'label'         => 'Blog Title',
            'required'      =>  true,
            'validators'    =>  array(
                array('StringLength', false, array(1,250)),
            ),
            'class'         =>  'required fullcontent'
        ));

        $this->addElement('textarea', 'body', array(
            'label'         => 'Body Content',
            'required'      =>  true,
            'validators'    => array(
                array('StringLength', false, array(1,65000)),
            ),
            'class'         =>  'required mce fullcontent',
            'maxlength'     =>  65000,
            'filters'    =>array('StringTrim', 'CleanHTML'),
        ));
```

Custom Filters and Validators

- Very easy to implement custom filters and validators
- These can both be used in other parts of your application

Help the form out

```php
<?php
    /**
     * initialize the form
     */
    public function init()
    {
        $this->addElementPrefixPath('Application_Validate', APPLICATION_PATH . '/validators', 'validate');
        $this->addElementPrefixPath('Application_Filter', APPLICATION_PATH . '/filters', 'filter');
```

Validator Code

```php
<?php
/**
 * Determines if the the date provided is current
 */
class Application_Validate_CurrentDate extends Zend_Validate_Abstract
{
    /**
     * @var int error key
     */
    const PAST_DATE = 'pastDate';

    /**
     * @var array error templates
     */
    protected $_messageTemplates = array(
        self::PAST_DATE  => "The date cannot occur in the past",
    );

    /**
     * Main validator
     * Is date current?
     *
     * @return boolean
     */
    public function isValid($value, $context = null)
    {
        if (strtotime($value) < strtotime('midnight')) {
            $this->_error(self::PAST_DATE);
            return false;
        }
        return true;
    }
}
```

Filter Code

```php
<?php
/**
 * Makes sure that the URL Path begins properly
 *
 * Basically this just means it has a leading slash
 */
class Application_Filter_UrlPathLeadingSlash implements Zend_Filter_Interface
{
    /**
     * Filters the item
     * @param string $value
     * @return filtered element
     * @see Filter/Zend_Filter_Interface::filter()
     */
    public function filter($value)
    {
        if (empty($value)) return $value;

        if (stripos($value, '/') !== 0) {
            $value = '/' . $value;
        }

        return $value;
    }
}

```

Custom Form Elements

```php
<?php
    public function init()
    {
        $this->addPrefixPath('Application_Form', APPLICATION_PATH . '/forms/');

```

Custom Form Elements

```php
<?php
...
class Application_Form_Element_State extends Zend_Form_Element_Multi
{
    /**
     * Specify the view helper
     */
    public $helper = 'formSelect';

    /**
     * @var array the state list
     */
    protected $_stateList = array( 'AL'=>"Alabama",
        ...
        'WY'=>"Wyoming",
    );

    /**
     * @see Zend_Form_Element::init()
     */
    public function init()
    {
        $stateKeys = array_keys($this->_stateList);
        $stateList = array('' => '--Select--') + array_combine($stateKeys, $stateKeys);
        $this->addMultiOptions($stateList);
    }
}
```

Preventing CSRF Attacks

- Verify request came from an authorized user
- Most sites only verify the browser of an authorized user
- First site can send request to second site

Stick It To 'Em

- One approach is a unique cookie for that session
- Another is generate unique key for every request
- http://www.codinghorror.com/blog/2008/10/preventing-csrf-and-xsrf-attacks.html

How To Prevent

```
$this->addElement('hash', 'csrf', array(
    'ignore'=>true,
    'timeout'=>Zend_Registry::get('options')->hashtimeout
));
```

Displaying

Wait for it...

Print This Mother Out!!

Pass the form to your view from controller

```
$this->view->form = $form;
```

Display it

```
echo $this->form;
```

<whine>
my designer doesn't like dts and dds
</whine>

Decorators

- Zend Form Elements are made of one view helper and a bunch of decorators
- Knowing decorators allows you to control the html

Normal Output

```
<form id="contactForm" enctype="application/x-www-form-urlencoded" method="post" action="">
<dl class="zend_form">
    <dt id="emailto-label"><label for="emailto" class="optional">Send Message To:</label></dt>
    <dd id="emailto-element">
        <select name="emailto" id="emailto">
            <option value="0" label="-- General Contact --">-- General Contact --</option>
        </select>
    </dd>
    <dt id="name-label">
        <label for="name" class="required">Name:</label>
    </dt>
    <dd id="name-element">
        <input type="text" name="name" id="name" value="" class="required">
    </dd>
    <dt id="email-label">
        <label for="email" class="required">Email:</label>
    </dt>
    ...
</dl>
```

Change All Elements at Once

- Awesome article on all of this http://goo.gl/RiekA

```
2 $this->setElementDecorators(array(
3     'viewHelper',
4     'Errors',
5     array(array('data'=>'HtmlTag'),array('tag'=>'td')),
6     array('Label',array('tag'=>'td')),
7     array(array('row'=>'HtmlTag'),array('tag'=>'tr'))
8 ));
```

Render the Form In Your View

Ignore this slide

```
1  <?php
2  $form = $this->form;
3  // Remove <dt> from label generation
4  foreach ($form->getElements() as $element) {
5      $element->getDecorator('label')->setTag(null);
6  }
7  ?>
8  <form method=\"<?php echo $form->getMethod() ?>\" action=\"<?php echo
9      $form->getAction()?>\">
10     <div class=\"element\">
11         <?php echo $form->title->renderLabel() . $form->title->renderViewHelper() ?>
12         <?php echo $form->firstName->renderLabel() . $form->firstName->renderViewHelper() ?>
13         <?php echo $form->lastName->renderLabel() . $form->lastName->renderViewHelper() ?>
14     </div>
15     <div class=\"element\">
~
~
```

What About The Controller?

- Controllers makes decisions based on what happens with the form
- Does it validate or not?
- isValid()!!

Controller Code

```php
<?php
    $form = new Application_Form_Blog();

    $request = $this->getRequest();
    if ($request->isPost()) {
        $post = $request->getPost();
        $form->populate($post);
        if ($form->isValid($post)) {
            ...
            $form->getValues();
            ...
        }
    }
    else {
        $form->populateFromBlog();
    }

```

Populating From Your Model

```php
<?php
    /**
     * @param Application_Model_Blog $blog
     */
    public function setBlog(Application_Model_Blog $blog)
    {
        $this->_blog = $blog;
        return $this;
    }

    /**
     * Populate the form from the blog object
     */
    public function populateFromBlog()
    {
        /** user atrributes **/
        $populateArray = array(
                                'title'=>$this->_blog->getTitle(),
                                'body'=>$this->_blog->getBody()
        );

        $this->populate($populateArray);
        return $this;
    }

```

Just Using Zend_Form

- Possible to only use certain parts of Zend
- Great way to get started
- Could go this route for legacy code

First the “Controller”

```php
<?php
    // Load needed Zend classes
    require_once("Zend/Loader.php");
    Zend_Loader::loadClass('Zend_View');
    Zend_Loader::loadClass('RequestForm');

    // Create a view to render the form
    $view = new Zend_View();

    // Create an instance of the form
    $form = new RequestForm();

    if (!$_POST) {
        // Render the blank form
        echo $form->render($view);
    }
    else if (!$form->isValid($_POST)) {
        // Renders form with error messages
        echo $form->render($view);
    }
    else {
        // Passed validation. Do someting with the values entered.
        $values = $form->getValues();
        extract($values);
    }
```

Now The Form

```php
<?php

Zend_Loader::loadClass('Zend_Form');

class RequestForm extends Zend_Form {

    public function init(){

        $this->setMethod('post');
        $this->setAttrib('id', 'siteForm');

        $this->addElement('text', 'name', array(
            'label'          => 'Site Name',
            'required'       =>  true,
            'filters'    =>array('StringTrim'),
            'validators'     =>  array(
                array('StringLength', false, array(0,250)),
            ),
        ));

        $this->addElement('submit', 'submitbutton', array(
            'ignore'=>true,
            'label'=>('Update About Us'),
        ));

    }

}
```

Thank You