

Zend Filter: A Secure Man's Best Friend

by Aaron Saray

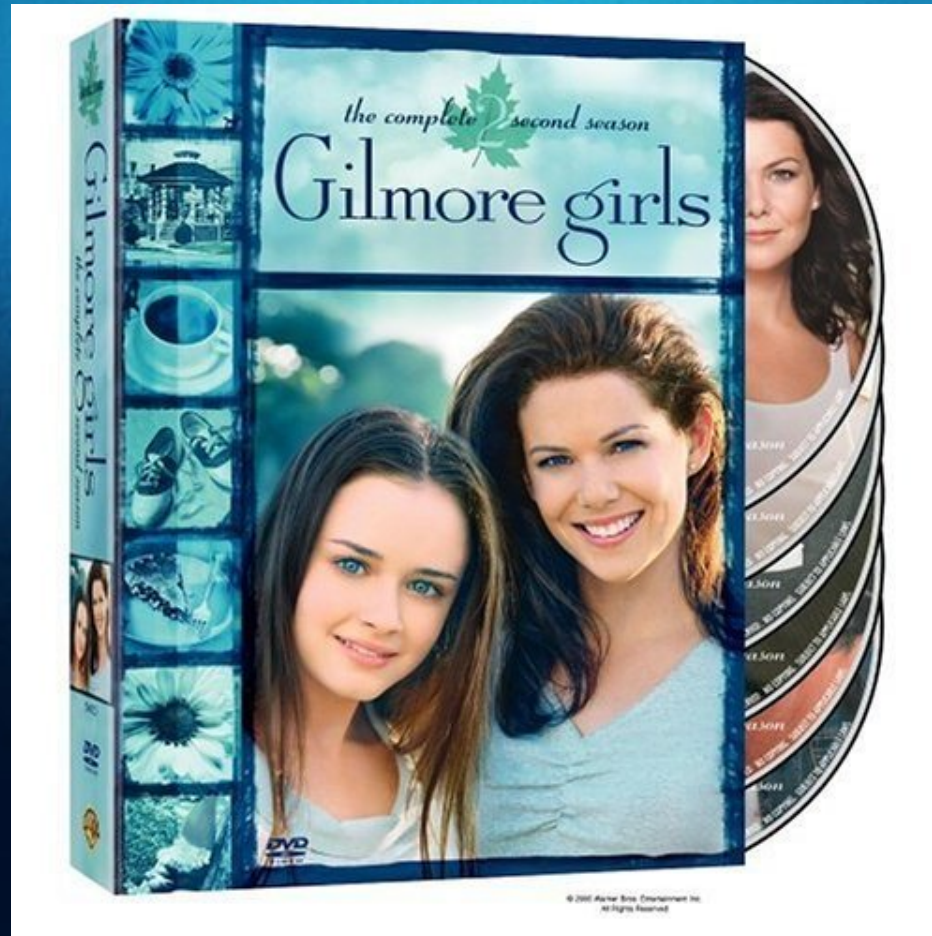
Why Trust This Guy?



- Programmer for nearly 2 decades (can you say Commodore 64?)
- Web Developer for 11 / PHP for a decade
- "Professional PHP Design Patterns"
 - Wrox Publishing
- Milwaukee PHP Users Group / MWDM
- Cuz I said so

What is a Secure Man?

A secure man can admit to owning this:



Real Security

Filtering data!!

- User Input
- System Output
- Server Requests

The Point: Filter out "Bad" things

But what is bad?

- Bad is complex
- Bad is unexpected
- Bad is malicious
- Bad is "the old version"
- Bad is placeholders
- Bad isn't always bad

Why Filter?

- Filter anything that would adversely effect
 - security
 - XSS
 - performance
 - business logic
 - dashes instead of dots
- Design Patterns:
 - filter is like decorate

What Does PHP Provide?

The Filter extensions (enabled by default since PHP 5.2.0)

Two filter types:

- Validate filters (referred to as Validators in ZF)
- Sanitize filters (removes unwanted data, Filters in ZF)

Some Useful/Notable Standard Filters

- FILTER_VALIDATE_EMAIL
 - duh
- FILTER_VALIDATE_IP
 - has flags for IPv4, IPv6, private/reserved ranges
- FILTER_VALIDATE_URL
 - according to RFC2396
- FILTER_SANITIZE_MAGIC_QUOTES
 - applies addslashes()

Example:

```
$userEmail = "someone@somewhere.com |oopsies";  
$filteredEmail = filter_var($userEmail, FILTER_SANITIZE_EMAIL);  
print $filteredEmail;
```

//output: someone@somewhere.com

Many more - check out <http://php.net/filter>

Getting into Zend Framework

- Using Zend Framework 1.11.9 for this presentation
- Get your own copy at <http://framework.zend.com/download/latest>
- Programmers are lazy - ZF from now on

What are Filters in relation to ZF?

- Zend Framework Filters are most akin to sanitize filters in PHP
- The core class is `Zend_Filter`
- Filter Classes implement `Zend_Filter_Interface`
 - primarily expose a `filter()` method
- Filters are built to be chained
- Three main ways to use ZF Filters...

Using ZF Filters: Zend_Form

```
class Application_Form_Signup extends Zend_Form
{
    public function init()
    {
        $this->addElement('text', 'email', array(
            'label'          =>'Email Address',
            'required'       =>true,
            'filters'        =>array('StringTrim', 'StringToLower'),
            'validators'=>array(
                'EmailAddress',
                array('StringLength', false, array(1, 250))
            ),
        ));

        $this->addElement('submit', 'submitbutton', array(
            'ignore'=>true,
            'label'=>'Sign Up Now',
        ));
    }
}
```

Using ZF Filters: Creating new instance

```
$filter = new Zend_Filter_Int();  
$value = "6 fish";  
echo $filter->filter($value);
```

```
//output: 6
```

Using ZF Filters: staticFilter() method

```
echo Zend_Filter::filterStatic('/var/www/ohhai.gif', 'BaseName');  
  
//output: ohhai.gif
```

What's in the box?

- ZF comes with a set of standard filters
- Check them out here: <http://framework.zend.com/manual/en/zend.filter.set.html>
- Or for the more curious, check out the classes:
 - library/Zend/Filter

Some notable filters included in ZF:

- Zend_Filter_Compress
 - compress/decompress
 - various algorithms
- Zend_Filter_Digits
 - Filters for Digits - unicode regular expression
- Zend_Filter_Encrypt
 - Super easy way to handle MCrypt
- Zend_Filter_LocalizedToNormalized
 - Think \$, date/time

even more notable filters included in ZF:

- `Zend_Filter_HtmlEntities`
 - converts your code into a pancake
- `Zend_Filter_StringToLower`
 - useful for cases like email addresses
- `Zend_Filter_StringTrim`
 - useful for forms to filter input fields of trailing spaces

Additional parts from Zend_Filter

Unfortunately, these are out of scope of this initial presentation:

- Chaining
 - Connecting many filters together
 - Changing order filters are applied
- Zend_Filter_Input
 - build complex set of filters/validators as parameters
 - apply all of them to input/array
- Zend_Filter_Inflexor
 - build filter workflow on patterned strings
 - patterns similar to ZF router route string format

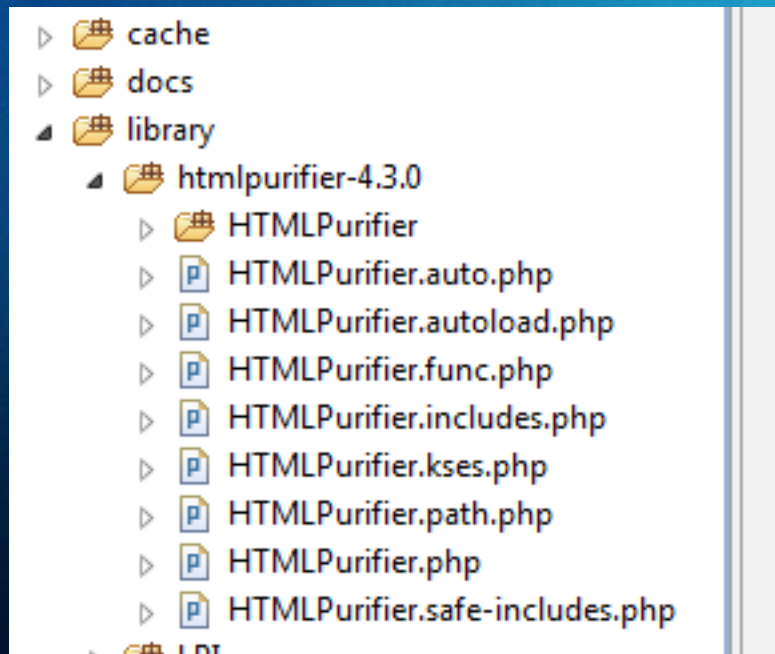
Real Life Time

- Problem: I need HTML input/output for my application. I'm using something snazzy like TinyMCE.
- Caveats:
 - Must allow some whitelisted html through
 - Must not allow for the hax0ring
- Initial findings:
 - HtmlEntities is somewhat helpful
 - pretty hammerish
 - StripTags is somewhat helpful
 - not precise enough

HTML Purifier

- I think HTML Purifier will solve my problem
 - <http://htmlpurifier.org>
- I want to use it in my ZF projects going forward
- I want it to be part of my "library" folder
- I want to do it the ZF way
 - Create an input filter with a `Zend_Filter_Interface` class

Add it to the Library



- Download it
- Add it to your library
 - I name mine by version

Basic Configuration

```
require_once 'htmlpurifier-4.3.0/HTMLPurifier.auto.php';
$config = HTMLPurifier_Config::createDefault();
$config->set('Cache.SerializerPath', '/path/to/cache');
$purifier = new HTMLPurifier($config);
$dirtyHTML = '<a href="/bar" onclick="alert('pwnd')">foo</a>';
echo $purifier->purify($dirtyHtml);
```

Output:

```
<a href="/bar">foo</a>
```

Create ZF Filter

- Create application/filters directory
- Register it in your bootstrap.php file with autoloader
 - So you can create new instances
 - Not necessary with forms
- Add it to your form definitions
 - So they can find it during filter stage
- Write filter

Add it to your Bootstrap Autoloader

```
protected function _initResources()  
{  
    $loader = $this->getResourceLoader();  
    $loader->addResourceType('filter', 'filters', 'Filter');  
}
```


Add it to your Form Class

```
<snippy>
public function init()
{
    $this->addElementPrefixPath(
        'Application_Filter',
        APPLICATION_PATH . '/filters/',
        'filter'
    );
</end snippy>
```

Write Filter Class

New file: application/filters/CleanHtml.php

```
class Application_Filter_CleanHTML implements Zend_Filter_Interface
{
    ....
}
```

```
/**
 * @var HTMLPurifier holds the instance of the purifier
 */
protected $_purifier = null;

/**
 * Filters the item
 * @param string $value
 * @return string filtered element
 */
public function filter($value)
{
    $this->_bootstrapPurifier();
    return $this->_purifier->purify($value);
}

/**
 * Start the purifier and store it locally
 */
protected function _bootstrapPurifier()
{
    ...
}
```

```
/**
 * Start the purifier and store it locally
 */
protected function _bootstrapPurifier()
{
    if ($this->_purifier == null) {
        require_once 'htmlpurifier-4.3.0/HTMLPurifier.auto.php';
        $config = HTMLPurifier_Config::createDefault();
        $config->set('Cache.SerializerPath', '/path/to/cache');
        $this->_purifier = new HTMLPurifier($config);
    }
}
```

All the code - for those with good eyes

```
class Application_Filter_CleanHTML implements Zend_Filter_Interface
{
    /**
     * @var HTMLPurifier holds the instance of the purifier
     */
    protected $_purifier = null;

    /**
     * Filters the item
     * @param string $value
     * @return string filtered element
     */
    public function filter($value)
    {
        $this->_bootstrapPurifier();

        return $this->_purifier->purify($value);
    }

    /**
     * Start the purifier and store it locally
     */
    protected function _bootstrapPurifier()
    {
        if ($this->_purifier == null) {
            require_once 'htmlpurifier-4.3.0/HTMLPurifier.auto.php';
            $config = HTMLPurifier_Config::createDefault();
            $config->set('Cache.SerializerPath', '/path/to/cache');
            $this->_purifier = new HTMLPurifier($config);
        }
    }
}
```

Blog Entry? This is how I roll...

```
class Application_Form_Blog extends Zend_Form
{

<snippy>

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

</snippy>
```

The End

Say Hi!

Aaron Saray

Milwaukee PHP Web Developer

<http://aaronsaray.com>

@aaronsaray

Download:

<http://aaronsaray.com/blog/2011/07/20/zend-filter-presentation>