Zend_Filter_Html - Thomas Weidner

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Filter_Html & Zend_Validate_Html Component Proposal

Proposed Component Name	Zend_Filter_Html & Zend_Validate_Html
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Filter_Html & Zend_Validate_Html
Proposers	Thomas Weidner
Zend Liaison	TBD
Revision	1.0 - 6 December 2009: Initial Draft. (wiki revision: 7)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Filter_Html is a component which filters a given input to be HTML conform.
Zend_Validate_Html is it's cousine which validates if a given input is HTML conform.

2. References

- Tidy
- HTMLPurifier

- See here for problems of actual ZF usage: Blog Post

3. Component Requirements, Constraints, and Acceptance Criteria

Zend_Filter_Html:

- This component **will** convert any input to conform HTML

- This component **will** prevent XSS attacks
- This component **will** produce 100% valid HTML output
- This component **will** use adapters to allow foreign libraries to be used

Zend_Validate_Html:

- This component **will** validate if input is 100% valid HTML
- This compontnt **will** use adapters to allow foreign libraries to be used

4. Dependencies on Other Framework Components

- Zend_Filter
- Zend_Validator

5. Theory of Operation

Actually Zend Framework does not have a component which really prevents XSS attacks.

Zend_Filter_Html filters given input, so it conforms the HTML standard. It prevents XSS attacks. Therefor it makes usage of Tidy to get a standard conform HTML output, and it uses HTMLPurifier to prevent any attacks.

Zend_Validate_Html validates is a given input conforms the HTML standard.

6. Milestones / Tasks

- Milestone 1: [DONE] Proposal finished
- Milestone 2: Proposal accepted
- Milestone 3: Working implementation
- Milestone 4: Unit tests
- Milestone 5: Documentation
- Milestone 6: Moved to core

7. Class Index

- Zend_Filter_Html
- Zend_Filter_Html_HtmlAbstract
- Zend_Filter_Html_Tidy
- Zend_Filter_Html_HtmlPurifier
- Zend_Validate_Html
- Zend_Validate_Html_HtmlAbstract
- Zend_Validate_Html_Tidy

8. Use Cases

UC-01

Filtering XSS attacks:

```
$input = '<a
href="data:text/html;base64,PHNjcmlwdD5hbGVydChkb2N1bWVudC5kb21haW4pPC9zY3JpcHQ%2B">Clic
Me And Win IPOD!!!!</a>';
$filter = new Zend_Filter_Html();
$filter->filter($input);

// This XSS attack would be prevented as the link is filtered
```

UC-02

Validating HTML:

```
$input = '<p>test</i>';
$validate = new Zend_Validate_Html();
$validate->isValid($input);

// Returns false as the input is not valid HTML
```

9. Class Skeletons

```
class Zend_Filter_Html {
    public function filter($value);
    // allows to use multiple adapters with the same filter in a defined order
}

class Zend_Filter_Html_HtmlAbstract {
    public function setOptions($options);
    public function getOptions();
}

class Zend_Filter_Html_Tidy {
    public function filter($value);
}

class Zend_Validate_Html {
    public function isValid($value);
    // allows to use multiple adapters with the same validator in a defined order
}

class Zend_Validate_Html_HtmlAbstract {
    public function setOptions($options);
    public function getOptions();
}

class Zend_Filter_Html_Tidy {
    public function isValid($value);
}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>