Zend_Filter_Hash - Justin Hendrickson

<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_Hash Component Proposal

Proposed Component Name	Zend_Filter_Hash
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Filter_Hash
Proposers	Justin Hendrickson
Zend Liaison	TBD
Revision	1.0 - 26 August 2009: Initial Draft. (wiki revision: 28)

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_Hash is an adapter driven filter class for hashing strings.

2. References

- http://www.php.net/crc32
- http://www.php.net/crypt
- http://www.php.net/hash
- http://www.php.net/md5
- http://www.php.net/sha1
- http://en.wikipedia.org/wiki/Cryptographic_hash_function

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide support for hashing strings.
- This component **will** provide multiple adapters.

4. Dependencies on Other Framework Components

- Zend_Filter_Interface
- Zend_Filter_Exception

5. Theory of Operation

Zend_Filter_Hash will give a Zend_Filter interface to various hash extensions. Additionally, the crypt adapter will provide facilities for salting strings and comparing crypt'ed strings.

6. Milestones / Tasks

- Milestone 1: Design notes will be published here. [DONE]
- Milestone 2: Pulished working prototype.
- Milestone 2: Working prototype checked into the incubator supporting use cases.
- Milestone 3: Unit tests exist, work, and are checked into SVN.
- Milestone 4: Initial documentation exists.

7. Class Index

- Zend_Filter_Hash
- Zend_Filter_Hash_Exception
- Zend_Filter_Hash_Adapter_Interface
- Zend_Filter_Hash_Adapter_Php
- Zend_Filter_Hash_Adapter_Hash
- Zend_Filter_Hash_Adapter_Crypt
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Interface
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Abstract
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Bcrypt
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_ExtendedDes
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Md5
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_StandardDes
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Sha256
- Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Sha512

8. Use Cases

UC-01

Crypt a password using the system default algorithm.

```
$password = "password";

$filter = new Zend_Filter_Hash(array(
    "adapter" => "crypt"
));

$ctext = $filter->filter($password);

$filter = new Zend_Filter_Hash(array(
    "adapter" => "crypt",
    "salt"    => $ctext
));

var_dump($filter->filter($password) == $ctext); // bool(true)
```

UC-02

Crypt a password using Bcrypt with a cost of 10 (direct loading of the salt generator).

```
$password = "password";

$salt = new Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Bcrypt(array(
    "cost" => 10
));

$filter = new Zend_Filter_Hash(array(
    "adapter" => "crypt",
    "salt"    => $salt
));

$ctext = $filter->filter($password);

$filter = new Zend_Filter_Hash(array(
    "adapter" => "crypt",
    "salt"    => $ctext
));

var_dump($filter->filter($password) == $ctext); // bool(true)
```

UC-03

Crypt a password using Bcrypt with a cost of 10 (lazy loading of the salt generator).

```
$password = "password";

$filter = new Zend_Filter_Hash(array(
    "adapter" => "crypt",
    "salt"    => array(
        "algorithm" => "bcrypt",
        "cost"      => 9
    )
));

$ctext = $filter->filter($password);

$filter = new Zend_Filter_Hash(array(
    "adapter" => "crypt",
    "salt" => $ctext
));

var_dump($filter->filter($password) == $ctext); // bool(true)
```

UC-04

Hash a string with the hash extensions sha512 algorithm.

```
$text = "password";

$filter = new Zend_Filter_Hash(array(
    "adapter" => "hash",
    "algorithm" => "sha512"
));

var_dump($filter->filter($text)); // string(128)
"b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d1
```

UC-05

Hash a string with the php md5 algorithm.

```
$text = "password";

$filter = new Zend_Filter_Hash(array(
    "adapter" => "php",
    "algorithm" => "md5"
));

var_dump($filter->filter($text)); // string(32) "5f4dcc3b5aa765d61d8327deb882cf99"
```

9. Class Skeletons

```


class Zend_Filter_Hash implements Zend_Filter_Interface
{

    public function __construct($options = null)
```

```
    {
    }

    public function __call($name, $arguements)
    {
    }

    public function filter($value)
    {
    }

    public function getAdapter()
    {
    }

    public function getPluginLoader()
    {
    }

    public function setAdapter($adapter)
    {
    }

    public function setConfig(Zend_Config $config)
    {
    }

    public function setOptions(array $options)
    {
    }

    public function setPluginLoader(Zend_Loader_PluginLoader $pluginLoader)
    {
    }

}

class Zend_Filter_Hash_Exception extends Zend_Filter_Exception
{
}

class Zend_Filter_Hash_Adapter_Interface implements Zend_Filter_Interface
{
}

class Zend_Filter_Hash_Adapter_Hash implements Zend_Filter_Hash_Adapter_Interface
{

    public function __construct($options = null)
    {
    }

    public function filter($value)
    {
    }

    public function getEnableRawOutput()
    {
    }

    public function getSupportedAlgorithms()
    {
    }

    public function setAlgorithm($algorithm)
    {
    }

    public function setEnableRawOutput($enableRawOutput)
    {
```

```
    }

    public function setConfig(Zend_Config $config)
    {
    }

    public function setOptions(array $options)
    {
    }

}

class Zend_Filter_Hash_Adapter_Php implements Zend_Filter_Hash_Adapter_Interface
{

    public function __construct($options = null)
    {
    }

    public function filter($value)
    {
    }

    public function getSupportedAlgorithms()
    {
    }

    public function setAlgorithm($algorithm)
    {
    }

    public function setConfig(Zend_Config $config)
    {
    }

    public function setOptions(array $options)
    {
    }

}

class Zend_Filter_Hash_Adapter_Crypt implements Zend_Filter_Hash_Adapter_Interface
{

    public function __construct($options = null)
    {
    }

    public function filter($key)
    {
    }

    public function getPluginLoader()
    {
    }

    public function getSalt()
    {
    }

    public function setConfig(Zend_Config $config)
    {
    }

    public function setOptions(array $options)
    {
    }

    public function setPluginLoader(Zend_Loader_PluginLoader $loader)
    {
```

```
    }

    public function setSalt($salt)
    {
    }

}

interface Zend_Filter_Hash_Adapter_Crypt_SaltGenerator_Interface
{

    public function __construct($options = null)
    {
    }

    public function buildSalt()
    {
```

```
        }

    }
```

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