Zend_Filter_Bool - Eric Coker

<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_Bool Component Proposal

Proposed Component Name	Zend_Filter_Bool
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Filter_Bool
Proposers	Eric Coker
Zend Liaison	TBD
Revision	1.0 - 28 August 2008: Initial Draft. (wiki revision: 3)

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_Bool converts common representations of boolean values in multiple types and string formats to a strongly-typed boolean value.

2. References

- PHP:Booleans
- Zend_Filter

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** convert input values into formatted and strongly-typed output.

4. Dependencies on Other Framework Components

- Zend_Filter

5. Theory of Operation

The `(bool)` and `(boolean)` casts in PHP consider `NULL` and empty strings to be `FALSE`, and any value that is not specifically `FALSE` to be `TRUE`. Zend_Filter_Bool is intended as an alternative to casting for applications where such assumptions will cause logic errors.

A different set of needs is addressed by filtering as opposed to casting to boolean:

- It is not always appropriate to assume intent regarding empty or unset input. If a default value is desired, it should be defined by setting the Zend_Filter_Input::DEFAULT_VALUE metacommand, or by checking/validating for a `NULL` value from the results of this filter.
- It is helpful to filter for valid boolean literals used by some databases that would not be recognized when casting.

The following values are considered `FALSE`:

- the boolean `FALSE` itself
- numeric values that equal 0 such as 0 or 0.0
- numeric strings that can be cast to the integer 0 such as "0.0" or "00"
- the string "false" (any capitalization allowed)
- the string "f" (any capitalization allowed)
- the string "no" (any capitalization allowed)
- the string "n" (any capitalization allowed)
- the string "off" (any capitalization allowed)

The following values are considered `TRUE`:

- the boolean `TRUE` itself
- numeric values that equal 1 such as 1 or 1.0
- numeric strings that can be cast to the integer 1 such as "1.0" or "01"
- the string "true" (any capitalization allowed)
- the string "t" (any capitalization allowed)
- the string "yes" (any capitalization allowed)
- the string "y" (any capitalization allowed)
- the string "on" (any capitalization allowed)

The filter will output `NULL` upon attempting to process any value not listed above.

6. Milestones / Tasks

- Milestone 1: [DONE] Proposal finished
- Milestone 2: Proposal accepted
- Milestone 3: Working prototype checked into the incubator supporting use cases.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

7. Class Index

- Zend_Filter_Bool

8. Use Cases

UC-01

```
$filter = new Zend_Filter_Bool();
print $filter(true);
// outputs TRUE
```

UC-02

```
$filter = new Zend_Filter_Bool();
print $filter(1);
// outputs TRUE
```

UC-03

```
$filter = new Zend_Filter_Bool();
print $filter('1');
// outputs TRUE
```

UC-04

```
$filter = new Zend_Filter_Bool();
print $filter('1.0');
// outputs TRUE
```

UC-05

```
$filter = new Zend_Filter_Bool();
print $filter('01');
// outputs TRUE
```

UC-06

```
$filter = new Zend_Filter_Bool();
print $filter('true');
// outputs TRUE
```

UC-07

```
$filter = new Zend_Filter_Bool();
print $filter('t');
// outputs TRUE
```

UC-08

```
$filter = new Zend_Filter_Bool();
print $filter('yes');
// outputs TRUE
```

UC-09

```
$filter = new Zend_Filter_Bool();
print $filter('y');
// outputs TRUE
```

UC-10

```
$filter = new Zend_Filter_Bool();
print $filter('on');
// outputs TRUE
```

UC-11

```
$filter = new Zend_Filter_Bool();
print $filter(false);
// outputs FALSE
```

UC-12

```
$filter = new Zend_Filter_Bool();
print $filter(0);
// outputs FALSE
```

UC-13

```
$filter = new Zend_Filter_Bool();
print $filter('0');
// outputs FALSE
```

UC-14

```
$filter = new Zend_Filter_Bool();
print $filter('0.0');
// outputs FALSE
```

UC-15

```
$filter = new Zend_Filter_Bool();
print $filter('00');
// outputs FALSE
```

UC-16

```
$filter = new Zend_Filter_Bool();
print $filter('false');
// outputs FALSE
```

UC-17

```
$filter = new Zend_Filter_Bool();
print $filter('f');
// outputs FALSE
```

UC-18

$filter = new Zend_Filter_Bool();
print $filter('no');
// outputs FALSE

UC-19

$filter = new Zend_Filter_Bool();
print $filter('n');
// outputs FALSE

UC-20

$filter = new Zend_Filter_Bool();
print $filter('off');
// outputs FALSE

UC-21

$filter = new Zend_Filter_Bool();
print $filter(null);
// outputs NULL

UC-22

$filter = new Zend_Filter_Bool();
print $filter('');
// outputs NULL

UC-23

$filter = new Zend_Filter_Bool();
print $filter('unhandled input');
// outputs NULL

9. Class Skeletons

```
class Zend_Filter_Bool implements Zend_Filter_Interface
{
    /**
     * @param  mixed $value
     * @return bool|null
     */
    public function filter($value);
}
```

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