clean/assure Documentation

Release latest

Roman Nowicki

August 12, 2016

CHAPTER 1

Data correction and validation tools for PHP

This package extends global namespace with `assure` method than can be used to correct and validate mixed data types.

Mainly used to support duck typing of primitive types in function parameters to simplify interface usage eg.:

```
class Foo
{
    public function filterById($id)
    {
        assure($name, ['arrayOfIntegers']);
        // after assure we can trust that name is an array of integers
        // otherwise exception will be raised
    }
}

$foo = new Foo();
$foo->filterById(1)
$foo->filterById('1')
$foo->filterById(['1', 2])
```

1.1 Example Of Usage

```
// if value is not integer and cannot be transform to integer assure will throw an exception
// correct values: '1', 1, 1.3
// invalid values: 'a', NULL, false, array()
assure($value, 'integer');

// if not integer OR string with integers separated by commas assure will throw an exception
// correct values: '1', '1,2,3,4,5';
// invalid values: 'a', '1,2,a,4,b';
assure($value, ['integer', 'commaSeparatedIntegers']);
```

1.2 Installation

via Composer: `composer require clean/assure`

1.2.1 ArrayOfIntegers

Ensure that variable is an array with elements that can be treated as integers. Each element of array is validated and converted to int on success.

Correct:

```
assure($x = 1, 'arrayOfIntegers');                  // $x => [1]
assure($x = [1,2], 'arrayOfIntegers');              // $x => [1,2]
assure($x = (object)[1,2], 'arrayOfIntegers')[1,2]) // $x => [1,2]
```

Incorect:

```
assure($x = [], 'arrayOfIntegers');
assure($x = [null], 'arrayOfIntegers');
assure($x = ['a'], 'arrayOfIntegers');
assure($x = [1,'a'], 'arrayOfIntegers');
assure($x = (object)[1,'a'], 'arrayOfIntegers');
```

All above method calls will throw `\InvalidArgumentException`

1.2.2 ArrayOfStrings

Ensure that variable is an array with elements that can be treated as string. Each element of array is validated and converted to string on success.

Correct:

```
assure($x = 'a', 'arrayOfStrings');                    // $x = ['a']
assure($x = 1, 'arrayOfStrings');                      // $x = ['1']
assure($x = [1, 'two'], 'arrayOfStrings');             // $x = ['1', 'two']
assure($x = (object)['one', 'two'], 'arrayOfStrings'); // $x = ['one', 'two']
```

Incorect:

```
assure($x = [], 'arrayOfStrings');
assure($x = [null], 'arrayOfStrings');
assure($x = 1, 'arrayOfStrings');
assure($x = [1,'a'], 'arrayOfStrings');
assure($x = (object)[1,'a'], 'arrayOfStrings');
```

All above method calls will throw `\InvalidArgumentException`

1.2.3 Boolean

Ensure that variable has value that can be treated as bool value. Variable is validated and converted to bool value on success.

Correct:

```
assure($x = true, 'boolean');  // $x => true
assure($x = false, 'boolean');  // $x => false
assure($x = 1, 'boolean');  // $x => true
assure($x = 0, 'boolean');  // $x => false
assure($x = 'true', 'boolean');  // $x => true
assure($x = 'false', 'boolean');  // $x => false
assure($x = 'TRUE', 'boolean');  // $x => true
assure($x = 'FALSE', 'boolean');  // $x => false
```

```
assure($x = 'True', 'boolean');  // $x => true
assure($x = 'False', 'boolean');  // $x => false
assure($x = 'on', 'boolean');  // $x => true
assure($x = 'off', 'boolean');  // $x => false
assure($x = 'ON', 'boolean');  // $x => true
assure($x = 'OFF', 'boolean');  // $x => false
assure($x = 'YES', 'boolean');  // $x => true
assure($x = 'NO', 'boolean');  // $x => false
assure($x = 'yes', 'boolean');  // $x => true
assure($x = 'no', 'boolean');  // $x => false
assure($x = 'falsE', 'boolean');  // $x => true
assure($x = 'truE', 'boolean');  // $x => false
```

Incorect:

```
assure($x = '2', 'boolean');
assure($x = 2, 'boolean');
assure($x = new \stdClass, 'boolean');
assure($x = 'a', 'boolean');
assure($x = 'one', 'boolean');
```

All above method calls will throw `\InvalidArgumentException`

1.2.4 CommaSeparatedIntegers

Ensure that variable is string with comma separated integers and transform it to array of integers on success

Correct:

```
assure($x = '1,2,3,4', 'commaSeparatedIntegers'); // $x => [1,2,3,4]
assure($x = '1', 'commaSeparatedIntegers');       // $x => [1]
assure($x = '1,', 'commaSeparatedIntegers');      // $x => [1]
assure($x = ',1', 'commaSeparatedIntegers');      // $x => [1]
assure($x = 1, 'commaSeparatedIntegers');         // $x => [1]
assure($x = 0, 'commaSeparatedIntegers');         // $x => [0]
assure($x = '0', 'commaSeparatedIntegers');       // $x => [0]
```

Incorect:

```
assure($x = [], 'commaSeparatedIntegers');
assure($x = null, 'commaSeparatedIntegers');
assure($x = '', 'commaSeparatedIntegers');
assure($x = 'a,2,3,4', 'commaSeparatedIntegers');
```

All above method calls will throw `\InvalidArgumentException`

1.2.5 CommaSeparatedStrings

Ensure that variable is a string with comma separated strings inside and transform it to array of strings on success

Correct:

```
assure($x = '0', 'commaSeparatedStrings');       // $x => ['0']
assure($x = 'a', 'commaSeparatedStrings');       // $x => ['a']
assure($x = 'a,', 'commaSeparatedStrings');      // $x => ['a']
```

```
assure($x = ',a', 'commaSeparatedStrings');      // $x => ['a']
assure($x = ',a,b', 'commaSeparatedStrings');    // $x => ['a', 'b']
assure($x = '1,2,3,4', 'commaSeparatedStrings'); // $x => ['1','2','3','4']
assure($x = 'a,b,c', 'commaSeparatedStrings');   // $x => ['a', 'b', 'c']
assure($x = '1#),#$%', 'commaSeparatedStrings'); // $x => ['1#)', '#$%']
```

Incorect:

```
assure($x = '[]', 'commaSeparatedStrings');
assure($x = '', 'commaSeparatedStrings');
assure($x = null, 'commaSeparatedStrings');
```

All above method calls will throw `\InvalidArgumentException`

1.2.6 Date

Validate if variable is a string that can be transformed to timestamp, and converted it to timestamp on success.

Correct:

```
'1983-02-02'
'12/15/1990'
'now'
'10 September 2000'
```

Incorect:

```
'2',
'15/11/1990',
'sometext',
'-',
```

All above will throw `\InvalidArgumentException`

1.2.7 EncodedUri

Ensure that string is encoded as valid uri path. It will convert all uri path elements by urlencode method

Example:

```
'/news//2014/03/14/4683864/'

will be converted to:

'/news/%E3%83%A8%E3%83%BC%E3%83%AD%E3%83%83%E3%83%91%E3%83%AA%E3%83%BC%E3%82%B0/2014/03/14/4
```

1.2.8 Float

Ensure that variable has value that can be treated as float value. Variable is validated and converted to float value on success.

Correct:

```
assure($x = 1.1, 'float'); // $x => 1.1
assure($x = '1.1', 'float'); // $x => 1.1
assure($x = 1, 'float'); // $x => 1.0
assure($x = '1', 'float'); // $x => 1.0
```

Incorect:

```
assure($x = null, 'float');
assure($x = 'one', 'float');
assure($x = false, 'float');
assure($x = true, 'float');
assure($x = [], 'float');
assure($x = <object>, 'float');
```

All above values will throw `\InvalidArgumentException`

1.2.9 Integer

Ensure that variable has value that can be treated as integer value. Variable is validated and converted to integer on success.

Correct:

```
1     => 1
'1'   => 1
'-1'  => -1
```

Incorect:

```
null
1.1
'one'
false
true
[]
```

All above values will throw `\InvalidArgumentException`

1.2.10 Null

Ensure that variable has null value assigned

Correct:

```
assure($x = null, 'null');
```

Incorect:

```
assure($x = 1, 'null');
assure($x = false, 'null');
assure($x = true, 'null');
assure($x = '', 'null');
assure($x = [], 'null');
```

All above values will throw `\InvalidArgumentException`

1.2.11 Object

Ensure that variable is an object. Associative array will be transformed automatcially to stdClass.

Correct:

```
assure($x = new stdClass, 'object');
assure($x = new SomeObject, 'object');
assure($x = ['x'=> 10, 'y'=20], 'object');
```

Incorect:

```
assure($x = 1, 'object');
assure($x = false, 'object');
assure($x = true, 'object');
assure($x = '', 'object');
assure($x = [], 'object');
```

All above values will throw `\InvalidArgumentException`