ClickBlocks\Data\Validators\Validator

General information

Inheritance	no
Child classes	ClickBlocks\Data\Validators\Type, ClickBlocks\Data\Validators\Number, ClickBlocks\Data\Validators\Collection, ClickBlocks\Data\Validators\Text, ClickBlocks\Data\Validators\Regex, ClickBlocks\Data\Validators\Required, ClickBlocks\Data\Validators\Compare, ClickBlocks\Data\Validators\XML, ClickBlocks\Data\Validators\JSON, ClickBlocks\Data\Validators\Custom.
Interfaces	no
Source	Framework/data/validators/validator.php

The base abstract class for all validator classes.

Public static methods

getInstance()

```
final public static ClickBlocks\Data\Validators getInstance(string $type, array $params = [])
```

$type	string	the validator type. The valid values are "type", "required", "compare", "regex", "text", "number", "collection", "xml", "json", "custom".
$params	string	the values of the validator properties.

Creates and returns a validator object of the required type.

verify()

```
public static boolean verify(mixed $value, array $validators, string $mode = 'and', array &$result = null)
```

$value	mixed	the value to be verified.
$validators	array	the validators to be matched with the given value.
$mode	string	determines the way of the calculation of the validation result for multiple validators. The valid values are "and", "or", "xor". The "and" value requires that all validators are valid. The "or" value requires that at least one validator is valid. And the "xor" values requires that exactly one validator is valid.
$result	array	array of the results of the validators work. Each array element contains TRUE if the appropriate validator is valid, otherwise the element contains the reason of validator failure.

Verifies the value for matching with one or more validators according to the **$mode** value. Example:

```
// Array of the validators:
$vals = [];
$vals['type'] = Validator::getInstance('type', ['type' => 'string']);
$vals['compare'] = Validator::getInstance('compare', ['value' => 'test']);
$vals['regex'] = Validator::getInstance('regex', ['pattern' => '/[0-9]+/']);
// Verifies some value:
$result = [];
if (Validator::verify('test', $vals, 'or', $result))
{
```

```
  // Verification is passed.
  print_r($result);
  // The output is shown below:
  // Array
  // (
  //     [type] => 1
  //     [compare] => 1
  //     [regex] => Array
  //                (
  //                    [code] => 0
  //                    [reason] => doesn't match
  //                )
  //  )
}
```

Public non-static properties

empty

```
public boolean $empty = true
```

The general property of all validators (except for **ClickBlocks\Data\Validators\Type**). Determines whether the value can be null or empty. If **$empty** is TRUE then validating empty value will be considered valid.

Public non-static methods

getReason()

```
public array getReason()
```

Returns the reason of validator failure. If the validator returns TRUE the reason will be TRUE as well otherwise the method returns an array with failure code and message.

validate()

```
abstract public boolean validate(mixed $entity)
```

$entity	mixed	the value to validate.

Validates a single value. The method returns TRUE if the validating value matches the validation conditions and FALSE otherwise.

Protected non-static properties

reason

```
protected array $reason
```

The reason of the last validator failure. The reason equals TRUE if the validator returns TRUE. The structure of the reason array as follows:

```
['code'    => ... // the reson code.
 'reason'  => ... // the short reason message.
 'details' => ... // the reason details. It can be omitted for some validators.    ]
```

Protected non-static method

isEmpty()

```
protected boolean isEmpty(mixed $entity)
```

$entity	mixed	the validating value.

Checks whether the given value is empty. For scalar values the method returns TRUE if the given value is empty (has zero length) and FALSE otherwise. If the given value is an array the methods return TRUE if this array has not any elements and FALSE otherwise. If the given value is an object the methods return TRUE if this object has not any public property and FALSE otherwise.