
Flask-Validator Documentation

Release 1.4

Jesus Roldan

Oct 27, 2018

1	Installation	3
2	Basic Usage	5
3	Validators availables	7
3.1	ValidateInteger	8
3.2	ValidateNumeric	8
3.3	ValidateString	9
3.4	ValidateBoolean	9
3.5	ValidateLength	9
3.6	ValidateNumber	9
3.7	ValidateLessThan	10
3.8	ValidateLessThanOrEqual	10
3.9	ValidateGreaterThan	10
3.10	ValidateGreaterThanOrEqual	10
3.11	ValidateEmail	11
3.12	ValidateRegex	11
3.13	ValidateRange	11
3.14	ValidateIP	11
3.15	ValidateURL	12
3.16	ValidateUUID	12
3.17	ValidateCountry	12
3.18	ValidateTimezone	13
3.19	ValidateLocale	13
3.20	ValidateCreditCard	13
3.21	ValidateCurrency	14
3.22	ValidateIBAN	14
3.23	ValidateBIC	14
3.24	ValidateISBN	14
4	Custom Validators	17
5	Exception Message	19
6	Indices and tables	21

Data validator for Flask using SQL-Alchemy, working at Model component with events, preventing invalid data in the columns. The extension works with event listeners from SQLAlchemy.

The code is available in this [repository](#)

Contents:

CHAPTER 1

Installation

Flask-Validator is available via `pip`, running the following command

```
pip install flask_validator
```

It might take a minute or so, it has quite a few things to download and install.

CHAPTER 2

Basic Usage

The most performant way to set up your validations is using the SQLAlchemy special directive `__declare_last__`, it occurs after mappings are assumed to be completed and the 'configure' step has finished. With this method, you will create the event one time, just before the class creation.

The only required argument is the Column to validate.

```
# import ...
from flask_validator import ValidateInteger, ValidateString, ValidateEmail

class User(db.Model):
    __tablename__ = 'user'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80))
    code = db.Column(db.Integer())
    email = db.Column(db.String(125))

    def __init__(self, string, integer):
        self.string = string
        self.integer = integer

    @classmethod
    def __declare_last__(cls):
        ValidateString(User.name)
        ValidateInteger(User.code)
        ValidateEmail(User.email, true, true, "The e-mail is not valid. Please check it
↪")
```

With that code, the Validator will execute an ORM event listening each field modification

Validators availables

At the moment, the library support this validations:

- Types
 - *ValidateInteger*
 - *ValidateNumeric*
 - *ValidateString*
 - *ValidateBoolean*
- Numeric
 - *ValidateLength*
 - *ValidateNumber*
- Comparison
 - *ValidateLessThan*
 - *ValidateLessThanOrEqual*
 - *ValidateGreaterThan*
 - *ValidateGreaterThanOrEqual*
- Internet
 - *ValidateEmail*
 - *ValidateIP*
 - *ValidateURL*
- Location
 - *ValidateCountry*
 - *ValidateTimezone*
 - *ValidateLocale*

- Financial
 - *ValidateCreditCard*
 - *ValidateCurrency*
 - *in_iban*
 - *in_bic*
- Others
 - *ValidateISBN*
 - *ValidateUUID*
 - *ValidateRegex*
 - *ValidateRange*

3.1 ValidateInteger

Check if the new value is a valid `int` or `long` type

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_null</code>	True	Allow null values
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

Note: `long` type is only available i Python 2.7

3.2 ValidateNumeric

Check if the new value is a valid `int`, `long`, `float` or `complex` type

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_null</code>	True	Allow null values
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

Note: `long` type is only available i Python 2.7

3.3 ValidateString

Check if the new value is a valid `string` type.

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_null</code>	True	Allow null values
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.4 ValidateBoolean

Check if the new value is a valid `bool` type.

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.5 Validatelength

Check if the new value has a length with a maximum and a minimum

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>max_length</code>	None	Maximum value length
<code>min_length</code>	0	Minimum value length
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.6 ValidateNumber

Check if the new value is a number or not (NaN)

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.7 ValidateLessThan

Check if the new value is a lesser than X value

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.8 ValidateLessThanOrEqual

Check if the new value is a lesser than X value or equal

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.9 ValidateGreaterThan

Check if the new value is a greater than X value

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.10 ValidateGreaterThanOrEqual

Check if the new value is a greater than X value or equal

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.11 ValidateEmail

Check if the new value is a valid e-mail, using `email_validator` library.

Parameters:

Parameter	De- fault	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_smtputf8</code>	True	Allow internationalized addresses that would require the <code>SMTPUTF8</code> extension.
<code>check_deliverability</code>	True	Check domain name resolution.
<code>allow_empty_local</code>	False	Allow an empty local part for validating Postfix aliases.
<code>allow_null</code>	True	Allow <code>null</code> values
<code>throw_exception</code>	False	Throw a <code>ValidateError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidateError</code> exception

3.12 ValidateRegex

Compare a value against a regular expression

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>throw_exception</code>	False	Throw a <code>ValidateError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidateError</code> exception

3.13 ValidateRange

Check if the new value is in a range

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>range</code>		Range values
<code>allow_null</code>	True	Allow <code>null</code> values
<code>throw_exception</code>	False	Throw a <code>ValidateError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidateError</code> exception

3.14 ValidateIP

Check if the value is a valid IP Address

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
ipv6	False	Check IPv6 Address instead of IPv4
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.15 ValidateURL

Check if the value is a valid URL

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.16 ValidateUUID

Check if the value is a valid UUID

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
version	4	UUID version
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.17 ValidateCountry

Check if the value is a valid Country. Validation provided by [iso3166](#). Allowed names:

- Name
- Alpha2
- Alpha3
- Numeric
- Apolitical Name

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.18 ValidateTimezone

Check if the value is a valid Timezone. Validation provided by `pytz`

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.19 ValidateLocale

Check if the value is a valid Locale.

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.20 ValidateCreditCard

Check if the new value is valid credit card number.

Allowed formats: * `XXXXYYYYWWWWZZZ` * `“XXXXYYYYWWWWZZZ”` * `“XXXX YYYY WWWW ZZZ”`
* `“XXXX-YYYY-WWWW-ZZZ”`

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

3.21 ValidateCurrency

Check if the new value is a valid Currency

Validation provided by: `moneyed`

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_null</code>	True	Allow null values
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.22 ValidateIBAN

Check if the new value is valid IBAN (International Bank Account Number)

More reference: https://en.wikipedia.org/wiki/International_Bank_Account_Number

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_null</code>	True	Allow null values
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.23 ValidateBIC

Check if the new value is valid BIC (SO 9362 defined standard format of Bank Identifier Codes)

More reference: https://en.wikipedia.org/wiki/ISO_9362

Parameters:

Parameter	Default	Description
<code>field</code>		SQLAlchemy column to validate
<code>allow_null</code>	True	Allow null values
<code>throw_exception</code>	False	Throw a <code>ValidationError</code> exception on validation fails
<code>message</code>	None	Add a custom message to the <code>ValidationError</code> exception

3.24 ValidateISBN

Check if the new value is valid ISBN (International Standard Book Number). Allows ISBN10 or ISBN13

Validation provided by: `isbnlib` More reference: https://en.wikipedia.org/wiki/International_Standard_Book_Number

Parameters:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow null values
throw_exception	False	Throw a <code>ValidationError</code> exception on validation fails
message	None	Add a custom message to the <code>ValidationError</code> exception

Custom Validators

You will be able to create custom validator implementing the class `Validator`.

You must define your own method `check_value()` and if you are receiving any argument, you also must call the parent `__init__()`

```
from flask_validator import Validator

class ValidateAorB(Validator)
    def __init__(self, field, useless, allow_null=True, throw_exception=False,
↳message=None):
        self.useless = useless

        Validator.__init__(self, field, allow_null, throw_exception, message):

    def check_value(self, value):
        return if value in ['A', 'B']

validator = ValidateAorB(field, True, True, 'yadayada')
```

Exception Message

You will be able to create custom exception messages, with a few variables available:

- `field`: Object and property
- `key`: property
- `new_value`: New value changed
- `old_value`: Previous value

```
from flask_validator import ValidateEmail

validator = ValidateEmail(field, False, True, 'Message: Field {field}, Key {key}, New_
↪value {new_value}, Old value {old_value}')
```


CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`