Flask-Validator Documentation

Release 1.4

Jesus Roldan

Dec 18, 2018

Contents

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 avalable 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 uring 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 tine, 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

CHAPTER 3

Validators availables

At the moment, the library support this validations:

- Types
  - *ValidateInteger*
  - *ValidateNumeric*
  - *ValidateString*
  - *ValidateBoolean*
- Numeric
  - *Validatelength*
  - *ValidateNumber*
- Comparision
  - *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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` 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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

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

3.3 ValidateString

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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.4 ValidateBoolean

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

Parametes:

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

3.5 Validatelength

Check if the new value has a length with a maximun and a minimun

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
max_length	None	Maximum value length
min_length	0	Minumum value length
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.6 ValidateNumber

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

Parametes:

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

3.7 ValidateLessThan

Check if the new value is a lesser than X value

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.8 ValidateLessThanOrEqual

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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.9 ValidateGreaterThan

Check if the new value is a greater than X value

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.10 ValidateGreaterThanOrEqual

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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
value		Value to check
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.11 ValidateEmail

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

Parametes:

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

3.12 ValidateRegex

Compare a value against a regular expresion

Parametes:

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

3.13 ValidateRange

Check if the new value is in a range

Parametes:

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

3.14 ValidateIP

Check if the value is a valid IP Address

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
ipv6	False	Check IPv6 Address instread of IPv4
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.15 ValidateURL

Check if the value is a valid URL

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.16 ValidateUUID

Check if the value is a valid UUUID

Parametes:

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

3.17 ValidateCountry

Check if the value is a valid Country. Validation provided by iso3166. Allowed names:

- Name
- Alpha2
- Alpha3
- Numeric
- Apolitic Name

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.18 ValidateTimezone

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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.19 ValidateLocale

Check if the value is a valid Locale.

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` 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”

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

3.21 ValidateCurrency

Check if the new value is a valid Currency

Validation provided by: moneyed

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` 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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` 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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` 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

Parametes:

Parameter	Default	Description
field		SQLAlchemy column to validate
allow_null	True	Allow `null` values
throw_exception	False	Throw a `ValidateError` exception on validation fails
message	None	Add a custom message to the `ValidateError` exception

CHAPTER 4

Custom Validators

You will be able to create customs 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')
```

CHAPTER 5

Exception Message

You will be able to create customs exception messages, with a few variables availables:

- `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