Bootstrap-Flask Documentation

Release 1.0.4

Grey Li

Sep 05, 2021

Contents

Bootstrap 4 helper for Flask/Jinja2.

CHAPTER 1

Contents

1.1 Basic Usage

1.1.1 Installation

```
$ pip install bootstrap-flask
```

This project can't work with Flask-Bootstrap at the same time. If you have already installed Flask-Bootstrap in the same Python enviroment, you have to uninstall it and then reinstall this project:

```
$ pip uninstall flask-bootstrap bootstrap-flask
$ pip install bootstrap-flask
```

Tip: See *Migrate from Flask-Bootstrap* to learn how to migrate from Flask-Bootstrap.

1.1.2 Initialization

```
from flask_bootstrap import Bootstrap
from flask import Flask

app = Flask(__name__)

bootstrap = Bootstrap(app)
```

1.1.3 Resources helpers

Bootstrap-Flask provides two helper functions to load Bootstrap resources in the template: `bootstrap.load_css()` and `bootstrap.load_js()`.

Call it in your base template, for example:

```
<head>
....
{{ bootstrap.load_css() }}
</head>
<body>
...
{{ bootstrap.load_js() }}
</body>
```

You can pass `version` to pin the Bootstrap 4 version you want to use. It defaults to load files from CDN. Set `BOOTSTRAP_SERVE_LOCAL` to `True` to use built-in local files. However, these methods are optional, you can also write `<href></href>` and `<script></script>` tags to include Bootstrap resources (from your `static` folder or CDN) manually by yourself.

1.1.4 Starter template

For reasons of flexibility, Bootstrap-Flask doesn't include built-in base templates (this may change in the future). For now, you have to create a base template yourself. Be sure to use an HTML5 doctype and include a viewport meta tag for proper responsive behaviors. Here's an example base template:

```
<!doctype html>
<html lang="en">
    <head>
        {% block head %}
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
↪fit=no">

        {% block styles %}
            <!-- Bootstrap CSS -->
            {{ bootstrap.load_css() }}
        {% endblock %}

        <title>Your page title</title>
        {% endblock %}
    </head>
    <body>
        <!-- Your page content -->
        {% block content %}{% endblock %}

        {% block scripts %}
            <!-- Optional JavaScript -->
            {{ bootstrap.load_js() }}
        {% endblock %}
    </body>
</html>
```

Use this in your templates folder (suggested names are `base.html` or `layout.html` etc.), and inherit it in child templates. See Template Inheritance for more details on inheritance.

1.1.5 Macros

Macro	Templates Path	Description
render_field()	bootstrap/form.html	Render a WTForms form field
render_form()	bootstrap/form.html	Render a WTForms form
render_form_row()	bootstrap/form.html	Render a row of a grid form
render_hidden_errors()	bootstrap/form.html	Render error messages for hidden form field
render_pager()	bootstrap/pagination.html	Render a basic Flask-SQLAlchemy pagniantion
render_pagination()	bootstrap/pagination.html	Render a standard Flask-SQLAlchemy pagination
render_nav_item()	bootstrap/nav.html	Render a navigation item
render_breadcrumb_item()	bootstrap/nav.html	Render a breadcrumb item
render_static()	bootstrap/utils.html	Render a resource reference code (i.e. `<link>`, `<script>`)
render_messages()	bootstrap/utils.html	Render flashed messages send by flash() function
render_icon()	bootstrap/utils.html	Render a Bootstrap icon
render_table()	bootstrap/table.html	Render a table with given data

How to use these macros? It's quite simple, just import them from the corresponding path and call them like any other macro:

```
{% from 'bootstrap/form.html' import render_form %}

{{ render_form(form) }}
```

Go to the *Use Macros* page to see the detailed usage for these macros.

1.1.6 Configurations

Configuration Variable	Default Value	Description
BOOTSTRAP_SERVE_LOCAL	`False`	If set to `True`, local resources will be used for `load_*` methods.
BOOTSTRAP_BTN_STYLE	`'primary'`	Default form button style, will change to `primary` in next major release
BOOTSTRAP_BTN_SIZE	`'md'`	Default form button size
BOOTSTRAP_ICON_SIZE	`'1em'`	Default icon size
BOOTSTRAP_ICON_COLOR	`None`	Default icon color, follow the context with `currentColor` if not set
BOOT-STRAP_BOOTSWATCH_THEME	`None`	Bootswatch theme to use, see available themes at *Bootswatch Themes*
BOOT-STRAP_MSG_CATEGORY	`'primary'`	Default flash message category
BOOT-STRAP_TABLE_VIEW_TITLE	`'View'`	Default title for view icon of table actions
BOOT-STRAP_TABLE_EDIT_TITLE	`'Edit'`	Default title for edit icon of table actions
BOOT-STRAP_TABLE_DELETE_TITLE	`'Delete'`	Default title for delete icon of table actions
BOOT-STRAP_TABLE_NEW_TITLE	`'New'`	Default title for new icon of table actions

Tip: See *Form Button Customization* to learn how to customize form buttons.

1.2 Use Macros

These macros will help you to generate Bootstrap-markup codes quickly and easily.

1.2.1 render_nav_item()

Render a Bootstrap nav item.

Example

```
{% from 'bootstrap/nav.html' import render_nav_item %}

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="navbar-nav mr-auto">
        {{ render_nav_item('index', 'Home') }}
        {{ render_nav_item('explore', 'Explore') }}
        {{ render_nav_item('about', 'About') }}
    </div>
</nav>
```

API

render_nav_item(*endpoint*, *text*, *badge=”*, *use_li=False*, ***kwargs*)

Parameters

- **endpoint** – The endpoint used to generate URL.
- **text** – The text that will displayed on the item.
- **badge** – Badge text.
- **use_li** – Default to generate `<a></a>`, if set to `True`, it will generate `<li><a></a></li>`.
- **kwargs** – Additional keyword arguments pass to `url_for()`.

1.2.2 render_breadcrumb_item()

Render a Bootstrap breadcrumb item.

Example

```
{% from 'bootstrap/nav.html' import render_breadcrumb_item %}

<nav aria-label="breadcrumb">
    <ol class="breadcrumb">
        {{ render_breadcrumb_item('home', 'Home') }}
        {{ render_breadcrumb_item('users', 'Users') }}
        {{ render_breadcrumb_item('posts', 'Posts') }}
        {{ render_breadcrumb_item('comments', 'Comments') }}
    </ol>
</nav>
```

API

render_breadcrumb_item(*endpoint*, *text*, ***kwargs*)

Parameters

- **endpoint** – The endpoint used to generate URL.
- **text** – The text that will displayed on the item.
- **kwargs** – Additional keyword arguments pass to `url_for()`.

1.2.3 render_field()

Render a form input for form field created by Flask-WTF/WTForms.

Example

```
{% from 'bootstrap/form.html' import render_field %}

<form method="post">
    {{ form.csrf_token() }}
    {{ render_field(form.username) }}
    {{ render_field(form.password) }}
    {{ render_field(form.submit) }}
</form>
```

API

`render_field`(*field*, *form_type="basic"*, *horizontal_columns=('lg', 2, 10)*, *button_style=""*, *button_size=""*, *button_map={}*)

Parameters

- **`field`** – The form field (attribute) to render.
- **`form_type`** – One of `basic`, `inline` or `horizontal`. See the Bootstrap docs for details on different form layouts.
- **`horizontal_columns`** – When using the horizontal layout, layout forms like this. Must be a 3-tuple of `(column-type, left-column-size, right-column-size)`.
- **`button_style`** – Accept Bootstrap button style name (i.e. primary, secondary, outline-success, etc.), default to `secondary` (e.g. `btn-secondary`). This will overwrite config `BOOTSTRAP_BTN_STYLE`.
- **`button_size`** – Accept Bootstrap button size name: sm, md, lg, block, default to `md`. This will overwrite config `BOOTSTRAP_BTN_SIZE`.
- **`button_map`** – A dictionary, mapping button field name to Bootstrap button style names. For example, `{'submit': 'success'}`. This will overwrite `button_style` and `BOOTSTRAP_BTN_STYLE`.

Tip: See *Form Button Customization* to learn how to customize form buttons.

1.2.4 render_form()

Render a complete form element for form object created by Flask-WTF/WTForms.

Example

```
{% from 'bootstrap/form.html' import render_form %}

{{ render_form(form) }}
```

API

`render_form`(*form*, *action=""*, *method="post"*, *extra_classes=None*, *role="form"*, *form_type="basic"*, *horizontal_columns=('lg', 2, 10)*, *enctype=None*, *button_style=""*, *button_size=""*, *button_map={}*, *id=""*, *novalidate=False*, *render_kw={}*)

Parameters

- **form** – The form to output.
- **action** – The URL to receive form data.
- **method** – `<form>` method attribute.
- **extra_classes** – The classes to add to the `<form>`.
- **role** – `<form>` role attribute.
- **form_type** – One of `basic`, `inline` or `horizontal`. See the Bootstrap docs for details on different form layouts.
- **horizontal_columns** – When using the horizontal layout, layout forms like this. Must be a 3-tuple of `(column-type, left-column-size, right-column-size)`.
- **enctype** – `<form>` enctype attribute. If `None`, will automatically be set to `multipart/form-data` if a `FileField` or `MultipleFileField` is present in the form.
- **button_style** – Accept Bootstrap button style name (i.e. primary, secondary, outline-success, etc.), default to `secondary` (e.g. `btn-secondary`). This will overwrite config `BOOTSTRAP_BTN_STYLE`.
- **button_size** – Accept Bootstrap button size name: sm, md, lg, block, default to `md`. This will overwrite config `BOOTSTRAP_BTN_SIZE`.
- **button_map** – A dictionary, mapping button field name to Bootstrap button style names. For example, `{'submit': 'success'}`. This will overwrite `button_style` and `BOOTSTRAP_BTN_STYLE`.
- **id** – The `<form>` id attribute.
- **novalidate** – Flag that decide whether add `novalidate` class in `<form>`.
- **render_kw** – A dictionary, specifying custom attributes for the `<form>` tag.

Tip: See *Form Button Customization* to learn how to customize form buttons.

1.2.5 render_hidden_errors()

Render error messages for hidden form field (`wtforms.HiddenField`).

Example

```
{% from 'bootstrap/form.html' import render_field, render_hidden_errors %}

<form method="post">
    {{ form.hidden_tag() }}
    {{ render_hidden_errors(form) }}
    {{ render_field(form.username) }}
    {{ render_field(form.password) }}
    {{ render_field(form.submit) }}
</form>
```

API

render_hidden_errors (*form*)

Parameters **form** – Form whose errors should be rendered.

1.2.6 render_form_row()

Render a row of a grid form with the given fields.

Example

```
{% from 'bootstrap/form.html' import render_form_row %}

<form method="post">
    {{ form.csrf_token() }}
    {{ render_form_row([form.username, form.password]) }}
    {{ render_form_row([form.remember]) }}
    {{ render_form_row([form.submit]) }}
    {# Custom col which should use class col-md-2, and the others the defaults: #}
    {{ render_form_row([form.title, form.first_name, form.surname], col_map={'title':
↪'col-md-2'}) }}
    {# Custom col which should use class col-md-2 and modified col class for the␣
↪default of the other fields: #}
    {{ render_form_row([form.title, form.first_name, form.surname], col_class_default=
↪'col-md-5', col_map={'title': 'col-md-2'}) }}
</form>
```

API

render_form_row (*fields*, *row_class='form-row'*, *col_class_default='col'*, *col_map={}*, *button_style=""*, *button_size=""*, *button_map={}*)

Parameters

- **fields** – An iterable of fields to render in a row.
- **row_class** – Class to apply to the div intended to represent the row, like `form-row` or `row`
- **col_class_default** – The default class to apply to the div that represents a column if nothing more specific is said for the div column of the rendered field.
- **col_map** – A dictionary, mapping field.name to a class definition that should be applied to the div column that contains the field. For example: `col_map={'username': 'col-md-2'})`
- **button_style** – Accept Bootstrap button style name (i.e. primary, secondary, outline-success, etc.), default to `secondary` (e.g. `btn-secondary`). This will overwrite config `BOOTSTRAP_BTN_STYLE`.
- **button_size** – Accept Bootstrap button size name: sm, md, lg, block, default to `md`. This will overwrite config `BOOTSTRAP_BTN_SIZE`.
- **button_map** – A dictionary, mapping button field name to Bootstrap button style names. For example, `{'submit': 'success'}`. This will overwrite `button_style` and `BOOTSTRAP_BTN_STYLE`.

Tip: See *Form Button Customization* to learn how to customize form buttons.

1.2.7 render_pager()

Render a simple pager for query pagination object created by Flask-SQLAlchemy.

Example

```
{% from 'bootstrap/pagination.html' import render_pager %}

{{ render_pager(pagination) }}
```

API

render_pager(*pagination, fragment=", prev=('<span aria-hidden="true">&larr;</span> Previous')|safe, next=('Next <span aria-hidden="true">&rarr;</span>')|safe, align=", **kwargs*)

Parameters

- **pagination** – `Pagination` instance.
- **fragment** – Add URL fragment into link, such as `#comment`.
- **prev** – Symbol/text to use for the "previous page" button.
- **next** – Symbol/text to use for the "next page" button.
- **align** – Can be 'left', 'center' or 'right', default to 'left'.
- **kwargs** – Additional arguments passed to `url_for`.

1.2.8 render_pagination()

Render a standard pagination for query pagination object created by Flask-SQLAlchemy.

Example

```
{% from 'bootstrap/pagination.html' import render_pagination %}

{{ render_pagination(pagination) }}
```

API

render_pagination(*pagination, endpoint=None, prev='«', next='»', ellipses='…', size=None, args={}, fragment=", align=", **kwargs*)

Parameters

- **pagination** – `Pagination` instance.

- **endpoint** – Which endpoint to call when a page number is clicked. `url_for()` will be called with the given endpoint and a single parameter, `page`. If `None`, uses the requests current endpoint.
- **prev** – Symbol/text to use for the "previous page" button. If `None`, the button will be hidden.
- **next** – Symbol/text to use for the "next page" button. If `None`, the button will be hidden.
- **ellipses** – Symbol/text to use to indicate that pages have been skipped. If `None`, no indicator will be printed.
- **size** – Can be 'sm' or 'lg' for smaller/larger pagination.
- **args** – Additional arguments passed to `url_for()`. If `endpoint` is `None`, uses `args` and `view_args`
- **fragment** – Add URL fragment into link, such as `#comment`.
- **align** – The align of the pagination. Can be 'left', 'center' or 'right', default to 'left'.
- **kwargs** – Extra attributes for the `<ul>`-element.

1.2.9 render_static()

Render a resource reference code (i.e. `<link>`, `<script>`).

Example

```
{% from 'bootstrap/utils.html' import render_static %}

{{ render_static('css', 'style.css') }}
```

API

render_static(*type*, *filename_or_url*, *local=True*)

Parameters

- **type** – Resources type, one of `css`, `js`, `icon`.
- **filename_or_url** – The name of the file, or the full URL when `local` set to `False`.
- **local** – Load local resources or from the passed URL.

1.2.10 render_messages()

Render Bootstrap alerts for flash messages send by `flask.flash()`.

Example

Flash the message in your view function with `flash(message, category)`:

```
from flask import flash

@app.route('/test')
def test():
    flash('a info message', 'info')
    flash('a danger message', 'danger')
    return your_template
```

Render the messages in your base template (normally below the navbar):

```
{% from 'bootstrap/utils.html' import render_messages %}

<nav>...</nav>
{{ render_messages() }}
<main>...</main>
```

API

`render_messages`(*messages=None, container=False, transform={...}, default_category=config.BOOTSTRAP_MSG_CATEGORY, dismissible=False, dismiss_animate=False*)

Parameters

- **`messages`** – The messages to show. If not given, default to get from `flask.get_flashed_messages(with_categories=True)`.
- **`container`** – If true, will output a complete `<div class="container">` element, otherwise just the messages each wrapped in a `<div>`.
- **`transform`** – A dictionary of mappings for categories. Will be looked up case-insensitively. Default maps all Python loglevel names to Bootstrap CSS classes.
- **`default_category`** – If a category does not has a mapping in transform, it is passed through unchanged. `default_category` will be used when `category` is empty.
- **`dismissible`** – If true, will output a button to close an alert. For fully functioning dismissible alerts, you must use the alerts JavaScript plugin.
- **`dismiss_animate`** – If true, will enable dismiss animate when click the dismiss button.

When you call `flash('message', 'category')`, there are 8 category options available, mapping to Bootstrap 4's alerts type:

primary, secondary, success, danger, warning, info, light, dark.

If you want to use HTML in your message body, just wrapper your message string with `flask.Markup` to tell Jinja it's safe:

```
from flask import flash, Markup

@app.route('/test')
def test():
    flash(Markup('a info message with a link: <a href="/">Click me!</a>'), 'info')
    return your_template
```

1.2.11 render_table()

Render a Bootstrap table with given data.

Example

```
@app.route('/test')
def test():
    data = Message.query.all()
    return render_template('test.html', data=data)
```

```
{% from 'bootstrap/table.html' import render_table %}

{{ render_table(data) }}
```

API

render_table(*data*, *titles=None*, *primary_key='id'*, *primary_key_title='#'*, *caption=None*, *table_classes=None*, *header_classes=None*, *responsive=False*, *responsive_class='table-responsive'*, *show_actions=False*, *actions_title='Actions'*, *model=None*, *custom_actions=None*, *view_url=None*, *edit_url=None*, *delete_url=None*, *new_url=None*)

Parameters

- **data** – An iterable of data objects to render. Can be dicts or class objects.
- **titles** – An iterable of tuples of the format (prop, label) e.g `[('id', '#')]`, if not provided, will automatically detect on provided data, currently only support SQLAlchemy object.
- **primary_key** – Primary key identifier for a single row, default to `id`.
- **primary_key_title** – Primary key title for a single row, default to `#`.
- **caption** – A caption to attach to the table.
- **table_classes** – A string of classes to apply to the table (e.g `'table-small table-dark'`).
- **header_classes** – A string of classes to apply to the table header (e.g `'thead-dark'`).
- **responsive** – Whether to enable/disable table responsiveness.
- **responsive_class** – The responsive class to apply to the table. Default is `'table-responsive'`.
- **show_actions** – Whether to display the actions column. Default is `False`.
- **model** – The model used to build custom_action, view, edit, delete URLs.
- **actions_title** – Title for the actions column header. Default is `'Actions'`.
- **custom_actions** – A list of tuples for creating custom action buttons, where each tuple contains ('Title Text displayed on hover', 'bootstrap icon name', 'URL tuple') (e.g. `[('Run', 'play-fill', ('run_report', [('report_id', ':id')]))]`).

- **view_url** – URL string or URL tuple in `('endpoint', [('url_parameter_name', ':db_model_fieldname')])` to use for the view action.
- **edit_url** – URL string or URL tuple in `('endpoint', [('url_parameter_name', ':db_model_fieldname')])` to use for the edit action.
- **delete_url** – URL string or URL tuple in `('endpoint', [('url_parameter_name', ':db_model_fieldname')])` to use for the delete action.
- **new_url** – URL string or endpoint to use for the create action (new in version 1.6.0).

To set the URLs for table actions, you will need to pass an URL tuple in the form of `('endpoint', [('url_parameter_name', ':db_model_fieldname')])`:

- `endpoint`: endpoint of the view, normally the name of the view function
- `[('url_parameter_name', ':db_model_fieldname')]`: a list of two-element tuples, the tuple should contain the URL parameter name and the corresponding field name in the database model (starts with a `:` mark to indicate it's a variable, otherwise it will becomes a fixed value). *db_model_fieldname*‘ may also contain dots to access relationships and their fields (e.g. `user.name`).

Remember to set the `model` when setting this URLs, so that Bootstrap-Flask will know where to get the actual value when building the URL.

For example, for the view below:

```
class Message(Model):
    id = Column(primary_key=True)

@app.route('/messages/<int:message_id>')
def view_message(message_id):
    pass
```

To pass the URL point to this view for `view_url`, the value will be: `view_url=('view_message', [('message_id', ':id')])`. Here is the full example:

```
@app.route('/test')
def test():
    data = Message.query.all()
    return render_template('test.html', data=data, Message=Message)
```

```
{% from 'bootstrap/table.html' import render_table %}

{{ render_table(data, model=Message, view_url=('view_message', [('message_id', ':id
↪')])) }}
```

The following arguments are expect to accpet an URL tuple:

- `custom_actions`
- `view_url`
- `edit_url`
- `delete_url`

You can also pass a fiexd URL string, but use a primary key placeholder in the URL is deprecated and will be removed in version 2.0.

When setting the `delete_url`, you will also need to enable the CSRFProtect extension provided by Flask-WTF, so that the CSRF protection can be added to the delete button:

```
$ pip install flask-wtf
```

```
from flask_wtf import CSRFProtect

csrf = CSRFProtect(app)
```

By default, it will enable the CSRF token check for all the POST requests, read more about this extension in its documentation.

1.2.12 render_icon()

Render a Bootstrap icon.

Example

```
{% from 'bootstrap/utils.html' import render_icon %}

{{ render_icon('heart') }}
```

API

`render_icon``(`*name*, *size=config.BOOTSTRAP_ICON_SIZE*, *color=config.BOOTSTRAP_ICON_COLOR*`)`

Parameters

- **`name`** – The name of icon, you can find all available names at Bootstrap Icon.
- **`size`** – The size of icon, you can pass any vaild size value (e.g. `32`/`'32px'`, `1.5em`, etc.), default to use configuration `BOOTSTRAP_ICON_SIZE` (default value is *'1em'*).
- **`color`** – The color of icon, follow the context with `currentColor` if not set. Accept values are Bootstrap style name (one of `['primary', 'secondary', 'success', 'danger', 'warning', 'info', 'light', 'dark', 'muted']`) or any valid color string (e.g. `'red'`, `'#ddd'` or `'(250, 250, 250)'`), default to use configuration `BOOTSTRAP_ICON_COLOR` (default value is `None`).

1.3 Migrate from Flask-Bootstrap

If you come from Flask-Bootstrap, looking for an alternative that supports Bootstrap 4, well, then you are in the right place.

Bootstrap-Flask originated as a fork of Flask-Bootstrap, but some APIs were changed, deleted and improved, some bugs were fixed, and on top of all that, some new macros were added. This tutorial will go through all the steps to migrate from Flask-Bootstrap.

1.3.1 Uninstall and Install

Flask-Bootstrap and Bootstrap-Flask can't live together, so you have to uninstall Flask-Bootstrap first and then install Bootstrap-Flask:

```
$ pip uninstall flask-bootstrap
$ pip install bootstrap-flask
```

if you accidentally installed both of them, you will need to uninstall them both first:

```
$ pip uninstall flask-bootstrap bootstrap-flask
$ pip install bootstrap-flask
```

If you want to use both Flask-Bootstrap and Bootstrap-Flask for different projects, you can use virtual environment.

1.3.2 Initialize the Extension

The initialization of this extension is the same as with Flask-Bootstrap. The package's name is still `flask_bootstrap`, in order to follow the rule of Flask extension development and easy the pain of migration.

```
from flask_bootstrap import Bootstrap
from flask import Flask

app = Flask(__name__)

bootstrap = Bootstrap(app)
```

1.3.3 Create Base Template

In Flask-Bootstrap, there is a built-in base template called `bootstrap/base.html`. This extension does not provide one. You have to create it by yourself; an example starter is given here:

```
<!doctype html>
<html lang="en">
    <head>
        {% block head %}
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
↪fit=no">

        {% block styles %}
            <!-- Bootstrap CSS -->
            {{ bootstrap.load_css() }}
        {% endblock %}

        <title>Your page title</title>
        {% endblock %}
    </head>
    <body>
        <!-- Your page content -->
        {% block content %}{% endblock %}

        {% block scripts %}
```

(continues on next page)

(continued from previous page)

```
            <!-- Optional JavaScript -->
            {{ bootstrap.load_js() }}
        {% endblock %}
    </body>
</html>
```

Just create a file called `base.html` inside your `templates` folder, copy the contents above into it. There are two resource helper methods used in the example template above (i.e. `bootstrap.load_css()` and `bootstrap.load_js()`). They will generate `<href></href>` and `<script></script>` codes to include Bootstrap's CSS and JavaScript files. These default to load the resources from CDN (provided by jsDelivr). If you set the configuration variable `BOOTSTRAP_SERVE_LOCAL` to `True` the local resources inside the package folder will be used instead.

It's optional to use these resources methods, you can write the codes by yourself to load Bootstrap resources in your application's static folder, or from a different CDN provider that you want to use.

1.3.4 Change Template and Macro Name

The template `bootstrap/wtf.html` changed to `bootstrap/form.html`, some macro's name was changed too:

For example, you will need to change the import statement:

```
{% from 'bootstrap/wtf.html' import quick_form, form_field %}
```

to:

```
{% from 'bootstrap/form.html' import render_form, render_field %}
```

The macros below were removed (or not supported yet):

- ie8()
- icon()
- form_button()
- analytics()
- uanalytics()

There are also some new macros were introduced, check them out at *Macros* section.

1.4 Advanced Usage

1.4.1 Form Button Customization

Button Style

When you use form related macros, you have a couple ways to style buttons. Before we start to dive into the solutions, let's review some Bootstrap basics: In Bootstrap 4, you have 9 normal button style and 8 outline button style, so you have 17 button style classes below:

- btn-primary
- btn-secondary

- btn-success
- btn-danger
- btn-warning
- btn-info
- btn-light
- btn-dark
- btn-link
- btn-outline-primary
- btn-outline-secondary
- btn-outline-success
- btn-outline-danger
- btn-outline-warning
- btn-outline-info
- btn-outline-light
- btn-outline-dark

Remove the `btn-` prefix, you will get what we (actually, I) called “Bootstrap button style name”:

- primary
- secondary
- success
- danger
- warning
- info
- light
- dark
- link
- outline-primary
- outline-secondary
- outline-success
- outline-danger
- outline-warning
- outline-info
- outline-light
- outline-dark

You will use these names in Bootstrap-Flask. First, you configuration variables `BOOTSTRAP_BTN_STYLE` to set a global form button style:

```
from flask import Flask
from flask_bootstrap import Bootstrap

app = Flask(__name__)
bootstrap = Bootstrap(app)

app.config['BOOTSTRAP_BTN_STYLE'] = 'primary'  # default to 'secondary'
```

Or you can use `button_style` parameter when using `render_form`, `render_field` and `render_form_row`, this parameter will overwrite `BOOTSTRAP_BTN_STYLE`:

```
{% from 'bootstrap/form.html' import render_form %}

{{ render_form(form, button_style='success') }}
```

Similarly, you can use this way to control the button size. In Bootstrap 4, buttons can have 4 sizes:

- btn-sm
- btn-md (the default size)
- btn-lg
- btn-block

So, the size names used in Bootstrap-Flask will be:

- sm
- md (the default size)
- lg
- block

Now you can use a configuration variable called `BOOTSTRAP_BTN_STYLE` to set global form button size:

```
from flask import Flask
from flask_bootstrap import Bootstrap

app = Flask(__name__)
bootstrap = Bootstrap(app)

app.config['BOOTSTRAP_BTN_SIZE'] = 'sm'  # default to 'md'
```

there also a parameter called `button_size` in form related macros (it will overwrite `BOOTSTRAP_BTN_SIZE`):

```
{% from 'bootstrap/form.html' import render_form %}

{{ render_form(form, button_size='lg') }}
```

if you need a **block level small** button (`btn btn-sm btn-block`), you can just do something hacky like this:

```
app.config['BOOTSTRAP_BTN_SIZE'] = 'sm btn-block'
```

What if I have three buttons in one form, and I want they have different styles and sizes? The answer is `button_map` parameter in form related macros. `button_map` is a dictionary that mapping button field name to Bootstrap button style names. For example, `{'submit':  'success'}`. Here is a more complicate example:

```
{% from 'bootstrap/form.html' import render_form %}

{{ render_form(form, button_map={'submit': 'success', 'cancel': 'secondary', 'delete
↪': 'danger'}) }}
```

It will overwrite `button_style` and `BOOTSTRAP_BTN_STYLE`.

1.4.2 Bootswatch Themes

Bootswatch is a collection of free and open source themes for Bootstrap. If you are using `bootstrap.load_css()` to include Bootstrap resources. Then you can set Bootswatch theme with configuration variable `BOOTSTRAP_BOOTSWATCH_THEME`.

The available theme names are: 'cerulean', 'cosmo', 'cyborg', 'darkly', 'flatly', 'journal', 'litera', 'lumen', 'lux', 'materia', 'minty', 'pulse', 'sandstone', 'simplex', 'sketchy', 'slate', 'solar', 'spacelab', 'superhero', 'united', 'yeti'.

Here is an example to use `lumen` theme:

```
app.config['BOOTSTRAP_BOOTSWATCH_THEME'] = 'lumen'
```

You can find these themes on https://bootswatch.com.

1.5 Run the demo application

Type these commands in the terminal:

```
$ git clone https://github.com/greyli/bootstrap-flask.git
$ cd bootstrap-flask/examples
$ pip install -r requirements.txt
$ flask run
```

Now go to http://localhost:5000.

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Reference

class flask_bootstrap.**Bootstrap**(*app=None*)

init_app(*app*)

static load_css(*version='4.3.1'*, *bootstrap_sri=None*)
Load Bootstrap's css resources with given version.

New in version 0.1.0.

Parameters **version** – The version of Bootstrap.

static load_js(*version='4.3.1'*, *jquery_version='3.4.1'*, *popper_version='1.14.0'*, *with_jquery=True*, *with_popper=True*, *bootstrap_sri=None*, *jquery_sri=None*, *popper_sri=None*)
Load Bootstrap and related library's js resources with given version.

New in version 0.1.0.

Parameters

- **version** – The version of Bootstrap.
- **jquery_version** – The version of jQuery.
- **popper_version** – The version of Popper.js.
- **with_jquery** – Include jQuery or not.
- **with_popper** – Include Popper.js or not.

CHAPTER 3

Changelog

3.1 Changelog

3.1.1 2.0.0

Release date: -

- Drop Python 2 and 3.5 support.

3.1.2 1.8.0

Release date: 2021/9/5

- Fix bootswatch theme bug: remove theme name `'default'` (#141).
- Add configuration `BOOTSTRAP_TABLE_VIEW_TITLE`, `BOOTSTRAP_TABLE_EDIT_TITLE`, `BOOTSTRAP_TABLE_DELETE_TITLE`, `BOOTSTRAP_TABLE_NEW_TITLE` to support changing the icon title of table actions.
- Introduce a new and better way to pass table action URLs to support the usage of `Flask`'s path converters (#146, #151).
- Deprecate `action_pk_placeholder` and placeholder action URLs in `render_table`.
- Support SRI for JS/CSS resources (#142).

3.1.3 1.7.0

Release date: 2021/6/10

- Add a `custom_actions` parameter for the `render_table` macro. When passing a list of tuples `[(title, bootstrap icon, link)]` to the `custom_actions` parameter, the `render_table` macro will create an icon (link) on the action column for each tuple in the list. The title text (first index of each tuple) will show when hovering over each `custom_actions` button (#134).

- Update Bootstrap Icons to v1.5.0.
- Improve action icons for `render_table`, now the icons can be styled with the `action-icon` CSS class (#137).
- Change the default `action_pk_placeholder` to `':id'`. The support to the old value will be removed in version 2.0 (#138).

3.1.4 1.6.0

Release date: 2021/5/29

- Add a `new_url` parameter for the `render_table` macro. When passing an URL to the `new_url` parameter, the `render_table` macro will create an icon (link) on the action header (#133).
- Fix the display of the delete icon for `render_table` macro (#132).

3.1.5 1.5.3

Release date: 2021/5/18

- Fix class for horizontal form label (#131).
- Fix hidden field label issue for `render_field` macro (#130).
- Refactor tests (#125).

3.1.6 1.5.2

Release date: 2021/4/13

- Fix *render_table* macro for SQLAlchemy >= 1.4 (#124).

3.1.7 1.5.1

Release date: 2020/11/9

- Fix missing end angle bracket for bootswatch CSS link tag (#110).
- Migrate tests to pytest (#109).

3.1.8 1.5

Release date: 2020/8/30

- Fix `tox` broken environments.
- Fix `ResourceWarning` in `test_local_resources` (#78).
- Fix `IndexError` when using `render_table` with empty data (#75).
- Add support for actions column in `render_table` macro (#76).
- Add support for Bootswatch theme via configuration `BOOTSTRAP_BOOTSWATCH_THEME` (#88).
- Fix checkbox render issue: add `for` attribute to link `<label>` with checkbox, only add `is-invalid` class when there are errors.

- Change default button style class from `btn-secondary` to `btn-primary` (#62).
- Deprecated `form_errors` macro and it will be removed in 2.0, add `render_hidden_errors` macro as replacement.
- Add `render_icon` macro to render Bootstrap icon with Bootstrap Icon SVG Sprite (#99).
- Add configuration `BOOTSTRAP_MSG_CATEGORY` to set default message category.

3.1.9 1.4

Release date: 2020/6/15

- Add `render_table` macro to render a Bootstrap table (#71).

3.1.10 1.3.2

Release date: 2020/5/30

- Support display error message for `RadioField` and `BooleanField`, display description for `RadioField`.

3.1.11 1.3.1

Release date: 2020/4/29

- Fix add `field.render_kw.class` to form label class attribute.
- Fix append extra space in class attribute when no `field.render_kw.class` presents (#63).

3.1.12 1.3.0

Release date: 2020/4/23

- Fix `enctype` attribute setting for WTForms `MultipleFileField` (Flask-Bootstrap #198).
- Fix WTForms field class append bug when using `render_kw={'class': 'my-class'}` (#53).
- Fix WTForms field description not showing for `BooleanField` (Flask-Bootstrap #197).
- Add configuration variable `` BOOTSTRAP_BTN_STYLE``(default to ``primary) `` and `` BOOTSTRAP_BTN_SIZE``(default to ``md) `` to set default form button style and size globally.
- Add parameter `button_style` and `button_map` for `render_form` and `render_field` to set button style and size.

3.1.13 1.2.0

Release date: 2019/12/5

- Add macro `render_messages` for rendering flashed messages.
- Fix rendering bug for WTForms `FormField` (#34).

3.1.14 1.1.0

Release date: 2019/9/9

- Update Bootstrap version to 4.3.1

3.1.15 1.0.10

Release date: 2019/3/7

- Added macro `render_form_row` for rendering a row of a bootstrap grid form.

3.1.16 1.0.9

Release date: 2018/11/14

- Fix missing error message when form type was horizontal.
- Fix missing input label for RadioField.
- Fix RadioField grid when form type was horizontal.

3.1.17 1.0.8

Release date: 2018/9/6

- Correct macro name used in `templates/bootstrap/form.html`: `form_field` -> `render_field`.

3.1.18 1.0.7

Release date: 2018/8/30

- Built-in resources loading not based on‘‘FLASK_ENV‘‘.

3.1.19 1.0.6

Release date: 2018/8/7

- Fix unmatched built-in jQuery filename. (#8)

3.1.20 1.0.5

Release date: 2018/8/7

- Fix KeyError Exception if ENV isn’t defined. (#7)

3.1.21 1.0.4

Release date: 2018/7/24

- Add missing `<script>` tag in resources URL. (#3)

3.1.22 1.0.3

Release date: 2018/7/22

- Built-in resources will be used when `FLASK_ENV` set to `development`.
- Change CDN provider to jsDelivr.

3.1.23 1.0.2

Release date: 2018/7/21

- Include `popper.js` before `bootstrap.js` in `bootstrap.load_js()`. (#2)

3.1.24 1.0.1

Release date: 2018/7/1

- Fix local resources path error
- Add basic unit tests

3.1.25 1.0

Release date: 2018/6/11

Initial release.

CHAPTER 4

Development

We welcome all kinds of contributions. You can build the development environment locally with the following commands:

```
$ git clone git@github.com:greyli/bootstrap-flask.git
$ cd bootstrap-flask
$ python3 -m venv venv
$ . venv/bin/activate
$ pip install ".[dev]"
```

Then run tests with tox:

```
$ tox
```

CHAPTER 5

Authors

Maintainer: Grey Li

See also the list of contributors who participated in this project.

CHAPTER 6

License

This project is licensed under the MIT License (see the `LICENSE` file for details).

Some macros were part of Flask-Bootstrap and were modified under the terms of its BSD License.

Python Module Index

Index