Bootstrap-Flask

Release 1.0.4
Grey Li

该文档由吴依杰根据Bootstrap-Flask的官方文档整理, 文档所有权归原作者所有, V1.0

用于Flask/Jinja2的Bootstrap4帮手. 基于Flask-Bootstrap, 但更轻量, 更好用

基本用法

安装

```
pip install bootstrap-flask
```

初始化

```
from flask_bootstrap import Bootstrap
from flask import Flask

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

资源助手

Bootstrap-Flask通过提供两个函数在模版中加载Bootstrap资源:

- bootstrap. load_css()
- bootstrap.load_js()

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

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

你可以通过version来锁定你想要使用的Bootstrap4版本. 默认从CDN加载文件, 如果你想从本地加载文件, 可以将`BOOTSTRAP_SERVE_LOCAL`设置为`True`来加载本地文件. 但是，建议您自己管理Bootstrap资源.

起动模板

出于灵活性的原因, Bootstrap-Flask不包含内置的基本模板(这在将来可能会改变). 现在, 您必须自己创建它. 一定要使用HTML5 doctype, 并包含一个viewport meta标签用于正确的响应行为. 下面是一个基本模板的例子:

```
<!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>
```

在你的模板文件夹中使用它(命名为base.html或layout.html等), 并在子模板中继承它, 可以在模版继承http://flask.pocoo.org/docs/1.0/patterns/templateinheritance/中查看详细内容.

Macros(宏)

宏[macros]	模版路径[Templates Path]	描述[Description]
`render_field()`	`bootstrap/form.html`	渲染WTForms表单字段
`render_form()`	`bootstrap/form.html`	渲染WTForms表单
`render_pager()`	`boot-strap/pagination.html`	渲染一个基本的Flask-SQLAlchemy分页

`render_pagination()`	`bootstrap/pagination.html`	渲染一个标准的Flask-SQLAlchemy分页
`render_nav_item()`	`bootstrap/nav.html`	渲染一个导航项
`render_breadcrumb_item()`	`bootstrap/nav.html`	渲染一个面包屑项目
`render_static()`	`bootstrap/utils.html`	渲染资源引用代码(例如<script>)
`render_messages()`	`bootstrap/utils.html`	渲染flash()函数发送的闪现消息

如何使用这些宏? 很简单, 只需从相应的路径导入它们, 然后像调用其他宏一样调用它们

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

使用宏

这些宏将帮助您快速、轻松地生成Bootstrap-markup代码

`render_nav_item()`

渲染一个BootStrap导航栏item

```
{% 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)`

功能:渲染一个BootStrap导航栏item
参数:

- endpoint:用于生成URL的端点
- text:将显示在item上的文本
- use_li:默认生成`<a></a>`, 当值为True时, 生成`<li><a></a></li>`
- kwargs:将关键字参数传递给url_for()函数

`render_breadcrumb_item()`

渲染一个面包屑项目

```
{% 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)`

功能:渲染一个面包屑项目
参数:

- endpoint:用于生成URL的端点
- text:将显示在item上的文本
- kwargs:将关键字参数传递给url_for()函数

`render_field()`

渲染一个由Flask-WTF/WTForms创建的表单字段

例子:

```
{% 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_map={})`

功能:渲染一个由Flask-WTF/WTForms创建的表单字段
参数:

- `field`:要渲染的表单字段(属性)
- `form_type`:`basic`, `inline` , `horizontal`其中之一, 详情参见Bootstrap文档
- `horizontal_columns`:当使用水平布局时, 布局表单如下所示. 必须是一个3元的(column-type, left-column-size, right-column-size)[柱型, 左柱型, 右柱型]
- `button_map`:一个字典, 映射button字段名到Bootstrap category 名, 如:`primary` `danger` or `success`, 在button_map中没有找到的button将使用`secondary`类型的button

render_form()

渲染一个由Flask-WTF/WTForms创建的表单对象

例子

```
{% 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", hor-izontal_columns=('lg', 2, 10), enctype=None, button_map={}, id="", novalidate=False, ren-der_kw={})`

功能:渲染一个由Flask-WTF/WTForms创建的表单对象
参数:

- `form`:要输出的表单
- `action`:接收表单数据的URL
- `method`:表单属性方法
- `extra_classes`:给表单添加的class
- `role`:表单role属性
- `form_type`:`basic`, `inline` or `horizontal`, 详见BootStrap文档
- `horizontal_columns`:当使用水平布局时, 布局表单如下所示. 必须是一个3元的(column-type, left-column-size, right-column-size)[柱型, 左柱型, 右柱型]
- `enctype`:表单enctype属性,如果为None, 将自动设置为`multipart/form-data`(如果表单中有一个`FileField`)
- `button_map`:一个字典, 映射button字段名到Bootstrap category 名, 如:`primary` `danger` or `success`, 例如:`{'submit': 'success'}`,在button_map中没有找到的button将使用`default`类型的button.
- `id`:表单id属性
- `novalidate`:标志, 它决定是否在表单中添加`novalidate`类
- `render_kw`:一个字典, 为表单标记指定自定义属性

`form_errors(form, hiddens=True)`

功能:呈现包含表单错误消息的段落. 这通常只用于输出隐藏字段表单错误, 因为其他错误都附加在表单字段上
参数:

- `form`:要被呈现错误的表单
- `hiddens`:如果为True, 也呈现隐藏字段的错误. 如果'only', 只渲染这些

`render_form_row()`

渲染网格表单的一行

例子

```
{% 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]) }}
</form>
```

API

`render_form_row(fields, row_class='form-row', col_class_default='col', col_map={})`

功能:渲染具有给定字段的引导行
属性:

- `fields`:要在一行中渲染的可迭代字段
- `row_class`:应用于表示行的div中的class, 如form-row或row
- `col_class_default`:如果没有为渲染字段的div列指定更具体的内容, 则应用于表示列的div的默认class
- `col_map`:一个字典, 将field.name映射到一个类定义, 这个类定义应该被映射到包含该字段的div列, 例如`col_map={'username': 'col-md-2'})`

render_pager()

渲染由Flask-SQLAlchemy创建的分页对象

例子

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

API

`render_pager(pagination, fragment='', prev=('<span aria-hidden="true">&larr;</span> Previ-ous')|safe, next=('Next <span aria-hidden="true">&rarr;</span>')|safe, align='',**kwargs)`
功能:渲染用于查询分页的简单pagination
参数:

- `pagination`:Pagination实例
- `fragment`:将url片段添加到链接中, 例如:`#comment`
- `prev`:用于"上一页"按钮的符号/文本
- `next`:用于"下一页"按钮的符号/文本
- `align`:可以是`left`, `center` or `right`, 默认是`left`
- `kwargs`:传递给url_for的其他参数

render_pagination()

渲染由Flask-SQLAlchemy创建的分页对象

例子

```
{% 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)`
功能:
参数:

- `pagination`:Pagination实例
- `endpoint`:单击页码时调用哪个端点. 将使用给定的端点和单个参数page调用url_for(). 如果没有, 则使用请求当前端点
- `prev`:用于"上一页"按钮的符号/文本. 如果没有, 按钮将被隐藏
- `next`:用于"下一页"按钮的符号/文本. 如果没有, 按钮将被隐藏
- `ellipses`:用来表示跳过页面的符号/文本. 如果没有,没有指示将被打印出来
- `size`:用来控制大小, 可以为: `sm`, `lg`
- `args`:给`url_for()`函数传递参数, 如果`endpoint=None`, 使用`args`和`view_args`
- `fragment`:将url片段添加到链接中, 例如:`#comment`
- `align`:可以是`left`, `center` or `right`, 默认是`left`
- `kwargs`:`<ul>`元素的额外属性

render_static()

渲染一个资源引用代码(例如:`<link>`, `<script>`)

例子

```
{% from 'bootstrap/utils.html' import render_static %}
{{ render_static('css', 'style.css') }}
```

API

`render_static(type, filename_or_url, local=True)`
功能:渲染一个资源引用代码(例如:`<link>`, `<script>`)
参数:

- `type`:资源类型, `css`, `js`, `icon`之一
- `filename_or_url`:文件的名称, 或local=False时的完整url
- `local`:加载本地资源传递的URL

render_messages()

渲染flask.flash()发送的消息

例子

用`flash(message, category)`在视图函数中闪现消息

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

在`base template`中呈现消息(通常在导航条下面)

```
{% from 'bootstrap/utils.html' import render_messages %}
<nav>...</nav>
{{ render_messages() }}
<main>...</main>
```

API

`render_messages(messages=None, container=False, transform={...},`

`default_category='primary', dis- missible=False, dismiss_animate=False)`
功能:通过`flask.flash()`渲染Bootstrap alerts信息
参数:

- `messages`:要显示的消息. 如果没有给出,默认从`flask.get_flashed_messages(with_categories=True)`获取
- `container`:如果为True, 将输出一个完整的`<div class="container">`元素, 否则，每个消息都包装在`<div>`中
- `transform`:类别映射字典。将被不加区分地查找。默认情况下，将所有Python日志级别的名称映射到引导CSS类
- `default_category`:如果一个类别在转换中没有映射, 那么它将以不变的方式传递. 当`category`为空时, 将使用`default_category`
- `dismissible`:如果为True, 则输出一个按钮来关闭alter, 为了充分发挥dismissible alerts功能, 您必须使用alter JavaScript插件
- `dismiss_animate`:如果为True,则在单击'解散'按钮时启用'解散动画'

当你调用`flash('message','category')`时,有8个类别选项可用,映射到Boot-strap4的alter类型:

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

如果您想在消息体中使用HTML, 只需用flask对消息字符串进行变形. 告诉Jinja它是安全的

```
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

```

如果您正在寻找关于特定函数、类或方法的信息, 文档的这一部分是为您准备的.

API Reference

class flask_bootstrap.Bootstrap(app=None)

`init_app(app)`
`static load_css(version='4.3.1')`

功能:按照给定的版本加载Bootstrap的css资源
参数:

- `version`:Bootstrap的版本

```
static load_js(version='4.3.1', jquery_version='3.3.1', popper_version='1.14.0', with_jquery=True, with_popper=True)
```

功能:按照给定的版本加载Bootstrap的js资源
参数:

- `version`:Bootstrap的版本
- `jquery_version`:jQuery的版本
- `popper_version`:Popper.js的版本
- `with_jquery`:是否包含jQuery
- `with_popper`:是否包含Popper.js