Python AdminUI

2022 年 06 月 25 日

目录

使用 Python 制作专业的 Web 前端界面

本 Python 包适用于想要制作简单的前端界面，但却不想处理 HTML, CSS, React, Angular, Webpack 和其他 Javascript 前端代码的情况。你可以直接用 Python 完成网页、表单、图标和仪表盘界面。

一些应用场景：数据项目；运维工具和脚本；简单 IT 系统和管理系统；业余项目和黑客马拉松项目。本项目服务于那些只需要一个简单的界面而不需要过多自定义样式和应付大流量访问的情况。

本项目基于 Flask 和 Ant Design Pro

特性：

- 轻松制作表单和详情页
- 折线图和条状图
- 多级菜单
- 带有分页的数据表
- 适配小屏幕和移动设备
- 不需要使用 HTML, CSS 或者 JS

CHAPTER 1

安装和快速入门

使用 pip 安装包:

```
pip install adminui
```

为您的项目创建一个 python 文件，例如 example.py：

```
from adminui import *

app = AdminApp()

def on_submit(form_data):
    print(form_data)

@app.page('/', 'Form')
def form_page():
    return [
        Form(on_submit = on_submit, content = [
            TextField('Title', required_message="The title is required!"),
            TextArea('Description'),
            FormActions(content = [
                SubmitButton('Submit')
            ])
        ])
```

(下页继续)

(续上页)

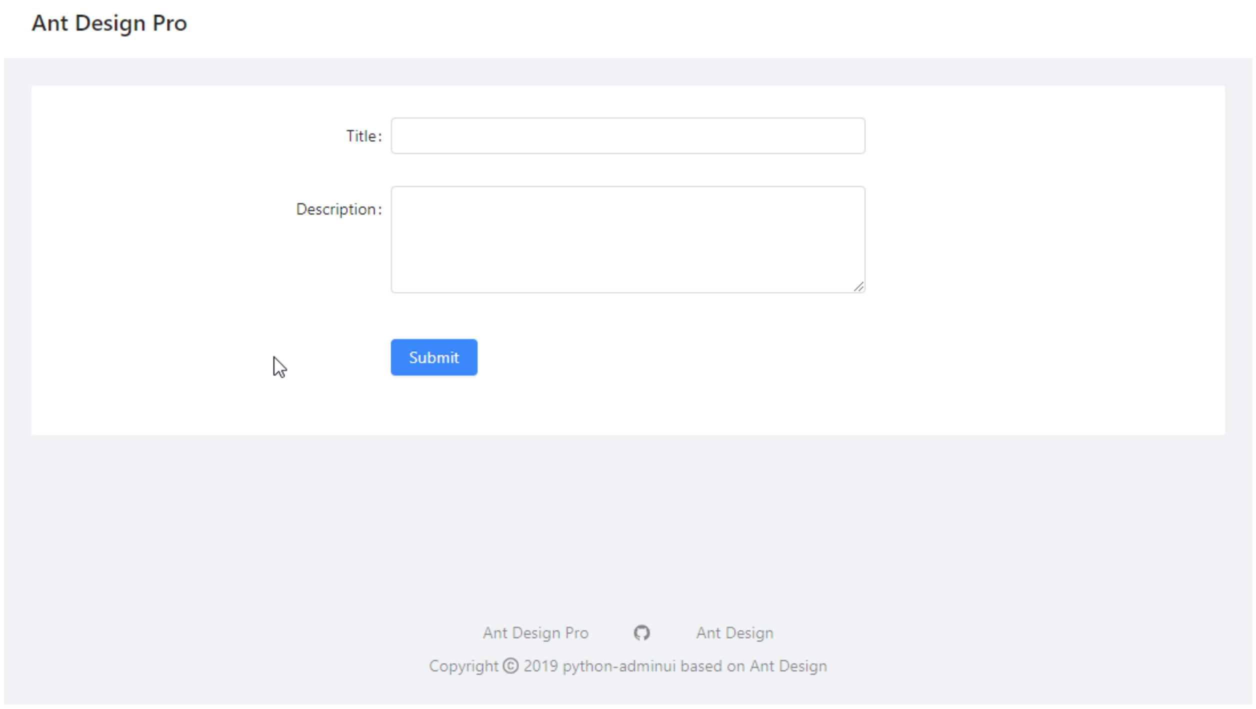
```
    ]

if __name__ == '__main__':
    app.run()
```

运行 python 文件:

```
python example_form.py
```

现在访问 http://127.0.0.1:5000/以查看索引页。它看起来这样:



1.1 基本概念和示例解析

首先，您需要创建一个 AdminApp 对象:

```
app = AdminApp()
```

然后，您将页面添加到你的应用中。使用 `@app.page` 修饰器，后跟一个自定义函数（可以任意命名），该函数返回一个页面元素数组，将在用户访问特定 url 时显示:

```
@app.page('/', 'Form')
def form_page():
```

(下页继续)

(续上页)

```
    return [ ...put page contents here... ]
```

@app.page 修饰器接收两个参数。一个用于 url（本例中为’ /’），另一个用于页面标题（本例中的” Form”）。

通过修饰函数，每次用户访问该页面时，您可以返回不同的页面元素或数据。

所有页面元素都是 Python 对象。您可以参考本文档的其他章节来了解如何使用它们。我们可以看到，上面的示例创建了一个表单、一个文本字段、一个文本区域和一个提交按钮。

最后，运行应用:

```
app.run()
```

1.2 Use FastAPI instead of Flask

Set use_fastapi=True when creating the app; and prepare() instead of run to expose the app to uvicorn.

The basic example will be:

```
from adminui import *

app = AdminApp(use_fastapi=True)

def on_submit(form_data):
    print(form_data)

@app.page('/', 'Form')
def form_page():
    return [
        Form(on_submit = on_submit, content = [
            TextField('Title', required_message="The title is required!"),
            TextArea('Description'),
            FormActions(content = [
                SubmitButton('Submit')
            ])
        ])
    ]

fastapi_app = app.prepare()
```

Then run from command line:

```
uvicorn example_fastapi:fastapi_app
```

CHAPTER 2

创建表单

若要创建表单，插入一个 Form 对象:

```
@app.page('/', 'Form')
def form_page():
    return [
        Form(on_submit = my_function,
             content = [ ... content of the form ... ])
    ]
```

这里可以传递一个函数来处理用户提交的数据。此函数需要有一个参数，是以字典表示的用户填写的表单的值:

```
def my_function(values):
    ... do things with values ...
```

In the content section of the form, you may add TextField and other form controls. See the next section for details

最后，不要忘记添加一个提交按钮。在窗体末尾插入 “FormActions“ 对象，并在其末尾插入 Submit:

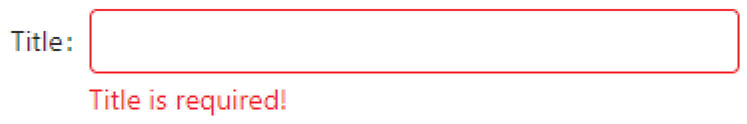
```
FormActions(content = [
    SubmitButton('Submit')
])
```

这样，就完成了一个表单。

2.1 List of Form Controls

Here are a list of controls you may use in your form:

2.1.1 TextField



```
TextField('Title', required_message='Title is required!')
```

Set `password=True` to setup a password field:

```
TextField('Enter Password', password=True)
```

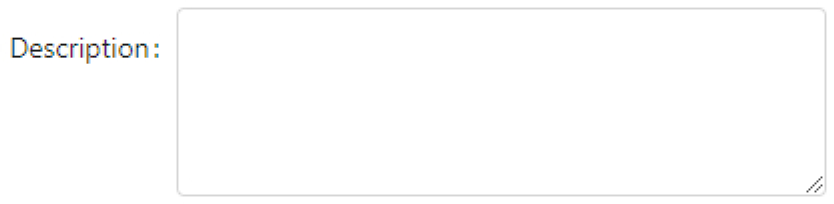
class adminui.TextField(*title*, *name=None*, *required_message=None*, *password=False*, *disabled=False*, *value=None*, *placeholder=None*, *on_change=None*, *id=None*)

在表单中创建文本字段。

参数

- **title** – 字段的标题
- **name** – 提交表单时传递给回调的字典数据的键
- **required_message** – if other than None, the field will be required and an message will be shown when the field is not filled at form submission.
- **placeholder** – the text message shown when the field is not filled.
- **password** – set to True if it’ s a password box
- **disabled** – set to True to make the control disabled

2.1.2 TextArea



```
TextArea('Description')
```

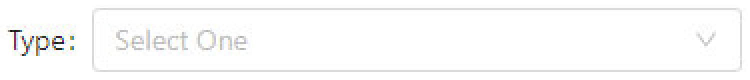
class `adminui.TextArea`(*title*, *name=None*, *required_message=None*, *value=None*, *placeholder=None*, *disabled=False*, *on_change=None*, *id=None*)

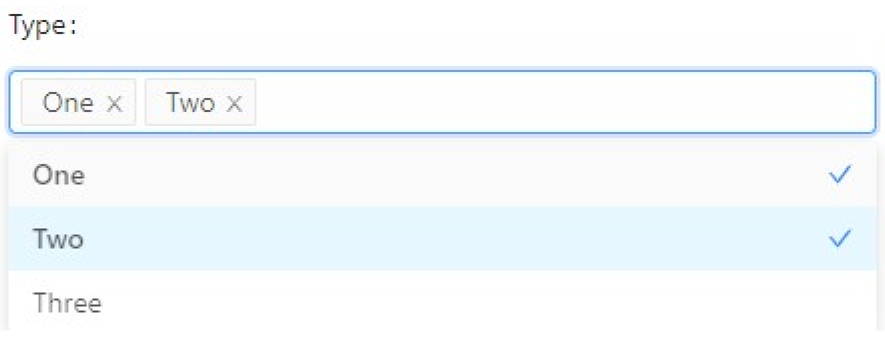
创建文本区域对象

参数

- **title** – 字段的标题
- **name** – 提交表单时传递给回调的字典数据的键
- **required_message** – if other than None, the field will be required and an message will be shown when the field is not filled at form submission.
- **placeholder** – the text message shown when the field is not filled.
- **disabled** – set to True to make the control disabled

2.1.3 Select





```
SelectBox('Type', data=['One', 'Two', 'Three'], placeholder="Select One")
# use multiple=True to allow multiple selecting:
SelectBox('Type', data=['One', 'Two', 'Three'], placeholder="Select Many", multiple=True)
# use tags=True to input tags - users can input arbitrary text as a tag:
SelectBox('Type', data=['One', 'Two', 'Three'], placeholder="Select Many", tags=True)
```

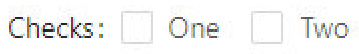
class `adminui.SelectBox`(*title*, *name=None*, *value=None*, *data=[]*, *placeholder=None*, *on_change=None*, *required_message=None*, *multiple=False*, *disabled=False*, *tags=False*, *id=None*)

Create a dropdown box for selecting in a list

参数

- **title** – 字段的标题
- **name** – 提交表单时传递给回调的字典数据的键
- **required_message** – if other than None, the field will be required and an message will be shown when the field is not filled at form submission.
- **data** – the options in the select box. in format of a list of str or a list of [title, value] list e.g. ['one' , 'two' , 'three'] or [['one' , 1], ['two' , 2]], both accepted
- **placeholder** – the text message shown when the field is not filled.
- **disabled** – set to True to make the control disabled

2.1.4 Checkboxes



```
CheckboxGroup('Checks', data=['One', 'Two'])
```

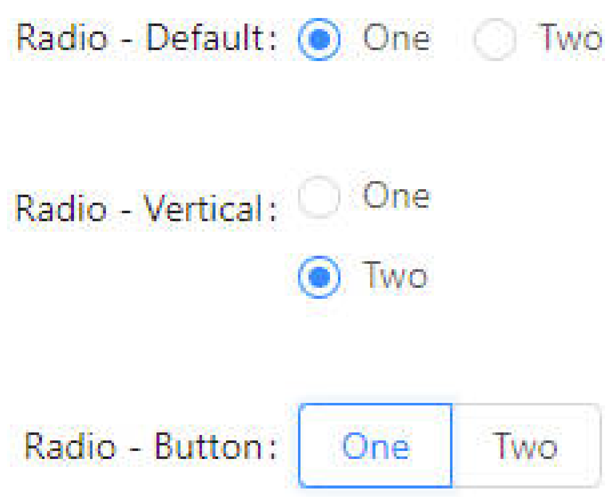
class adminui.CheckboxGroup(*title*, *name=None*, *data=[]*, *value=None*, *disabled=False*, *id=None*, *on_change=None*)

Create a group of checkbox for multi-selecting

参数

- **title** – 字段的标题
- **name** – 提交表单时传递给回调的字典数据的键
- **data** – the title and value of individual checkboxes. in format of a list of str or a list of [title, value] e.g. ['one' , 'two' , 'three'] or [['one' , 1], ['two' , 2]], both accepted disabled: set to True to make the control disabled

2.1.5 Radios



```
RadioGroup('Radio - Default', data=['One', 'Two'], on_change=on_change),
RadioGroup('Radio - Vertical', data=[['One', 1], ['Two', 2]], on_change=on_change,␣
↪format='vertical'),
RadioGroup('Radio - Button', data=[['One', 1], ['Two', 2]], on_change=on_change, format=
↪'button'),
```

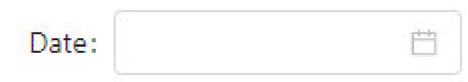
class adminui.RadioGroup(*title*, *name=None*, *data=[]*, *value=None*, *format='default'*, *disabled=False*, *on_change=None*, *id=None*)

Create a group of radio buttons

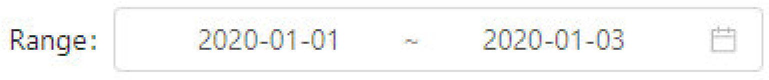
参数

- **title** – 字段的标题
- **name** – 提交表单时传递给回调的字典数据的键
- **value** – the default value of the selected radio
- **data** – the title and value of individual checkboxes. in format of a list of str or a list of [title, value] e.g. ['one' , 'two' , 'three'] or [['one' , 1], ['two' , 2]], both accepted
- **format** – default | button; how the radio box is displayed and arranged
- **disabled** – set to True to make the control disabled

2.1.6 DatePicker



```
DatePicker('Date')
```



```
DatePicker('Range', pick='range')
```

class adminui.DatePicker(*title*, *name=None*, *value=None*, *pick='date'*, *on_change=None*, *id=None*)

Create a date picker to pick a date or date range

参数

- **title** – 字段的标题
- **name** – 提交表单时传递给回调的字典数据的键
- **pick** – ‘date’｜‘month’｜‘week’｜‘range’.

2.2 Callback when form item changes

If you want to do something, say update a part of the page when user select an item in the SelectBox or input text on the TextFields, you may add on_change handlers in your Python code:

```
TextField('Title', required_message='Title is required!', on_change=on_change),

# for the handler:
def on_change(value, values):
    print(value)
    print(values)
```

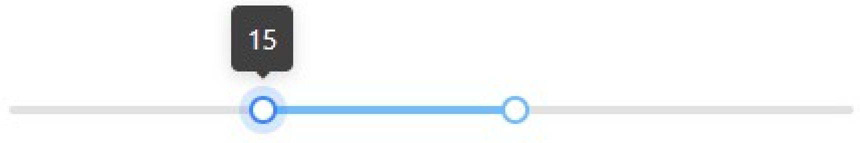
See chapter 页面操作. for details on what can handlers do.

The handler could be a function taking one or two parameters: the first will be the new value of the form item; the second one will be a dictionary of all the values in the form where the form item lives. This will be useful for example when your data shown in the page is filtered by a list of criterions.

2.3 Slider, Switch and other controls

These are not form controls. But they respond to on_change handlers.

2.3.1 Slider



```
Slider(on_change=on_change)
# this will render a slider, with 0~50 range
# and user can pick up a range, with an initial value 20~30
Slider(0, 50, range=True, value=[20,30])
```

2.3.2 Switch

```
Switch(on_change=on_change)
```

2.3.3 Checkbox

```
Checkbox(on_change=on_change)
```

CHAPTER 3

页面操作

如果要在表单提交后将用户重定向到另一个页面，则可以在窗体的 on_submit 回调中返回页面操作：

```
def on_detail():
    return NavigateTo('/detail')
```

AdminUI 支持 NavigateTo 和 Notification 作为页面操作：

class adminui.**NavigateTo**(*url='/'*)

页面操作：将用户导航到其他页面

参数 url – 新页面的 URL

class adminui.**Notification**(*title="*, *text=None*, *type='default'*)

向用户发送右上角通知

参数

- **title** – 通知的标题
- **text** – 通知的文本正文
- **type** – default | success | info | warning | error: type of the notification box

If you want to combine two actions together, return a list. Say you want to notify the user twice:

```
return [
    Notification('A Notification', 'the content of the notification'),
```

(下页继续)

(续上页)

```
    Notification('A Notification', 'the content of the notification'),
]
```

除了表单提交之外，您还可以在其他元素（如按钮被点击时）上使用页面操作。

`class adminui.Button`(*title='Go'*, *style=None*, *icon=None*, *on_click=None*, *link_to=None*, *id=None*)

在页面上创建常规按钮

参数

- **title** – 按钮上显示的标题
- **style** – 设置为”主”按钮（蓝色大按钮）
- **icon** – 按 钮 标 题 前 面 的 图 标。 有 关 字 符 串 值， 请 参 阅 https://ant.design/components/icon/
- **on_click** – 单击按钮时将调用自定义函数

您可以在按钮的 on_click 回调中使用页面操作。

3.1 Update page content in Page Actions

If you wish to change a part of the page in Page Actions, first identify the element with `id` attribute:

```
@app.page('/', 'Control Page')
def control_page():
    return [
        Card(content=[
            Button('Change Content', on_click=on_change_content),
            Button('Change Element', on_click=on_change_self),
        ]),
        Card(id='detail_card'),
        Card('Paragraph Card', [
            Paragraph('This is the original content', id='paragraph')
        ])
    ]
```

Then during a page action, you may return a `ReplaceElement` to replace an element with another:

```
def on_change_self():
    return ReplaceElement('paragraph', Paragraph('This element has been changed'))
```

Thus when users click the button “Change Element” , a new paragraph will replace the old one.

You may also choose to update some attributes of an element. The following code changes the `content` value of “detail_card” element when the users click the “Change Content” button.:

```
return UpdateElement('detail_card', content=[
        DetailGroup('Refund Request', content=[
            DetailItem('Ordre No.', 1100000),
            DetailItem('Status', "Fetched"),
            DetailItem('Shipping No.', 1234567),
            DetailItem('Sub Order', 1135456)
        ]),
    ])
```

3.2 Use a timer

Use timer, to let your Python function run every some seconds. You can also return Page Actions in the timer’s `on_fire` function:

```
@app.page('/', 'Detail Page')
def detail_page():
    return [
        Timer(on_fire=on_timer_fire, data='hello timer'),
        ...
    ]
```

Timers can be set at any position of the page. To remove a timer, replace it with `ReplaceElement` page action.

See example_page_actions.py for the complete example. https://github.com/bigeyex/python-adminui/blob/master/python/example_page_actions.py

CHAPTER 4

Upload Files

File uploader can be a individual component or be a part of a form:

```
@app.page('/', 'form')
def form_page():
    return [
        Card('Upload Form', [
            Upload(on_data=on_upload)  # use Upload individually
        ]),

        Form(on_submit = on_submit, content = [
            Upload(name='upload', on_data=on_upload),     # embed uploads in a form
            FormActions(content = [
                SubmitButton('Submit')
            ])
        ])
    ]
```

The `on_data` handler will be called if a file is uploaded:

```
def on_upload(file):
    print("=== file name will be: ===")
    print(file['file_name'])
    print('=== the file is stored in: ===')
```

(下页继续)

(续上页)

```
    print(app.uploaded_file_location(file))
```

Use `app.uploaded_file_location(file)` to get the absolute path of the uploaded file.

When you gave a `name` to the Upload Component, you can also retrive the file when the form is submitted:

```
def on_submit(form_data):
    print(app.uploaded_file_location(form_data['upload'])) # e.g. the upload component's␣
↪name is 'upload'
```

By default, the file will be stored in /upload of the folder containing the starter Python file, but you can change it by passing an `upload_folder` when creating the AdminUI App:

```
app = AdminApp(upload_folder='Your custom folder')
```

CHAPTER 5

创建菜单

要为项目创建菜单，使用 `AdminApp` 对象的 `set_menu` 函数:

```
app.set_menu([ ... a list of menu items ...])
```

the `set_menu` method takes an array of `MenuItem` objects:

class adminui.MenuItem(*name*, *url=''*, *icon=None*, *auth_needed=None*, *children=[]*)

表示菜单项

参数

- **name** (*str*) – 菜单的标题
- **url** (*str, optional*) – 菜单将导航到的 URL。默认值为””。
- **icon** (*str, optional*) – 菜单的图标。请参阅 https://ant.design/components/icon/。默认值为” 无”。
- **auth_needed** (*str, optional*) – the permission needed for user to access this page. e.g. ‘user’ or ‘admin’
- **children** (*list, optional*) – 如果菜单具有子菜单，则设置此选项。默认值为 []

您可以嵌套” 菜单” 以创建子菜单。下面是一个完整的示例:

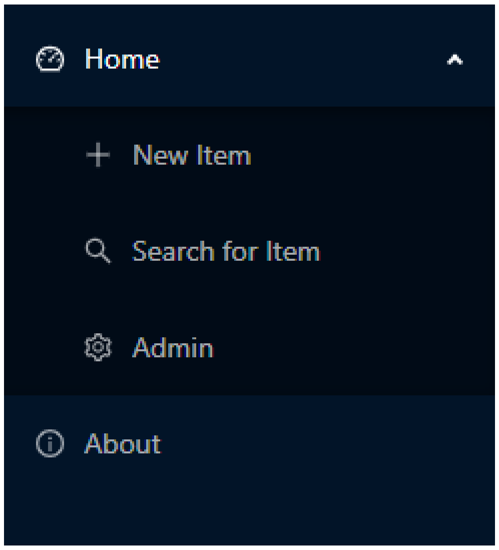
```
app.set_menu(
    [
        MenuItem('Home', '/', icon="dashboard", children=[
```

(下页继续)

(续上页)

```
            MenuItem('New Item', '/new', icon="plus"),
            MenuItem('Search for Item', '/search', icon="search"),
            MenuItem('Admin', '/admin', icon="setting")
        ]),
        MenuItem('About', '/about', icon="info-circle")
    ]
)
```

它看起来像这样：



CHAPTER 6

布局和创建详情页

当您想在页面上显示复杂信息时，您可以使用 Card, Header, DetailGroup, DetailItem 和 Divider 来布局页面:

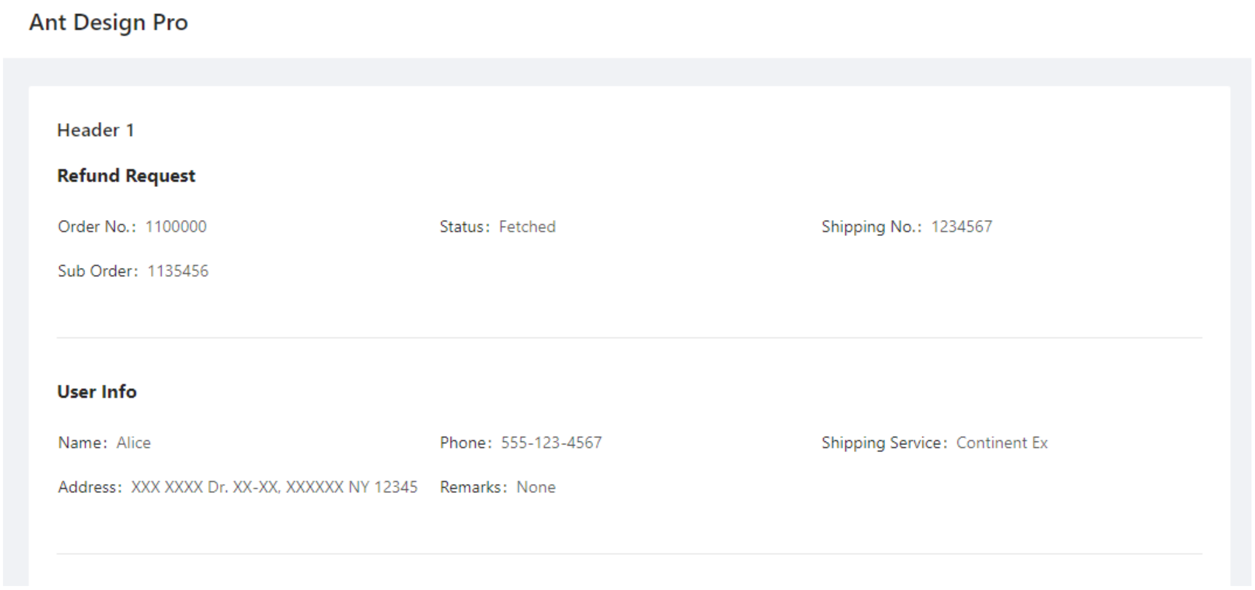
```
@app.page('/detail', 'Detail Page')
def detail_page():
    return [
        Card(content=[
            Header('Header of the record', 1),
            DetailGroup('Refund Request', content=[
                DetailItem('Order No.', 1100000),
                DetailItem('Status', "Fetched"),
                DetailItem('Shipping No.', 1234567),
                DetailItem('Sub Order', 1135456)
            ]),
            Divider(),
            DetailGroup('User Info', content=[
                DetailItem('Name', "Alice"),
                DetailItem('Phone', "555-123-4567"),
                DetailItem('Shipping Service', 'Continent Ex'),
                DetailItem('Address', 'XXX XXXX Dr. XX-XX, XXXXXX NY 12345'),
                DetailItem('Remarks', "None")
            ]),
        ])
```

(下页继续)

(续上页)

```
    ]
```

它看起来像这样：


Ant Design Pro

Header 1

Refund Request

Order No.: 1100000
Status: Fetched
Shipping No.: 1234567
Sub Order: 1135456

User Info

Name: Alice
Phone: 555-123-4567
Shipping Service: Continent Ex
Address: XXX XXXX Dr. XX-XX, XXXXXX NY 12345
Remarks: None

`class adminui.DetailGroup(`*title='', content=None, bordered=False, column=3, size=False, layout='horizontal', id=None*`)`

DetailItem 的容器，用于显示具有多个字段的记录

参数

- **title** – DetailGroup 的标题
- **content** – 一个包含 DetailItem 的列表
- **bordered** – show a bordered description list, see https://3x.ant.design/components/descriptions/
- **size** – default | middle | small - the size of the table when in bordered mode
- **column** – number of columns shown in the list
- **layout** – horizontal | vertical s

`class adminui.DetailItem(`*title='', value='', id=None*`)`

带有标题和值的简单文本，用于在记录中显示字段

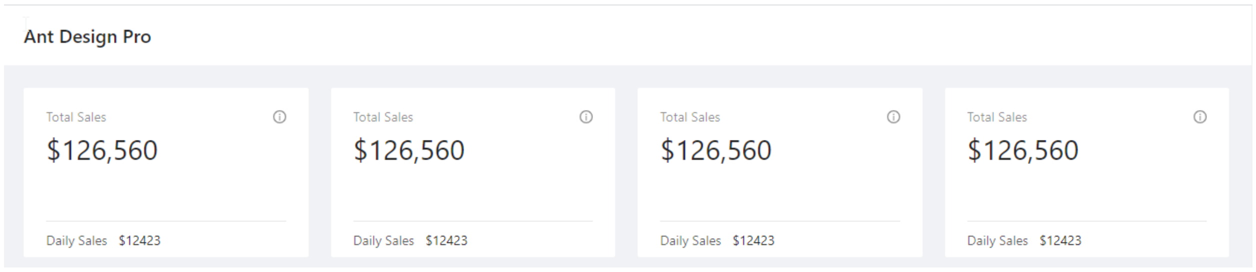
参数

- **title** – 字段的标题
- **value** – 字段的值

如果您正在做仪表盘，可以使用 Row, Column, Statistic。ChartCard 会显示一个用于展示数字和图表的区域，如:

```
@app.page('/dashboard', 'Dashboard')
def dashboard_page():
return [
    Row([
        Column([
            ChartCard('Total Sales', '$126,560', 'The total sales number of xxx',␣
↪height=50,
                footer=[Statistic('Daily Sales', '$12423', inline=True)])
        ]),
        Column([
            ChartCard('Total Sales', '$126,560', 'The total sales number of xxx',␣
↪height=50,
                footer=[Statistic('Daily Sales', '$12423', inline=True)])
        ]),
        Column([
            ChartCard('Total Sales', '$126,560', 'The total sales number of xxx',␣
↪height=50,
                footer=[Statistic('Daily Sales', '$12423', inline=True)])
        ]),
        Column([
            ChartCard('Total Sales', '$126,560', 'The total sales number of xxx',␣
↪height=50,
                footer=[Statistic('Daily Sales', '$12423', inline=True)])
        ]),
    ])
]
```

它会产生这样的页面:



If you just want to display some text, use Paragraph. You may set the color of the Paragraph:

```
Paragraph("The text of Paragraph", color="red")
```

If you wish to format rich text or other complex content, you may use RawHTML Element. Beware this is dangerous because if you pass unfiltered user text (e.g. from a piece of user inputted text stored in the database), this user text may contain dangerous code that may run on the client' s computer:

```
RawHTML('a raw <font color="red">HTML</font>')
```

下面是与布局相关的类:

`class adminui.Card`(*title=None*, *content=None*, *id=None*)

显示一个白色的框，作为容器

参数

- `title` – 容器的标题
- `content` – 将在容器内显示的页面元素的列表

`class adminui.Header`(*text=”*, *level=4*, *id=None*)

在屏幕上显示标题文本

参数

- `text` – 标题的文本正文
- `level` – 标头级别。级别 =1 表示第一级标头（h1）

`class adminui.Paragraph`(*text=”*, *id=None*, *color=None*)

显示文本段落

`class adminui.Row`(*content=None*, *id=None*)

用于布局的行元素，可以有多个 Columns（列）

行的宽度将自动由 len(content) 计算。例如，包含四列的行将构成 4 列布局。自动适配小屏幕和移动设备。

参数 `content` – 列对象列表

`class adminui.Column`(*content=None*, *size=1*, *id=None*)

多列布局行中的列

参数

- `content` – 一个页面元素的列表
- `size` – 列宽度的”权重”。例如，两个 size=1 的列具有相同的宽度; 但 size=2 的列和 size=1 的列将构成 2 比 1 的宽度布局。

`class adminui.ChartCard`(*title=None*, *value=None*, *tooltip=None*, *footer=None*, *content=None*, *height=46*, *id=None*)

带有值、工具提示和页脚的卡片容器。主要用于仪表盘

参数

- `title` – 容器标题

- **value** – （str），卡片上显示的大字
- **tooltip** – 当用户在工具提示图标上悬停时显示的文本
- **footer** – 页面元素列表，将在卡片页脚上显示
- **content** – 页面元素列表，将作为卡片内容显示
- **height** – 高度，使格列的卡片看起来同高

class adminui.Statistic(*title=", value=0, show_trend=False, inline=False, id=None*)

用于显示统计数字的文本片段，可能包括一个小趋势箭头（向上或向下）

参数

- **title** – 统计数字的标题
- **value** – 数字本身
- **show_trend** – 如果设置为 True，则数字为正数时出现上箭头，为负数时出现下箭头
- **inline** – 如果设置为 True，则标题和值将位于同一行中，而且使用小字体

CHAPTER 7

带参数的页面

有时，您需要根据 url 参数返回不同的页面。例如:

```
http://yourdomain.com/article/3
```

显示了数据库中的第三篇文章。在这种情况下，这样注册页面:

```
@app.page('/article', 'Article')
def form_page(param):
    return [ ... the content of the page ... ]
```

然后，当用户带着 `/article/<param>` 这样的 url 时，`param` 部分将作为处理函数的第一个参数传递。

If you the page has multiple parameters in the url like `/page_name?param_a=value_a&param_b=value_b`, you can get them with the second parameter in your page handler:

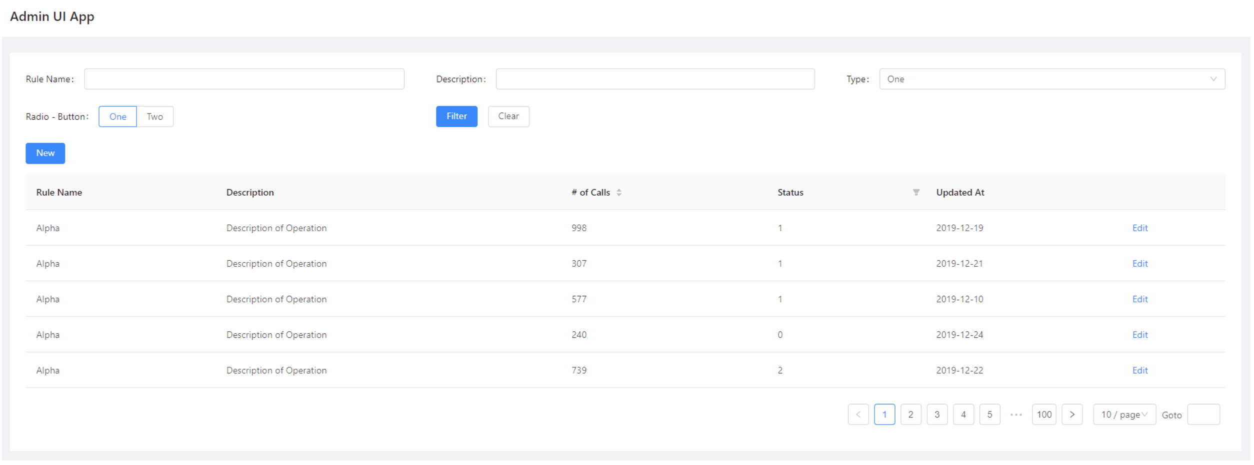
```
@app.page('/detail', 'Detail Page')
def detail_page(arg, all_args):
    ... # all_args will be a dictionary of all the params
```

CHAPTER 8

使用数据表格

表格页面看起来这样:

```
@app.page('/', 'Table')
def table_page():
    return [
        Card(content = [
            DataTable("Example Table", columns=table_columns,
                data=TableResult(table_data))])
    ]
```



Admin UI App

Rule Name:
Description:
Type: One
Radio - Button: [✓ One] Two
Filter Clear
New

Rule Name	Description	# of Calls	Status	Updated At	
Alpha	Description of Operation	998	1	2019-12-19	Edit
Alpha	Description of Operation	307	1	2019-12-21	Edit
Alpha	Description of Operation	577	1	2019-12-10	Edit
Alpha	Description of Operation	240	0	2019-12-24	Edit
Alpha	Description of Operation	739	2	2019-12-22	Edit

< 1 2 3 4 5 ··· 100 > 10 / page Goto

使用 DataTable 类来构造一个用来展示数据的表:

`class adminui.DataTable`(*title='', columns=[], data=[], row_actions=[], table_actions=[], filter_form=None, on_data=None, size='default', scroll_x=None, scroll_y=None, id=None*)

将数据表插入页面

参数

- **title** – 表的标题
- **columns** – a list-of-dictionaries as column definition. e.g.: [{ 'title' : 'Rule Name' , 'dataIndex' : 'name' }, …other columns] [{ 'title' : 'Rule Name' , 'dataIndex' : 'name' , 'sorter' : True, 'filterOptions' : ['abc' , 'def']}, …other columns] for each column,

  title: the column title dataIndex: its key for the TableResult data dictionary (optional) sorter: (True/False) the column is sortable (optional) filterOptions: ([strings]) a list of options shown as filters (optional) linkTo: display the data as a link to another field of the dataIndex specified (optional) status: dataIndex of another column, shown as the status badge fo the data. The value of the other column must be one from success | processing | default | error | warning
- **data** – TableResult 对象，作为表的初始数据
- **row_actions** – TableRowAction 对象的列表，在表的每一行中展示链接等用户操作。如果您不需要任何操作，将其留空
- **table_actions** – 显示在表顶部的页面元素的列表。如" 新建" 按钮等。
- **on_data** – a callback function that returns a TableResult object if the user turns a page. an argument will be passed as { 'current_page' :…, 'page_size' :…, 'sorter' : {sorted_column_data_index}_{ascend|decsend}} Leave it None if you' re sure there is only one page of data.
- **size** – size of the table (default | middle | small)

8.1 为表准备数据

DataTable 需要提供 columns（列的定义）和 data（表格数据）。columns 字段中所需的数据如下所示：

```
table_columns = [
    {'title': 'Rule Name', 'dataIndex': 'name'},
    {'title': 'Description', 'dataIndex': 'desc'},
    {'title': '# of Calls', 'dataIndex': 'callNo'},
    {'title': 'Status', 'dataIndex': 'status'},
    {'title': 'Updated At', 'dataIndex': 'updatedAt'}
]
```

每列为字典，带有 `title` 和 `dataIndex` 。`dataIndex` 将用作提供的表行数据的键：

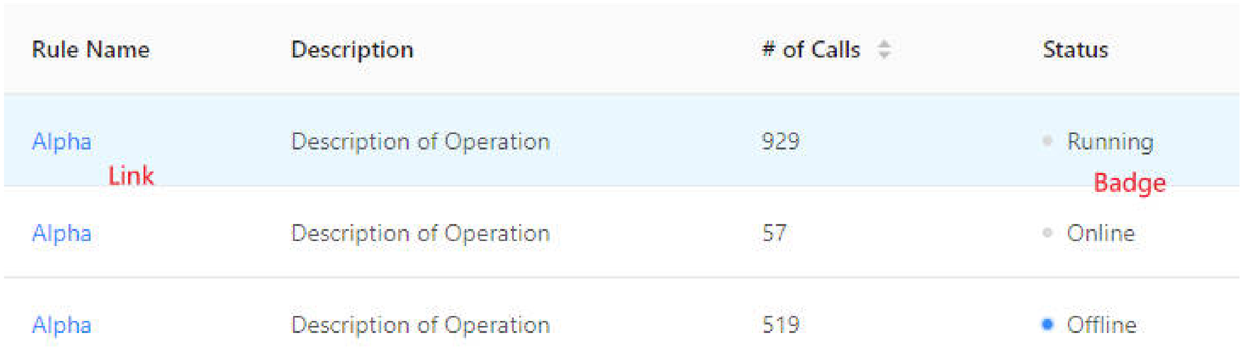
```
table_data = [
                {
                "callNo": 76,
                "desc": "Description of Operation",
                "id": 0,
                "name": "Alpha",
                "status": 3,
                "updatedAt": "2019-12-13"
            },
            ... other rows
]
```

`table_data` 接受一个 `TableResult` 对象：

```
DataTable("Example Table", columns=table_columns,
                data=TableResult(table_data))
```

`TableResult` 在分页时也有用

8.2 Render a column as a link



Rule Name	Description	# of Calls	Status
Alpha	Description of Operation	929	● Running
Alpha	Description of Operation	57	● Online
Alpha	Description of Operation	519	● Offline

If you want a column in the table shown as a link, set the column definition as:

```
{'title': 'Rule Name', 'dataIndex': 'name', 'linkTo': 'link'},
```

Then the column' s `title` will be shown as a link, linking to data field with key `link`

8.3 Render a column as a badge

To render badges on columns, define the column as:

```
{'title': 'Rule Name', 'dataIndex': 'name', 'status': 'badgeStatus'},
```

Whereas `badgeStatus` in the example is the dataIndex of another column, whose value takes from `success | processing | default | error | warning`, which will be the appearance of the badge.

8.4 分页

如果表有多页数据，并且只能在每页中显示一部分数据（例如，展示从数据库读取的记录），则需要分页。

需要做两件事：

首先，在 "TableResult"中提供记录总数、当前页面编号和每页数据条目数。这样系统知道如何展示分页按钮:

```
DataTable("Example Table", columns=table_columns,
            data=TableResult(table_data, 1000), on_data=on_page)
```

class `adminui.TableResult`(*data=[]*, *total=None*, *current_page=1*, *page_size=10*)

数据表的" data" 属性使用的数据，或在分页请求时返回

参数

- **data** – 用作表数据的列表，每个元素是一个字典。例如：[{id: 1, name: 'Alice', '_actions': ['view', 'edit'], …更多列} …更多信息行] 。这里 id 是必要的，" _actions" 字段展示了用户可以在这行做什么操作。如果没有 _actions，则将展示所有定义的操作；如果 _actions 为空列表，则用户无法对此行做操作。
- **total** – 所有记录的总数。如果大于 len(data)，则显示分页条。
- **current_page** – 记录的当前页面，因此前端将知道要突出显示的页面
- **page_size** – 一页中有多少条记录。

其次，为 `DataTable` 提供 `on_data` 回调函数，以便在用户翻页时加载数据:

```
def on_page(args):
    records = (... load records somewhere from the database,
        with args['current_page'] and args['page_size'])
    return TableResult(mock_table_data(5), 1000, args['current_page'])
```

这样，就完成了一个提供多页数据的表。

8.5 每行的操作链接

您还可以向每行添加操作链接，这样用户可以在每行做一些操作。例如，对于显示“文章”的表，用户可以“编辑”每一篇文章。

此时，将 `row_actions` 参数设为拥有 `TableRowAction` 的列表:

```
DataTable("Example Table", columns=table_columns,
                data=TableResult(table_data)
                row_actions=[
                    TableRowAction('edit', 'Edit', on_click=on_edit),
                    TableRowAction('edit', icon='edit', on_click=on_edit), # use icons
                ])
```

class `adminui.TableRowAction`(*id*, *title=”*, *on_click=None*, *icon=None*)

表示表每行中显示的操作链接

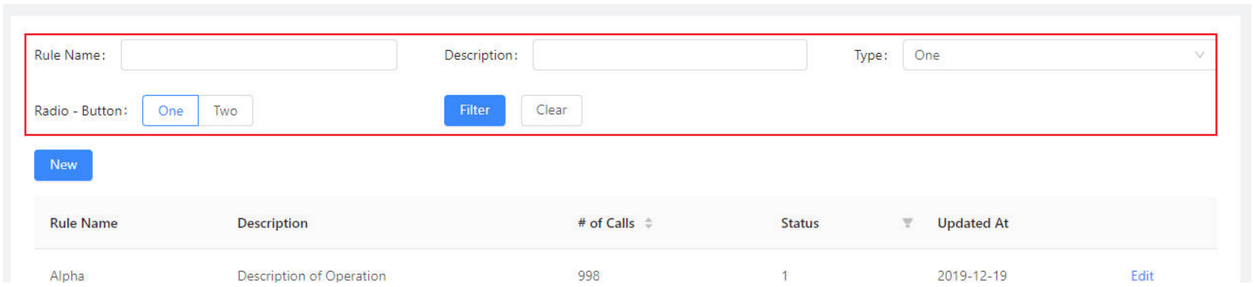
参数

- **id** – 操作的 ID，用于表结果数据的”_actions”字段。
- **title** – 操作链接的标题
- **on_click** – 当用户单击链接时调用的回调函数。数据行（字典）将作为函数的参数传递

在这种情况下，表每行的右侧将显示一个”Edit”链接。如果用户单击其中一个函数，则传递的函数”on_edit”将调用：

```
def on_edit(record):
    ...do something with the table record
```

8.6 Use Filter Forms



Rule Name	Description	# of Calls	Status	Updated At	
Alpha	Description of Operation	998	1	2019-12-19	Edit

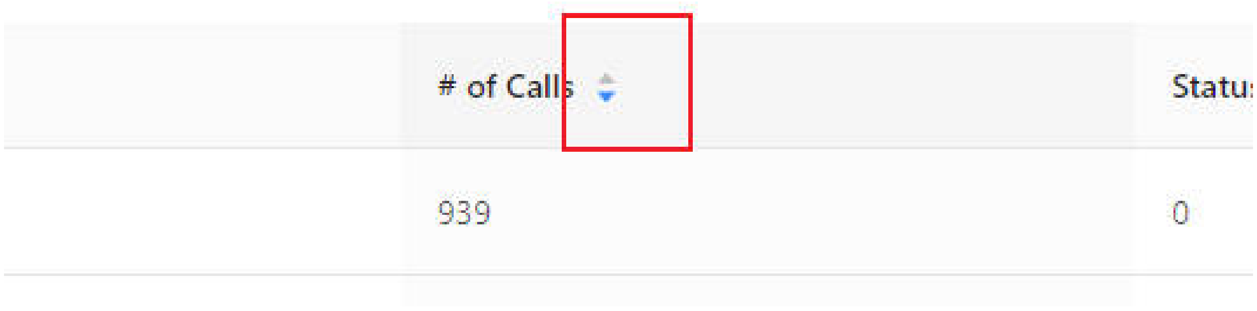
You may add a filter form, to let users search in your table. Each time the user submits the form or switch pages, values in the form will be passed along to the `on_data` callback.

To add filter form, set `filter_form` field in DataTable element:

```
DataTable("Example Table", columns=...,  data=..., on_data=on_page,
            filter_form=FilterForm([
                TextField('Rule Name'),
                TextField('Description'),
                SelectBox('Type', data=['One', 'Two', 'Three'], placeholder="Select One
↪"),
                RadioGroup('Radio - Button', data=[['One', 1], ['Two', 2]], format=
↪'button'),
            ], submit_text='Filter', reset_text='Clear'),
            row_actions=[...],
            table_actions=[...])
```

Note that you can change the text on submit button and reset button, using `submit_text` and `reset_text` field.

8.7 Sortable and Filterable Columns



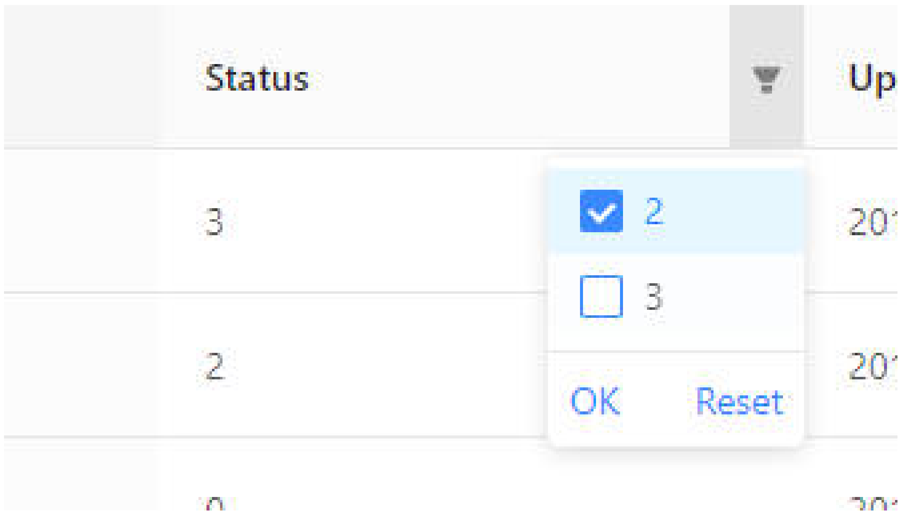
To make table column sortable, set `sorter=True` to the column definition:

```
table_columns = [
    {'title': '# of Calls', 'dataIndex': 'callNo', 'sorter': True},
    ...
]
```

Then, when the user click on the header of the column, `on_data` callback will receive an additional `sorter` argument like:

```
sorter: "callNo_descend"
```

Separated by underscore, the first part is the `dataIndex` field, the second part is descend or ascend according the current sorting status.



To make a table column filterable, set filters on the column definition like:

```
{'title': 'Status', 'dataIndex': 'status', 'filters': [{'text': 2, 'value': 2}, {'text':␣
↪3, 'value': 3}]},
```

Then the column will be filterable. When the user filters some columns, `on_data` will receive arguments like:

```
filters: {status: "3,2"}
```

where the key will be the filtered column' s data index, and the value will be the filtered values, separated by commas.

8.8 Change the height of rows

pass `size` to DataTable will allow you to change the row height. You may choose from `'default' | 'middle' | 'small'`:

```
DataTable(..., size='small')
```

8.9 Scroll X for the table

setting the `scroll_x` and `scroll_y` attributes for `DataTable`, will let scroll bar show up when the table is too long. Useful for tables with many columns or rows:

```
DataTable(..., scroll_x=1000)
```

此处列出了表的完整示例 https://github.com/bigeyex/python-adminui/blob/master/python/example_table.py

CHAPTER 9

使用图表

9.1 Line Chart

`LineChart` takes a list of of the data (numbers, will be the x axis), and a list of the lables:

```
chart_labels = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
chart_data = [1.5, 2.3, 4.3, 2.2, 5.1, 6.5, 2.3, 2.3, 2.2, 1.1]
LineChart(chart_data, chart_labels)
```

You may put more than 1 series on the line chart:

```
LineChart({'series1': chart_data, 'series2': chart_data2}, chart_labels)
```

set `show_area=True` to fill the area of the chart, and set `smooth=False` to stop smoothing the line.

You may set the `height` in every type of chart.

class adminui.LineChart(*data=[], labels=None, show_axis=True, show_line=True, show_area=False, smooth=True, height=300, line_color=None, area_color=None, columns=['x', 'y'], id=None*)

创建折线图

参数

- **data** – the data shown on the chart. It can be: a) an array like [2, 3, 4, 5] b) a dict of series, like { ‘series1’ : [2, 3, 4, 5], ‘series2’ : [6, 7, 8, 9] }

- **labels** – labels corresponding to the data. Must be to the same length of the data e.g. ['a' , 'b' , 'c' , 'd']
- **show_axis** – 为 True 时显示坐标轴和坐标轴标注
- **show_line** – 如果想要隐藏折现、只显示填充区域，则设为 False
- **show_area** – 为 True 时显示填充区域
- **smooth** – 为 True 时改折线为曲线
- **height** – 图表高度
- **line_color** – line color as a string if data is a list else an array of colors
- **area_color** – area color as a string if data is a list else an array of colors

9.2 Bar Chart

Basic usage:

```
BarChart(chart_data, chart_labels)
```

Multiple bars will be put side-by-side. Stack them with `stack=True`:

```
BarChart({'series1': chart_data, 'series2': chart_data2}, chart_labels, stack=True),
```

class adminui.BarChart(*data=[], labels=None, show_axis=True, height=300, color=None, columns=['x', 'y'], stack=False, id=None*)

创建条形图

参数

- **data** – the data shown on the chart. It can be: a) an array like [2, 3, 4, 5] b) a dict of series, like { 'series1' : [2, 3, 4, 5], 'series2' : [6, 7, 8, 9] }
- **labels** – labels corresponding to the data. Must be to the same length of the data e.g. ['a' , 'b' , 'c' , 'd']
- **show_axis** – 为 True 时显示坐标轴和坐标轴标注
- **height** – 图表高度
- **color** – color as a string if data is a list else an array of colors

9.3 Pie Chart

Basic usage:

```
PieChart(chart_data, chart_labels)
```

class **adminui.PieChart**(*data=[]*, *labels=None*, *height=300*, *color=None*, *title=None*, *id=None*)

创建条形图

参数

- **data** – the data shown on the chart. e.g. an array like [2, 3, 4, 5]
- **labels** – labels corresponding to the data. Must be to the same length of the data e.g. [‘a’ , ‘b’ , ‘c’ , ‘d’]
- **height** – 图表高度
- **title** – the title of the chart

9.4 Scatter Plot

Scatter Plot takes `x`, `y` series, and optionally `size` and `color` (both can be a constant or an array of data):

```
ScatterPlot(list_of_x, list_of_y),
ScatterPlot(list_of_x, list_of_y, color=list_of_color_data, size=list_of_size_data),
```

class **adminui.ScatterPlot**(*x=[]*, *y=[]*, *labels={'color': 'color', 'size': 'size', 'x': 'x', 'y': 'y'}*, *height=300*, *color=None*, *size=3*, *opacity=0.65*, *id=None*)

创建条形图

参数

- **x** – data in the x axis e.g. an array like [2, 3, 4, 5]
- **y** – data in the y axis
- **color** –

  a) color of the points; b) a series of data shown as the color e.g. [1, 2, 1, 2]
- **size** –

  a) (int) size of data points; b) a series of data as the size e.g. [1, 2, 3, 1, 2, 3]
- **labels** – labels of the x, y axis
- **height** – 图表高度
- **opacity** – the opacity of the data points

An example with chart is listed here: https://github.com/bigeyex/python-adminui/blob/master/python/example_chart.py https://github.com/bigeyex/python-adminui/blob/master/python/example_dash.py

CHAPTER 10

Require users to Log in

AdminUI comes with a login page. To enable it, specify a function to handle login:

```
@app.login()
def on_login(username, password):
    if username=='alice' and password=='123456':
        return LoggedInUser("Alice")
    else:
        return LoginFailed()
```

The function will receive the username and password users inputted; you need to return LoggedInUser or LoginFailed depending on the result. Instead of a simple if statement, typically you need to check the credential against a database or something.

If you want to redirect logged in user to a different page, set the redirect_to argument in LoggedInUser, like:

```
return LoggedInUser("Alice", redirect_to='/detail')
```

The returned LoggedInUser and LoginFailed may contain more information:

`class adminui.LoggedInUser`(*display_name=", auth=['user'], avatar='https://gw.alipayobjects.com/zos/antfincdn/XAosX*
user_info=None, redirect_to=None)

Returned by login handler, represent a successfully logged in user.

参数

- **`display_name`** – the display name of the user

- **auth** – a list of permission string the user have. Will be checked against in pages or menus
- **avatar** – the avatar of the user
- **user_info** – info for future use, accessible by app.current_user()[‘user_info’]

`class adminui.LoginFailed`(*title='Login Failed', message='Username or password is incorrect'*)

Returned by login handler, represent a failed login attempt

参数

- **title** – the title shown in the error message. default: ‘Login Failed’
- **message** – the error message content. default: ‘Username or password is incorrect’

10.1 Pages requires authorization

By default, any user can visit the page you described. If you want to show the page to users only with certain permission, add an auth_needed attribute to your page:

```
@app.page('/', 'Logged In', auth_needed='user')
def form_page():
    return [
        Card('Logged In', [
            Header('You are logged in')
        ])
    ]
```

In this way, only logged in users can visit this page. Other users will be redirected to the login page.

You may also use *auth_needed=' admin'* , then a user logged in with:

```
LoggedInUser("Alice", auth=['user', 'admin'])
```

May access this page, since the user Alice has ‘admin’ authority.

10.2 Menus for different user roles

Menus can also be protected, by attaching *auth_needed* attribute. For example:

```
MenuItem('User Home', '/user_home', icon="dashboard", auth_needed='user')
```

10.3 Forget password link and register link

You may change the forget password link, or register link on the login page, like:

```
app.register_link={'Sign Up': '/signup'}
app.forget_password_link={'Forget Password': '/forget'}
```

If they are set to `None`, they won’ t show up on the screen.

CHAPTER 11

Customization

You can change the title of your app:

```
app.app_title = "AdminUI APP"
```

And you can change the logo. It accepts an URL:

```
app.app_logo = "http://...."
```

And you can change the copyright (in the footer) text like:

```
app.copyright_text = 'App with Login by AdminUI'
```

(Leave a blank string if you don' t need it)

Or change the footer links:

```
app.footer_links = {'Github': 'https://github.com/bigeyex/python-adminui', 'Ant Design':
↪'https://ant.design'}
```

Change menu style with:

```
app.app_styles = {'nav_theme': 'light', 'layout': 'topmenu'}
```

where nav_theme takes 'dark' (default) or 'light' ; layout takes 'sidemenu' (default) or 'topmenu'

11.1 Favicons

You may set another favicon other than the default one:

```
app.app_favicon = os.path.join(Path(__file__).parent.absolute(), 'new-favicon.png')
```

CHAPTER 12

Serve Static Files

If you have static files, like images, to serve, or you want to let the users access the upload folder, you can add more static path with:

```
app.static_files = {'/upload': os.path.join(Path(__file__).parent.absolute(), 'upload')}
```

then the user can access the files in the upload folder (of the starting python file' s path), with urls like `/upload/....` `app.static_files` takes a dictionary, with the key as the path in the URL, and the value as the path in your file system.

CHAPTER 13

Other Components

13.1 Icons

You can add one of the Ant Design icons using `Icon`:

```
Icon('sync')
```

A full list can be seen at: https://3x.ant.design/components/icon-cn/

only outline style icons are supported.

You may set the color and size of the icon.

class adminui.Icon(*name=”, color='rgba(0, 0, 0, 0.8)', size=16, rotate=False, spin=False, id=None*)

Display an Ant Design Icon

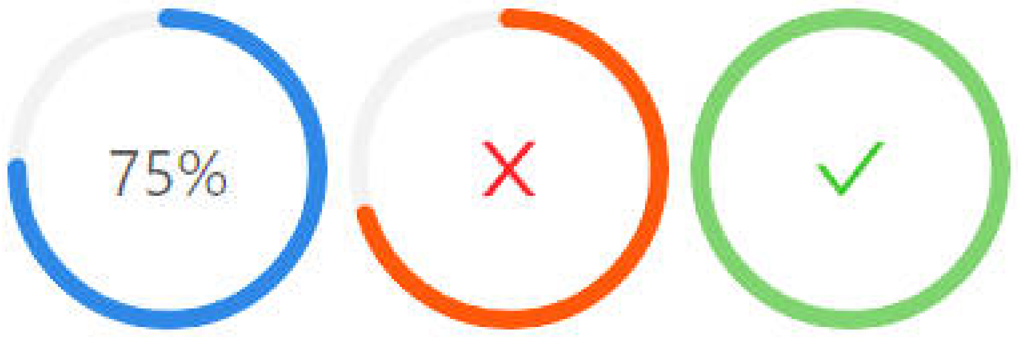
参数

- **name** – name of the icon, see https://ant.design/components/icon/ for a detailed list
- **color** – color of the icon
- **size** – (font) size of the icon
- **rotate** – (bool)rotation angle of the icon
- **spin** – (bool)whether the icon is spinning

13.2 Progress

You can draw a progress bar with:

```
Progress(30)
Progress(30, format='circle')
```



13.3 Image

To add an image:

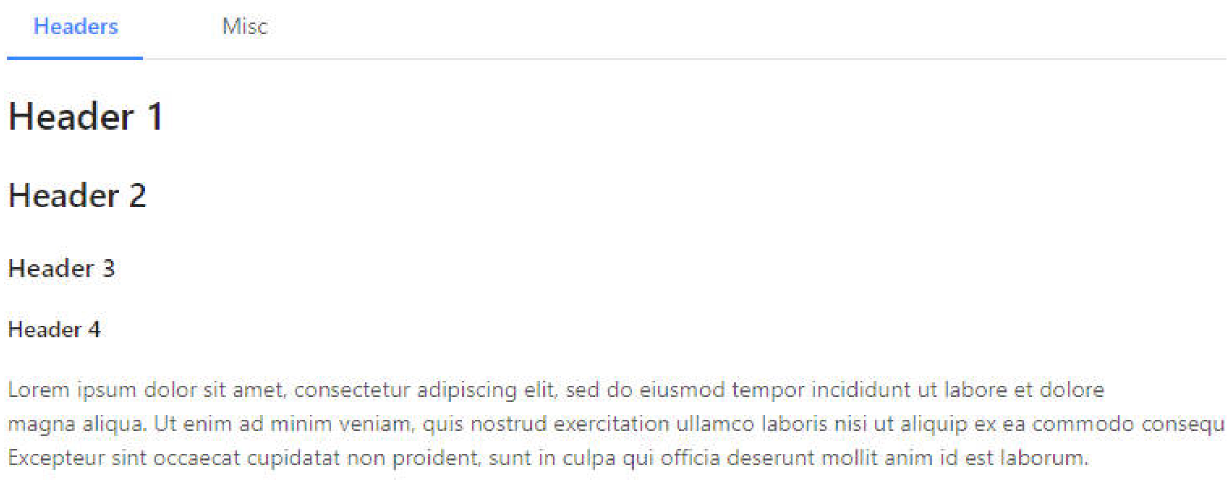
```
Image('https://url-of-the-image', 'alt-text', width=300)
```

13.4 Group (HTML Div)

Use Group to group content together, so you may change them with `UpdateElement` page action (in fact, it just creates a <div/> element in html):

```
Group(id='id of the content', content=[...content in the group])
```

13.5 Tabs



Header 1

Header 2

Header 3

Header 4

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequ
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Use tabs to group content together:

```
Tabs([
    Group(name='title of tab 1', content=[
        ... content of tab 1
    ]),
    Group(name='title of tab 2', content=[
        ... content of tab 2
    ]),
]),
```

You may set the tab appearance with **position**, **format**, and **size**, see

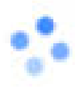
`class adminui.`**`Tabs`**(*content=[], position='top', format='line', size='default', id=None*)

Display a group of tabs. Each content item is a tab

参数

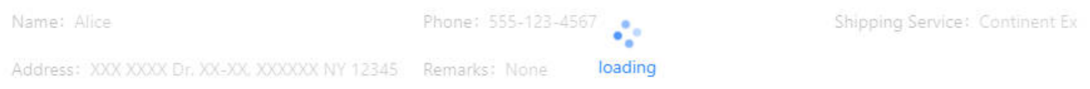
- **position** – top | left | bottom | right, where the tab is (decide if it’ s horizontal or vertical)
- **format** – line | card - the style of the tabs
- **size** – default | large | small

13.6 Spin



Creates a spinning wheel:

```
Spin()
```



Name: Alice

Phone: 555-123-4567

Shipping Service: Continent Ex

Address: XXX XXXX Dr. XX-XX, XXXXXX NY 12345

Remarks: None

loading

You can also setup a region masked under a spinning wheel. When the loading is done, remove it with Page Actions:

```
Spin('loading', content=[
    ... content under the mask...
]),
```

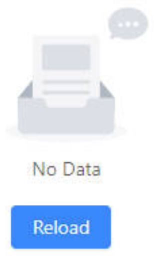
`class adminui.Spin(`*title=", content=[], size='default', id=None*`)`

A spinning icon, indicating something is loading

参数

- **`size`** – default | small | large: the size of the spinning wheel
- **`content`** – (optional), when creating a container with a "loading" mask, put its content here
- **`title`** – the text shown below the spinning wheel

13.7 Empty Status



Display the empty status:

```
Empty()
```

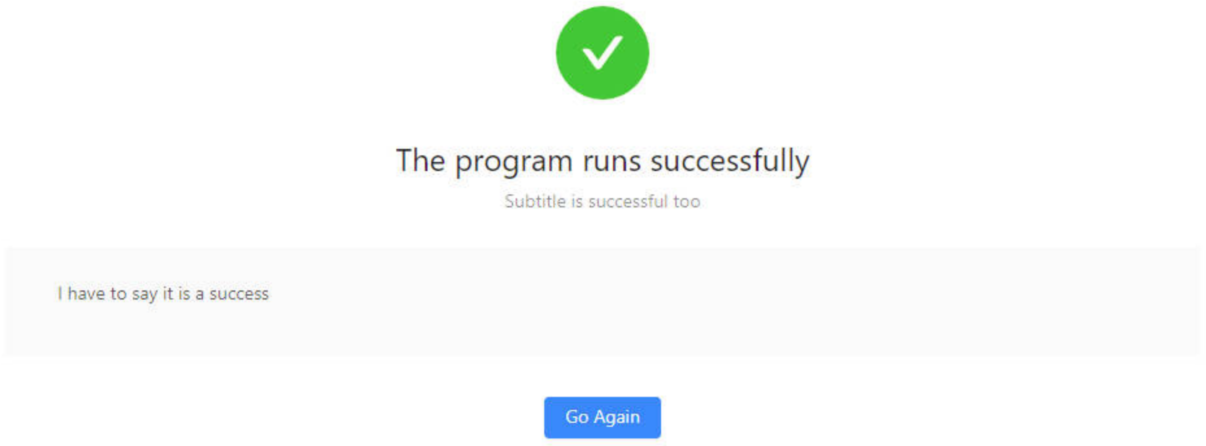
class adminui.Empty(*title=None*, *content=[]*, *simple_style=False*, *id=None*)

Display an Empty status

参数

- **title** – the description text shown on the empty indicator
- **content** – (optional), put additional buttons or elements below the icon
- **simple_style** – set True to deploy a simple style of empty box

13.8 Result


The program runs successfully
Subtitle is successful too
I have to say it is a success
Go Again

Display the result of an operation:

```
Result('The program runs successfully')
```

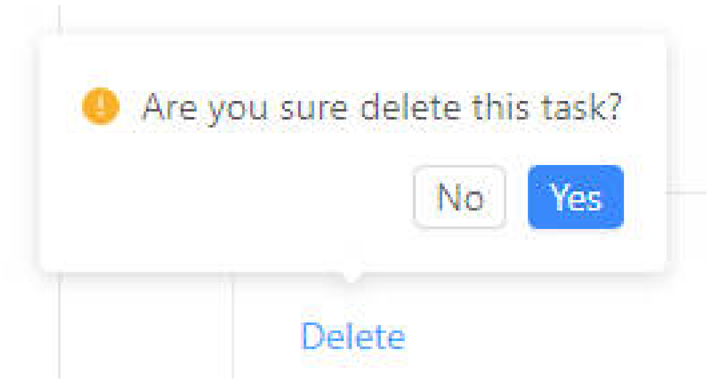
class adminui.Result(*title=None*, *status='success'*, *sub_title=None*, *content=[]*, *extra=[]*, *id=None*)

Display the result (success, failure, 403, 404⋯) of an action

参数

- **title** – the title of the result feedback
- **sub_title** – the sub-title of the result section
- **status** – 'success' | 'error' | 'info' | 'warning' | '404' | '403' | '500'
- **content** – put additional buttons or elements inside a result box
- **extra** – extra action buttons on the result section

13.9 Popconfirm



Let the user confirm before performing an action:

```
Popconfirm('Are you sure to do something?', on_submit=on_submit, content=[
    ... put adminui elements here, which will trigger the confirm box when clicked ...
], data=CUSTOM_DATA)
```

note that `CUSTOM_DATA` may be passed to the `on_submit` callback, when the user choose YES in the confirm box.

You may customize the OK and Cancel button of the pop confirm box by setting `ok_text` and `cancel_text` of the element

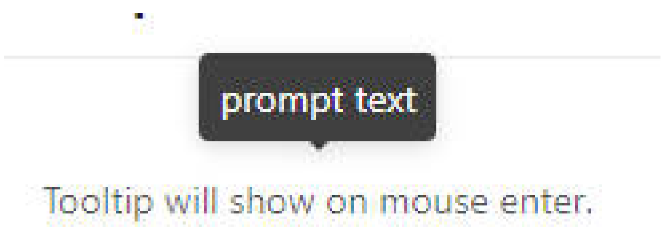
class adminui.Popconfirm(*title=None, content=[], on_submit=None, ok_text='Yes', cancel_text='No', data=None, id=None*)

Before the user perform an action, ask him/her to confirm twice.

参数

- **title** – the text shown on the pop confirm box
- **content** – the enclosed content, which shows the Popconfirm when clicked
- **on_submit** – (func) callback after user clicked 'ok'
- **ok_text** – text shown on the OK button
- **cancel_text** – text shown on the Cancel button
- **data** – data which will passed as the parameter to the callback

13.10 Tooltip



Display a tooltip when the mouse is hovering some elements:

```
Tooltip('Text on the tooltip', [
    ... content which will trigger the tooltip when hovered ...
])
```

class adminui.**Tooltip**(*title=None*, *content=[]*, *placement='top'*, *id=None*)

Show a tooltip when the user moves the mouse upon its content

参数

- **title** – the text shown on the tooltip
- **content** – the content, which will show the tooltip when the mouse is on it
- **placement** – one of top | left | right | bottom | topLeft | topRight | bottomLeft | bottomRight | leftTop | leftBottom | rightTop | rightBottom

CHAPTER 14

Organizing your App

When your app grows bigger, you may need to split it to multiple files.

While there are many ways to do this in Python, adminui provides a simple way for simple apps.

For (a minimal simple) example, you want to make an app with a home page and a detail page, so the structure is like:

```
home.py
detail.py
```

in `home.py`, you layout the home page like:

```
from adminui import *

app = AdminApp()
@app.page('/', 'home')
def home_page():
    ... layout the home page ...

app.set_as_shared_app() # set the app as the shared app, so it can be accessed globally
import detail           # import all the other files in your project (you can import␣
↪files recursively
                        # meaning if you have admin_pages.py, you can import all the␣
↪admin related pages there
```

(下页继续)

(续上页)

```
                                # and in home.py just import admin_pages)

if __name__ == '__main__':
    app.run()
```

note the `app.set_as_shared_app()` makes the app exposed in the whole project, and then in the `home.py` , `detail.py` is imported.

in `detail.py`, use `AdminApp.shared_app()` to access the app add add more pages:

```
# content in detail.py
from adminui import *

app = AdminApp.shared_app() # now you have the app ready to add more pages

@app.page('/detail', 'Detail Page')
def detail_page():
    ... layout the detail page ...
```

CHAPTER 15

索引和表格

- genindex
- search

索引