第 5 章

用户输入类组件

用户输入类组件是指允许用户在 Web 应用界面上输入信息的各种组件，它们可以获取用户输入并传递给程序进行处理或显示。常见的有按钮类、文本输入类、日期输入类、数字输入类等。通过这些组件，用户可以与应用进行交互，执行相应的操作。利用好各种功能可以大大提升 Web 应用的可用性和用户的交互体验。Streamlit 库中提供了多种用户输入类组件，使用起来也非常简单和方便。

5.1 普通按钮

Streamlit 库可以使用 st.button()方法展示普通按钮，具体的使用说明如下：

```
    st.button(label, key=None, help=None, on_click=None, args=None, kwargs=None, *,
type="secondary", disabled=False, use_container_width=False)
```

（1）label：向用户解释此按钮用途的简短标签。该参数的类型为字符串类型，该参数支持部分 Markdown 语法，如粗体、斜体、删除线、内联代码和表情符号。

（2）key：用于生成唯一标记该组件的键，避免与同类组件混淆。该参数的类型为字符串、整型或 None，默认为 None，将自动生成。

（3）help：可选的工具提示。当鼠标经过按钮时将显示在按钮的上方，默认为 None，表示没有工具提示。

（4）on_click：设置单击该组件的回调函数或方法，默认为 None。

（5）args：传递给 on_click 参数对应回调函数或方法的位置参数元组，默认为 None。

（6）kwargs：传递给 on_click 参数对应回调函数或方法的关键字参数字典，默认为 None。

（7）type：可选参数，用于指定按钮类型的字符串。对于带有额外强调样式的按钮应该是 primary，对于普通样式的按钮应该是 secondary。该参数只能由关键字传入，默认为 secondary。

（8）disabled：是否将该组件设置为禁用状态，默认值为 False，是可用状态。

（9）use_container_width：是否将数据框的宽度设置为父容器的宽度。默认值为 False，

不设置。

（10）返回值：如果在 Web 应用上次运行时单击了该按钮，则为 True，否则为 False。

新建一个名为 button_simple.py 的 Python 文件，整体的思路是先创建两个普通按钮，第 1 个按钮用于调用 st.button()方法，如果该按钮被单击，则会在页面上显示“按钮被单击了!”的文本。第 2 个按钮只是简单地调用 st.button()方法以创建一个按钮，没有绑定单击事件。整个文件中的代码如下：

```
#第 5 章/button_simple.py
import streamlit as st

st.subheader("简单的普通按钮示例")
if st.button('单击这里'):
    st.write('按钮被单击了!')

st.button('另一个按钮')
```

初始化效果如图 5-1 所示。

当单击第 1 个按钮后，整个页面会刷新，紧接着会出现“按钮被单击了!”的字样。该 Web 应用所显示的内容如图 5-2 所示。

图 5-1　初始化状态

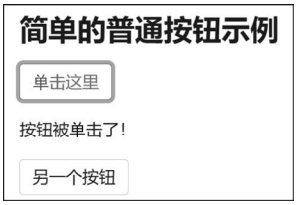

图 5-2　单击第 1 个按钮后的状态

然后单击第 2 个按钮，整个页面会再次刷新，紧接着“按钮被单击了!”的字样会消失，该 Web 应用所显示的内容如图 5-3 所示。

图 5-3　单击第 2 个按钮后的状态

读者可以结合代码理解一下，这两次单击及各自的结果。尤其是单击第 2 个按钮之后，字样会消失。这是因为第 2 个按钮虽然没有指定绑定回调事件，但是单击它也会刷新整个 Web 应用，而这一次因为单击的是第 2 个按钮，而不是第 1 个按钮，所以第 1 个按钮的返回值为 False，由此可知不会显示“按钮被单击了!”的字样。

5.2　单选按钮

14min

单选按钮允许用户从多个互斥的选项中选择一个。在 Streamlit 中通过 st.radio()方法实现单选按钮，具体的使用说明如下：

```
    st.radio(label, options, index=0, format_func=special_internal_function,
key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False,
horizontal=False, label_visibility="visible")
```

（1）label：向用户解释此按钮用途的简短标签。该参数的类型为字符串类型，该参数支持部分 Markdown 语法，如粗体、斜体、删除线、内联代码和表情符号。

（2）options：设置单选按钮的选项。该参数的类型为序列类型，如列表、numpy.ndarray、pandas.Series、pandas.DataFrame 或 pandas.Index 等，如果是 pandas.DataFrame 类型，则选择第 1 列。默认会将这些序列类型中的每个值转换成字符串类型。

（3）index：第 1 次渲染时，预选的选项的索引，默认为 0，即第 1 个选项会被预选。

（4）format_func：对选项进行格式化的函数。它接收原始选项作为参数，并输出该选项显示的标签。对整个方法的返回值没有任何影响，默认为 special_internal_function 函数，实际就是 str()函数。

（5）key：用于生成唯一标记该组件的键，避免与同类组件混淆。该参数的类型为字符串、整型或 None，默认为 None，将自动生成。

（6）help：可选的工具提示，显示在 label 参数文本的右侧，默认为 None，表示没有工具提示。

（7）on_change：设置该组件选项变化时的回调函数或方法，默认为 None。

（8）args：传递给 on_change 参数对应回调函数或方法的位置参数元组，默认为 None。

（9）kwargs：传递给 on_change 参数对应回调函数或方法的关键字参数字典，默认为 None。

（10）disabled：是否将该组件设置为禁用状态。默认值为 False，是可用状态。该参数只能通过关键字设置。

（11）horizontal：设置是否水平方向排列选项。默认值为 False，即垂直方向排列。该参数只能通过关键字设置。

（12）label_visibility：设置 label 参数的可见性。如果为 hidden，则标签不会显示，但小部件的上方仍然有空白空间（相当于 label=""）。如果为 collapsed，则标签和空间都会被删除。默认为 visible。该参数只能通过关键字设置。

（13）返回值：被选中的选项。

新建一个名为 radio.py 的 Python 文件，整体的思路是通过 3 个示例说明设置不同参数的效果。第 1 个示例设置自定义 format_func 函数，选项的显示将不同于原始选项，并且不会影响返回值。第 2 个示例设置 label_visibility 参数，label 参数不可见且不占位置。第 3 个示

例设置 label_visibility 和 horizontal 参数，label 参数不可见但仍有位置，而且选项都是水平排列的。整个文件中的代码如下：

```
    #第 5 章/radio.py
    import streamlit as st
    #自定义一个名为 my_format_func 的函数
    def my_format_func(option):
        return f'选{option}'

    st.header('单选按钮示例')
    st.subheader('示例 1')
    city = st.radio('选择你最喜爱的城市：', ['北京', '太原', '临汾'], format_func=
my_format_func)
    #根据返回值的不同选择不同的特色回答
    #同时应注意返回值不受自定义 my_format_func 函数的影响
    if city == '北京':
        st.write('你最喜爱的城市是首都**北京**')
    elif city == '太原':
        st.write('你最喜爱的城市是**太原**，它是山西的省会')
    else:
        st.write('你最喜爱的城市是**临汾**，古时称“平阳”')

    st.subheader('示例 2')
    size = st.radio(
        '选择尺码',
        ['S', 'M', 'L'],
        label_visibility='collapsed'
    )
    #没有特色回答，可直接使用 f-strings 的回答
    st.write(f'你选择的是{size}号')

    st.subheader('示例 3')
    st.write('选择午饭')
    #将标签设置为"hidden"
    #设置水平排列
    lunch = st.radio(
        '你中午想吃什么？',
        ['馒头', '大米', '面条'],
        horizontal=True,
        label_visibility='hidden'
    )
    st.write(f'你选择的是吃{lunch}')
```

读者可以运行上面的代码，以便通过实际感受进行理解，初始效果如图 5-4 所示。

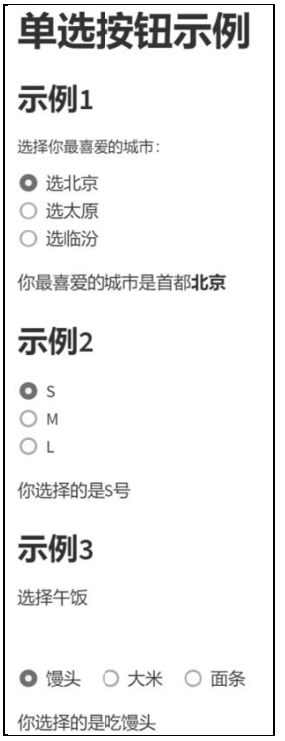

图 5-4　单选按钮示例

5.3　复选框

复选框用来实现“选中”和“取消选中”功能。在 Streamlit 中通过 st.checkbox()方法实现复选框，具体的使用说明如下：

```
    st.checkbox(label, value=False, key=None, help=None, on_change=None, args=
None, kwargs=None, *, disabled=False, label_visibility="visible")
```

（1）label：向用户解释此复选框用途的简短标签。该参数的类型为字符串类型，该参数支持部分 Markdown 语法，如粗体、斜体、删除线、内联代码和表情符号。

（2）value：是否首次渲染时预选复选框。默认值为 False。

（3）key：用于生成唯一标记该组件的键，避免与同类组件混淆。该参数的类型为字符串、整型或 None，默认为 None，将自动生成。

（4）help：可选的工具提示，显示在 label 参数文本的右侧，默认为 None，表示没有工具提示。

（5）on_change：设置该组件选项变化时的回调函数或方法，默认为 None。

（6）args：传递给 on_change 参数对应回调函数或方法的位置参数元组，默认为 None。

（7）kwargs：传递给 on_change 参数对应回调函数或方法的关键字参数字典，默认为 None。

（8）disabled：是否将该组件设置为禁用状态。默认值为 False，是可用状态。该参数只能通过关键字设置。

（9）horizontal：设置是否水平方向排列选项。默认值为 False，即垂直方向排列。该参数只能通过关键字设置。

（10）label_visibility：设置 label 参数的可见性。如果为 hidden，则标签不会显示，但小部件上方仍然有空白空间（相当于 label=""）。如果为 collapsed，则标签和空间都会被删除。默认为 visible。该参数只能通过关键字设置。

（11）返回值：复选框是否被选中。

新建一个名为 checkbox.py 的 Python 文件，整体的思路是通过 3 个示例简单地介绍如何使用 st.checkbox()方法。第 1 个示例为询问用户是否同意我们的用户协议，第 2 个示例同时设置选中和未选中的结果，第 3 个示例询问用户的爱好并将“游泳”设置为预先选中状态。整个文件中的代码如下：

```
#第 5 章/checkbox.py
import streamlit as st

st.header("复选框示例")
st.subheader("示例 1")
agree = st.checkbox('是否同意我们的用户协议')

if agree:
    st.write('是的，我同意')

st.subheader("示例 2")
result = st.checkbox('请勾选我')

if result:
    st.write('非常好！你勾选了')
else:
    st.write('快勾选上方的复选框')

st.subheader("示例 3")
st.write('选择你的爱好')
check_1 = st.checkbox('游泳', value=True)
check_2 = st.checkbox('唱歌')
check_3 = st.checkbox('看电影')
```

效果如图 5-5 所示。

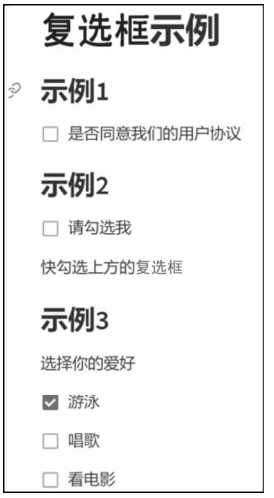

图 5-5　复选框示例

5.4　下拉按钮


11min

下拉按钮允许用户从下拉列表的多个选项中选择一个。在 Streamlit 中通过 st.selectbox() 方法实现下拉按钮，具体的使用说明如下：

```
    st.selectbox(label, options, index=0, format_func=special_internal_function,
key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False,
label_visibility="visible")
```

（1）label：向用户解释此按钮用途的简短标签。该参数的类型为字符串类型，该参数支持部分 Markdown 语法，如粗体、斜体、删除线、内联代码和表情符号。

（2）options：设置下拉按钮的选项。该参数的类型为序列类型，如列表、numpy.ndarray、pandas.Series、pandas.DataFrame 或 pandas.Index 等，如果是 pandas.DataFrame 类型，则选择第 1 列。默认会将这些序列类型中的每个值转换成字符串类型。

（3）index：第 1 次渲染时，预选的选项的索引，默认为 0，即第 1 个选项会被预选。

（4）format_func：对选项进行格式化的函数。它接收原始选项作为参数，并输出该选项显示的标签。对整个方法的返回值没有任何影响。默认为 special_internal_function 函数，实际就是 str()函数。

（5）key：用于生成唯一标记该组件的键，避免与同类组件混淆。该参数的类型为字符串、整型或 None，默认为 None，将自动生成。

（6）help：可选的工具提示，显示在文本的右侧，默认为 None，表示没有工具提示。

（7）on_change：设置该组件选项变化时的回调函数或方法，默认为 None。

（8）args：传递给 on_change 参数对应回调函数或方法的位置参数元组，默认为 None。

（9）kwargs：传递给 on_change 参数对应回调函数或方法的关键字参数字典，默认为

None。

（10）disabled：是否将该组件设置为禁用状态。默认值为 False，是可用状态。该参数只能通过关键字设置。

（11）label_visibility：设置 label 参数的可见性。如果为 hidden，则标签不会显示，但小部件上方仍然有空白空间（相当于 label=""）。如果为 collapsed，则标签和空间都会被删除。默认为 visible。该参数只能通过关键字设置。

（12）返回值：被选中的选项。

新建一个名为 selectbox.py 的 Python 文件，整体的思路是通过 3 个示例说明设置不同参数的效果。第 1 个示例设置自定义 format_func 函数，选项的显示将不同于原始选项，在每个选项的后边增加“市”，并且不会影响返回值，然后将 index 参数设置为 2，第 1 次渲染的预选选项为临汾。第 2 个示例设置 label_visibility 参数，label 参数不可见且不占位置。第 3 个示例设置 label_visibility，label 参数不可见但仍有位置。整个文件中的代码如下：

```
    #第 5 章/selectbox.py
    import streamlit as st
    #自定义一个名为 my_format_func 的函数
    def my_format_func(option):
        return f'{option}市'

    st.header('下拉按钮示例')
    st.subheader('示例 1')
    city = st.selectbox('选择你的故乡：', ['北京', '太原', '临汾'], format_func=
my_format_func, index=2)
    #根据返回值的不同选择不同的特色回答
    #同时应注意返回值不受自定义 my_format_func 函数的影响
    if city == '北京':
        st.write('你的故乡是首都**北京**')
    elif city == '太原':
        st.write('你的故乡是**太原**，它是山西的省会')
    else:
        st.write('你的故乡是**临汾**，古时称“平阳”')

    st.subheader('示例 2')
    size = st.selectbox(
        '选择尺码',
        ['S', 'M', 'L'],
        label_visibility='collapsed'
    )
    #没有特色回答，可直接使用 f-strings 的回答
    st.write(f'你选择的是{size}号')
```

```
st.subheader('示例 3')
st.write('选择午饭')
#将标签设置为"hidden"
#设置水平排列
lunch = st.selectbox(
    '你中午想吃什么菜？',
    ['糖醋里脊', '北京烤鸭', '宫保鸡丁'],
    label_visibility='hidden'
)
st.write(f'你选择的是吃{lunch}')
```

效果如图 5-6 所示。

图 5-6　下拉按钮示例

5.5　多选下拉按钮


11min

多选下拉按钮允许用户从下拉列表的多个选项中选择一个或多个。在 Streamlit 中可以通过 st.multiselect()方法实现多选下拉按钮，具体的使用说明如下：

```
st.multiselect(label, options, default=None, format_func=special_internal_function, key=None, help=None, on_change=None, args=None, kwargs=None, *, disabled=False, label_visibility="visible", max_selections=None)
```

（1）label：向用户解释此按钮用途的简短标签。该参数的类型为字符串类型，该参数支持部分 Markdown 语法，如粗体、斜体、删除线、内联代码和表情符号。

（2）options：设置多选下拉按钮的选项。该参数的类型为序列类型，如列表、numpy.ndarray、pandas.Series、pandas.DataFrame 或 pandas.Index 等，如果是 pandas.DataFrame 类

型，则选择第 1 列。默认会将这些序列类型中的每个值转换成字符串类型。

（3）default：设置默认选中的值，默认为 None，无默认选中的值，可以是单个值，也可以是多个值组成的列表。

（4）format_func：对选项进行格式化的函数。它接收原始选项作为参数，并输出该选项显示的标签。对整个方法的返回值没有任何影响，默认为 special_internal_function 函数，实际就是 str()函数。

（5）key：用于生成唯一标记该组件的键，避免与同类组件混淆。该参数的类型为字符串、整型或 None，默认为 None，将自动生成。

（6）help：可选的工具提示，显示在文本的右侧，默认为 None，表示没有工具提示。

（7）on_change：设置该组件选项变化时的回调函数或方法，默认为 None。

（8）args：传递给 on_change 参数对应回调函数或方法的位置参数元组，默认为 None。

（9）kwargs：传递给 on_change 参数对应回调函数或方法的关键字参数字典，默认为 None。

（10）disabled：是否将该组件设置为禁用状态。默认值为 False，是可用状态。该参数只能通过关键字设置。

（11）label_visibility：设置 label 参数的可见性。如果为 hidden，则标签不会显示，但小部件上方仍然有空白空间（相当于 label=""）。如果为 collapsed，则标签和空间都会被删除。默认为 visible。该参数只能通过关键字设置。

（12）max_selections：设置可选中的最大个数，默认为 None，不限制最大选中个数。该参数只能通过关键字设置。

（13）返回值：被选中的选项。结果的数据类型为列表类型。

新建一个名为 multiselect.py 的 Python 文件，整体的思路是通过 3 个示例说明设置不同参数的效果。第 1 个示例设置 default 参数，预选选中“北京”和“临汾”，设置自定义 my_format_func 函数，选项的显示将不同于原始选项，在每个选项的后边增加“市”，并且不会影响返回值。第 2 个示例设置 max_selections 参数，将可选中的最大个数设置为两个。第 3 个示例设置 disabled 参数，该多选下拉按钮为禁用状态。整个文件中的代码如下：

```
#第 5 章/multiselect.py
import streamlit as st

st.header('多选下拉按钮示例')
st.subheader('示例 1')
#自定义 my_format_func 函数
def my_format_func(option):
    return f'{option}市'

options_1 = st.multiselect(
    '选择你去过的城市',
```