Case Study I

Julian Huber & Matthias Panny

Einführung in `streamlit`

Wir wollen unsere Web-UI mittels `streamlit` implementieren

Einführung in `streamlit`

Tipp: Zeichen Sie ein Mockup des User Interfaces

- Wir werden im Beispiel streamlit nutzen, um das User-Interface zu bauen. Schauen Sie sich also vorab die Beispiele an, um ein Gefühl zu bekommen was möglich ist
- Basierend auf den Anwendungsfälle und den Daten:
  - Welche Eingaben gibt es?
  - Welche Anzeigen gibt es?

Einführung in `streamlit`

- Installation → `pip install streamlit` in einer neuen venv
- Starten von `streamlit` `python -m streamlit run main.py` bzw. `streamlit run main.py`

Funktionsweise von `streamlit`

- Wir schreiben ein Python-Skript, das die User-Interface Elemente definiert
- Ein Server wird im Hintergrund gestartet und stellt unser User-Interface dar
- Änderungen werden übernommen, sobald wir die `main.py` (siehe nächste Folie) speichern
- Eingaben auf dem User-Interface werden automatisch berücksichtigt:
  Bei jede Interaktion läuft das Python Skript von oben nach unten einmal durch

Erste Streamlit App

- Legen sie eine Datei `main.py` an und fügen Sie den folgenden Code ein:
- Die Felder einer Selectbox finden Sie in der API-Referenz

Beispiel in `main.py`

```
import streamlit as st

# Eine Überschrift der ersten Ebene
st.write("# Gerätemanagement")

# Eine Überschrift der zweiten Ebene
st.write("## Geräteauswahl")

# Eine Auswahlbox mit hard-gecoded Optionen, das Ergebnis

current_device = st.selectbox(label='Gerät auswählen',
        options = ["Gerät_A", "Gerät_B"])

st.write(F"Das ausgewählte Gerät ist {current_device}")
```

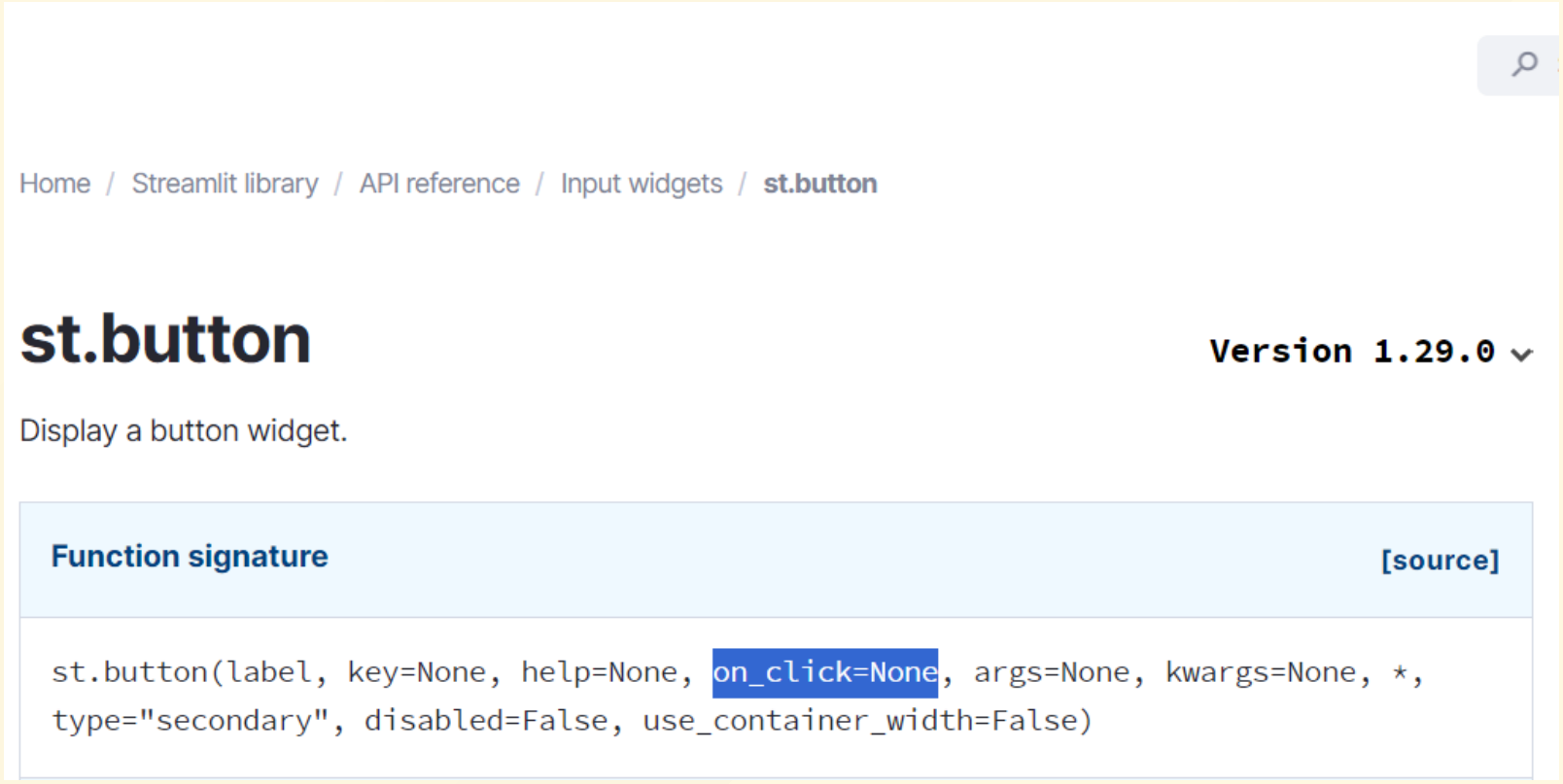
Session States

- Speichern Variablenwerte während der Sessions eines Users (Browser Tabs)
- Auch normale Variablen werden gespeichert, die Session States gelten aber für genau eine Session
- englischsprachiges Erklärvideo

Beispiel in `main_session_state.py`

```python
import streamlit as st

if "sb_current_device" not in st.session_state:
    st.session_state.sb_current_device = ""

# Eine Überschrift der ersten Ebene
st.write("# Gerätemanagement")

# Eine Überschrift der zweiten Ebene
st.write("## Geräteauswahl")

# Eine Auswahlbox mit hard-gecoded Optionen, das Ergebnis

st.session_state.sb_current_device = st.selectbox(label='Gerät auswählen',
        options = ["Gerät_A", "Gerät_B"])

st.write(F"Das ausgewählte Gerät ist {st.session_state.sb_current_device}")
```

Callbacks bei UI-Elementen



Home / Streamlit library / API reference / Input widgets / **st.button**

st.button

Version 1.29.0

Display a button widget.

Function signature [source]

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

Verhalten von UI-Elementen in `callback.py`

- Die Interaktion mit den UI-Elementen soll eine Funktion auslösen
- `st.button` hat `on_click`-Parameter
- `st.selectbox` hat `on_change`-Parameter
- Alle Callback-Parameter übernehmen ein `callable` (Funktion, die aufgerufen werden kann)

Callbacks bei UI-Elementen

- Callbacks müssen definiert oder importiert werden, bevor sie aufgerufen werden
- sie übernehmen keine Argumente, sondern greifen auf die Session States zu
- in diesem Fall werden die Session States `price_euro` direkt mit dem `number_input`-Element verknüpft, so dass der Wert in beide Richtungen aktualisiert wird

Callsbacks in `main_callbacks.py`

```
import streamlit as st
st.write("## Wartungskosten")

# Callbacks for the number inputs
def update_price_from_euro():
    st.session_state.price_dollar = st.session_state.price_euro * 1.1

def update_price_from_dollar():
    st.session_state.price_euro = st.session_state.price_dollar * 0.9

# Number inputs for the maintenance costs
st.number_input(label="Wartungskosten in Euro",
                            key = "price_euro",
                            on_change=update_price_from_euro)

st.number_input(label="Wartungskosten in Dollar",
                            key = "price_dollar",
                            on_change=update_price_from_dollar)
```

Design

- `streamlit` bietet viele Möglichkeiten an um das User-Interface zu gestalten
- Von besonderer Bedeutung sind höchstwahrscheinlich:
  - die Layout Elemente
  - die Control Flow Elemente → insbesondere `streamlit.form`
  - die Input Elemente

Beispiel Columns in columns.py

```
import streamlit as st

col1, col2 = st.columns(2)

with col1:
   st.header("A cat")
   st.image("https://static.streamlit.io/examples/cat.jpg")

with col2:
   st.header("A dog")
   st.image("https://static.streamlit.io/examples/dog.jpg")
```

Beispiel Tabs in tabs.py

```
import streamlit as st

tab1, tab2, tab3 = st.tabs(["Cat", "Dog", "Owl"])

with tab1:
   st.header("A cat")
   st.image("https://static.streamlit.io/examples/cat.jpg", width=200)

with tab2:
   st.header("A dog")
   st.image("https://static.streamlit.io/examples/dog.jpg", width=200)

with tab3:
   st.header("An owl")
   st.image("https://static.streamlit.io/examples/owl.jpg", width=200)
```

Modularer Aufbau

Beispiel in `modular.py`

- Es kann hilfreich sein das ganze User Interface in einzelne Module aufzuteilen, damit es beherrschbar bleibt
- Dazu wird nur die Datei `modular.py` gestartet
- Diese Datei ruft dann die `run()` Funktionen der einzelnen Module für jeder Seite, z.B. in `tab1.py` auf

🏆 Bauen Sie Ihr Mock-Up nach

- Machen Sie sich mit den Elementen der API reference vertraut
- Bauen Sie sich die Struktur ihrer App nach
- Sie sollen sich noch keine Gedanken, um die dynamischen Prozesse machen
- Wenn Sie dynamische Inhalte anzeigen wollen, definieren Sie eine Variable im Session State und füllen Sie mit einem festen Platzhalter-Wert

```
if 'current_device' not in st.session_state:
    st.session_state.current_device = 'Der Gerät'
```