第7回 Streamlit基本(3)

レイアウト、状態管理、ファイル入力

出席認証コード: **2242**

授業資料: https://x.gd/NoqkC (←実際のURLに置き換えてください)

本日の目標

1. **レイアウト機能**: `st.columns`, `st.expander`, `st.sidebar` を理解し、使いこなせるようになる。
2. **状態管理**: `st.session_state` の概念と使い方を習得する。
3. **フォーム機能**: `st.form`, `st.form_submit_button` を理解し、複数の入力をまとめて扱えるようになる。

タイムスケジュール (目安)

時間	内容
00:00 - 00:05	前回の復習と本日の目標
00:05 - 00:25	レイアウト機能
00:25 - 00:45	状態管理 (`st.session_state`)
00:45 - 01:05	フォーム (`st.form`)
01:05 - 01:20	演習課題と実践
01:20 - 01:25	質疑応答と次回予告
01:25 - 01:30	まとめ

1. レイアウト機能 (1) - `st.sidebar`

- アプリケーションの主要なコンテンツから分離してコントロールを配置
- ナビゲーションや設定項目に適しています

```
import streamlit as st

# サイドバーに要素を追加
add_selectbox = st.sidebar.selectbox(
    "連絡方法を選択してください",
    ("メール", "携帯電話", "LINE")
)

st.sidebar.write(f"選択された連絡方法: {add_selectbox}")

st.write("メインコンテンツエリア")
# (演習: src/lecture07/app_lecture.py の該当箇所に各自追記)
```

1. レイアウト機能 (2) - `st.columns`

- 画面を複数の列に分割してコンテンツを配置
- `st.columns(N)` で N 個の列を作成
- `st.columns([ratio1, ratio2, ...])` で列の幅の比率を指定可能

```
import streamlit as st

col1, col2, col3 = st.columns(3)

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

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

with col3:
   st.header("Owl")
   st.image("https://static.streamlit.io/examples/owl.jpg")
```

1. レイアウト機能 (3) - `st.expander`

- クリックでコンテンツの表示/非表示を切り替えられるコンテナ
- 詳細情報や補助的な設定を隠すのに便利

```
import streamlit as st

st.bar_chart({"data": [1, 5, 2, 6, 2, 1]})

with st.expander("詳細を見る"):
     st.write(\'\'\'
          このグラフは架空のデータを表示しています。
          詳細な分析やデータの背景については、提供元にお問い合わせください。
     \'\'\')
```

レイアウト機能 演習

- **目標**: `st.sidebar`, `st.columns`, `st.expander` を使って簡単なレイアウトを作成する。
- **演習ファイル**: `src/lecture07/exercises/section1_layout_template.py` を編集
- **課題例**:
  i. サイドバーに自分の名前と学籍番号を表示するテキスト入力を配置。
  ii. メインエリアを2列に分け、左列に好きなものの画像、右列にその説明文を表示。
  iii. 説明文の下にエキスパンダーを設け、さらに詳細な情報を隠して表示できるようにする。

2. 状態管理 `st.session_state` (1)

- Streamlitアプリはウィジェット操作のたびにスクリプトが再実行され、変数がリセットされる。
- `st.session_state`: ユーザーのインタラクション間でデータを保持するための仕組み。
  - 辞書のようなオブジェクト。
  - キーを使って値の読み書きが可能。

```
import streamlit as st

# st.session_state に \'count\' がなければ初期化
if \'count\' not in st.session_state:
    st.session_state.count = 0

increment = st.button("カウントアップ")
if increment:
    st.session_state.count += 1

st.write("カウント: ", st.session_state.count)
# (演習: src/lecture07/app_lecture.py の該当箇所に各自追記)
```

`st.session_state` 演習

- **目標**: `st.session_state` を使って、インタラクション間で状態を保持するアプリを作成する。
- **演習ファイル**: `src/lecture07/exercises/section2_session_state_template.py` を編集
- **課題例**: 今日の気分記録アプリを作成
    i. 気分を選択するボタンを複数作成（😊 嬉しい、😢 悲しい、😴 眠い、🍕 お腹すいた など）
    ii. ボタンを押すと、選択した気分が `st.session_state` のリストに追加される
    iii. 今までに記録した気分の履歴を表示する
    iv. (発展) 各気分の回数をカウントして表示する

3. フォーム `st.form` (1)

フォームとは

- ネットショッピングの注文画面のような仕組み
- 名前、住所、連絡先などを全部入力してから「注文する」ボタンを押す流れ
- Streamlitでも同じように、複数の項目をまとめて送信できる機能

基本の使い方:

- `st.form("フォーム名")` でフォームの範囲を作成
- フォーム内では、項目を入力しても画面が自動更新されない
- `st.form_submit_button("送信")` で送信ボタンを作成

```
import streamlit as st

with st.form("cafe_order"):
    st.write("カフェで注文してみよう")
    name = st.text_input("お名前")
    drink = st.selectbox("ドリンク", ["コーヒー", "紅茶", "ココア"])

    # 送信ボタン
    submitted = st.form_submit_button("注文する")

    if submitted:
```

3. フォーム `st.form` (2) - メリット

通常の入力とフォーム入力の違い

通常の入力の場合:

- 名前を入力 → 画面が更新される
- 年齢を入力 → 再び画面が更新される
- 住所を入力 → また画面が更新される

フォームを使った場合:

- 名前、年齢、住所を全て入力完了
- 「送信」ボタンを押して一度だけ画面が更新される

メリット:

- 入力途中での不要な画面更新を防ぐ
- 全項目入力後に一括確認が可能
- サークル申し込みフォームのような快適な操作感

`st.form` 演習

- **目標**: `st.form` を使って、サークル入会申し込みフォームを作成する
- **演習ファイル**: `src/lecture07/exercises/section3_form_template.py` を編集
- **課題例**: サークル入会申し込みフォーム
    i. 基本情報の入力欄を作成
        - お名前 (テキスト入力)
        - 学年 (セレクトボックス: 1年生、2年生、3年生、4年生)
        - 好きな活動 (セレクトボックス: 文化祭、合宿、勉強会、交流会など)
        - 意気込み (テキストエリア)
    ii. 「申し込む」ボタンを設置
    iii. 送信ボタンが押されたら、入力された情報をまとめて表示する

まとめ

- **レイアウト**:
  - `st.sidebar`: サイドバーにウィジェットを配置
  - `st.columns`: 画面を列に分割
  - `st.expander`: コンテンツを折りたたみ表示
- **状態管理**:
  - `st.session_state`: インタラクション間でデータを保持
- **フォーム**:
  - `st.form` & `st.form_submit_button`: 複数入力をまとめて送信

これらの機能を組み合わせることで、より複雑でインタラクティブなWebアプリが作成できます！

次回予告

第8回: 課題演習 (1)

- これまでの知識を活用した総合的なStreamlitアプリ作成演習
- 成績評価の対象に入れるので可能であれば出席してください！

アプリのデプロイ

Streamlit Community Cloud を利用してアプリを公開する手順を確認します。

- GitHub の public レポジトリーの作成
- アプリファイルの追加とコミット（GitHub GUI または Git）
- Streamlit Community Cloud への登録
- GitHub リポジトリーを指定してアプリをデプロイ

参考資料