第7回 Streamlit基本(3)

レイアウト、状態管理、ファイル入力

担当: [担当者名] 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。
4. **ファイルアップロード**: `st.file_uploader` を用いて、ユーザーがファイルをアップロードし、その情報を表示できるようになる。

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	ファイルアップロード (`st.file_uploader`)
01:20 - 01:25	質疑応答と次回予告
01:25 - 01:30	まとめと演習課題の提示

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")

# (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 (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。
     \'\'\')
# (演習: src/lecture07/app_lecture.py の該当箇所に各自追記)
```

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/app_lecture.py` の「レイアウト機能 演習セクション」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)
```

2. 状態管理 `st.session_state` (2) - 初期化とコールバック

- **初期化のタイミング**:
  - 通常、キーが存在しない場合に初期値を代入。
- **コールバック関数**:
  - ウィジェットの `on_change` 引数に関数を指定すると、ウィジェットの値が変更されたときにその関数が実行される。
  - コールバック関数内で `st.session_state` を更新することで、より複雑な状態管理が可能。

```
import streamlit as st

def count_up():
    st.session_state.counter += 1

if \'counter\' not in st.session_state:
```

`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/app_lecture.py` の「状態管理 (`st.session_state`) 演習セクション」を編集
- **課題例**:
  i. テキスト入力欄と追加ボタンを作成。
  ii. ボタンを押すと、入力されたテキストがリストに追加され、`st.session_state` に保存されるようにする。
  iii. 保存されたリストを画面に表示する。
  iv. (発展) 各リストアイテムに削除ボタンを付け、クリックするとそのアイテムがリストから削除されるようにする。

3. フォーム `st.form` (1)

- 複数の入力ウィジェットをグループ化し、一度に送信するための仕組み。
- `st.form("my_form")` でフォームブロックを開始。
- フォーム内のウィジェットは、送信ボタンが押されるまで再実行をトリガーしない。
- `st.form_submit_button("送信")` で送信ボタンを作成。

```
import streamlit as st

with st.form("user_form"):
    st.write("ユーザー情報を入力してください")
    name = st.text_input("名前")
    age = st.number_input("年齢", min_value=0, max_value=120)

    # フォーム送信ボタン
    submitted = st.form_submit_button("登録")

    if submitted:
        st.write(f"名前: {name}, 年齢: {age} で登録しました。")
# (演習: src/lecture07/app_lecture.py の該当箇所に各自追記)
```

3. フォーム `st.form` (2) - メリット

- **意図しない再実行の防止**:
  - フォーム内のウィジェットを操作しても、送信ボタンが押されるまではアプリ全体が再実行されない。
  - ユーザーが複数の項目を入力し終えてから処理を行いたい場合に有効。
- **UXの向上**:
  - 多数の入力項目がある場合、入力途中で画面が何度もリフレッシュされるのを防ぐ。

`st.form` 演習

- **目標**: `st.form` を使って、複数の入力をまとめて処理するアプリを作成する。
- **演習ファイル**: `src/lecture07/app_lecture.py` の「フォーム (`st.form`) 演習セクション」を編集
- **課題例**:
    i. アンケートフォームを作成する。
        - 名前 (テキスト入力)
        - メールアドレス (テキスト入力)
        - 好きな食べ物 (セレクトボックス: 寿司、ラーメン、カレーなど)
        - 意見 (テキストエリア)
    ii. 「送信」ボタンを設置。
    iii. 送信ボタンが押されたら、入力された全情報をまとめて表示する。

4. ファイルアップロード `st.file_uploader` (1)

- ユーザーがローカルからファイルをアップロードできるようにするウィジェット。
- `st.file_uploader("ラベル", type=["拡張子1", "拡張子2"])`
  - `type` で許可するファイル形式を指定可能 (例: `type=["csv", "txt"]`, `type="png"`)。
- アップロードされたファイルは `UploadedFile` オブジェクトとして返される。
  - `None` (ファイル未選択時) または `UploadedFile` インスタンス。

```
import streamlit as st

uploaded_file = st.file_uploader("ファイルを選択してください (PNG or JPG)", type=["png", "jpg"])

if uploaded_file is not None:
    # ファイル詳細の表示
    st.write("ファイル名:", uploaded_file.name)
    st.write("ファイルタイプ:", uploaded_file.type)
    st.write("ファイルサイズ:", uploaded_file.size, "bytes")

    # 画像ファイルなら表示
```

4. ファイルアップロード `st.file_uploader` (2) - `UploadedFile` オブジェクト

`UploadedFile` オブジェクトから取得できる主な情報:

- `name`: アップロードされたファイルの名前 (例: `my_image.png`)
- `type`: ファイルのMIMEタイプ (例: `image/png`)
- `size`: ファイルサイズ (バイト単位)
- `read()`: ファイルの内容をバイト列として読み込む
- `getvalue()`: `read()` と同様 (BytesIOの場合)

```
# (前スライドのコードの続き)
# テキストファイルの場合、内容を表示
if uploaded_file is not None and uploaded_file.type == "text/plain":
    # バイト列を文字列にデコード
    string_data = uploaded_file.read().decode("utf-8") # または適切なエンコーディング
    st.text_area("ファイル内容", string_data, height=200)
```

`st.file_uploader` 演習

- **目標**: `st.file_uploader` を使ってファイルをアップロードし、その情報を表示するアプリを作成する。
- **演習ファイル**: `src/lecture07/app_lecture.py` の「ファイルアップロード (`st.file_uploader`) 演習セクション」を編集
- **課題例**:
  i. CSVファイルをアップロードできるようにする (`type=["csv"]`)。
  ii. アップロードされたら、ファイル名、ファイルタイプ、ファイルサイズを表示する。
  iii. (発展) アップロードされたCSVファイルの内容をPandas DataFrameとして読み込み、最初の5行を `st.dataframe` で表示する。(Pandasの知識が必要になります。ヒント: `pd.read_csv(uploaded_file)`)

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
- **ファイルアップロード**:
  - `st.file_uploader`: ローカルファイルをアプリにアップロード

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！

質疑応答

何か質問はありますか？

次回予告

第8回: 課題演習 (1) と GitHub

- これまでの知識を活用した総合的なStreamlitアプリ作成演習
- 作成したアプリをGitHubへアップロードする方法
- 補足: `st.image`, `st.video` の使い方

準備しておくこと:

- GitHubアカウント
- これまでの復習

お疲れ様でした！