

掲示板を作ろう

この教材でできること

- htmlの連携ができる
- DBとのやり取りができる

どんな教材？

DBに入っている内容をweb上で出力したり、編集したりできるようになる。

目次

- ① コードを書く前の準備をしよう
- ② 完成系を確認してみよう
- ③ コードを書いてみよう
- ④ 答え合わせをしてみよう

目次

- ① コードを書く前の準備をしよう
- ② 完成系を確認してみよう
- ③ コードを書いてみよう
- ④ 答え合わせをしてみよう

① コードを書く前の準備をしよう

このページでは環境設定などの準備をするためのページ

① コードを書く前の準備をしよう

このページでは掲示板の情報を保存するDBを準備する



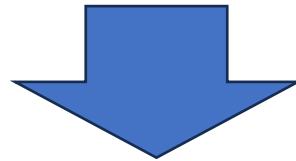
The screenshot shows a database management tool interface. At the top, there is a menu bar with icons and labels for '表示' (Display), '構造' (Structure), 'SQL', '検索' (Search), '挿入' (Insert), 'エクスポート' (Export), 'インポート' (Import), '権限' (Permissions), and '操作' (Operations). Below the menu bar, there are two tabs: 'テーブルの構造' (Table Structure) and 'リレーションビュー' (Relation View). The 'テーブルの構造' tab is active, displaying a table structure for a table with four columns: #, 名前 (Name), タイプ (Type), 照合順序 (Collation), 属性 (Attributes), Null, デフォルト値 (Default Value), コメント (Comment), その他 (Other), and 操作 (Operations). The table structure is as follows:

#	名前	タイプ	照合順序	属性	Null	デフォルト値	コメント	その他	操作
<input type="checkbox"/>	1 id	int(5)			いいえ	なし			変更 削除 その他
<input type="checkbox"/>	2 title	varchar(30)	utf8mb4_general_ci		いいえ	なし			変更 削除 その他
<input type="checkbox"/>	3 content	varchar(50)	utf8mb4_general_ci		いいえ	なし			変更 削除 その他
<input type="checkbox"/>	4 created	date			いいえ	なし			変更 削除 その他

① コードを書く前の準備をしよう

このページでは環境設定や使うファイル、ライブラリなどの準備を行う

```
CondaError: KeyboardInterrupt
(base) PS C:\Users\小西 翔\python\python14> conda create --name board python=3.9
Retrieving notices: ...working... DEBUG:urlllib3.connectionpool:Starting new HTTPS
```



```
# ...$ conda deactivate
(base) PS C:\Users\小西 翔\python\python14> conda activate board
(board) PS C:\Users\小西 翔\python\python14>
```

① コードを書く前の準備をしよう

ライブラリは必要であればその都度追加しよう！

vscodeを開いて、コードを書いていこう！

目次

① コードを書く前の準備をしよう

② 完成系を確認してみよう

③ コードを書いてみよう

④ 答え合わせをしてみよう

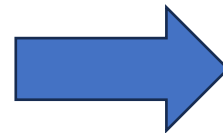
②完成系を確認してみよう

このページでは完成系を見てどのようなコードを書いたらいいかをイメージしてみよう

②完成系を確認してみよう

httpの後の文字をコピーしてwebに張り付けてエンターキーを押す

```
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use this for production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 399-545-312
```



②完成系を確認してみよう

htmlで投稿できるページにする

掲示板

投稿一覧

[初めての投稿](#)
掲示板を作りました
2024-10-25 00:00

②完成系を確認してみよう

タイトルと内容を入力して投稿ボタンを押す

The screenshot shows a web interface for creating a post. At the top, there is a header labeled "掲示板" (Bulletin Board). Below this, there is a form with two text input fields. The first field contains the text "テスト投稿" (Test Post), and the second field contains "テストのための投稿" (Post for testing). Below the input fields is a blue button labeled "投稿" (Post). Underneath the form is a section titled "投稿一覧" (Post List). This section contains a single post entry with the title "初めての投稿" (First Post), the content "掲示板を作りました" (I made a bulletin board), and the timestamp "2024-10-25 00:00".

②完成系を確認してみよう

新しい投稿が下に出てくるようにする

掲示板

タイトル

内容

投稿

投稿一覧

初めての投稿
掲示板を作りました
2024-10-25 00:00

テスト投稿
テストのための投稿
2024-10-25 00:00

目次

- ① コードを書く前の準備をしよう
- ② 完成系を確認してみよう
- ③ コードを書いてみよう
- ④ 答え合わせをしてみよう

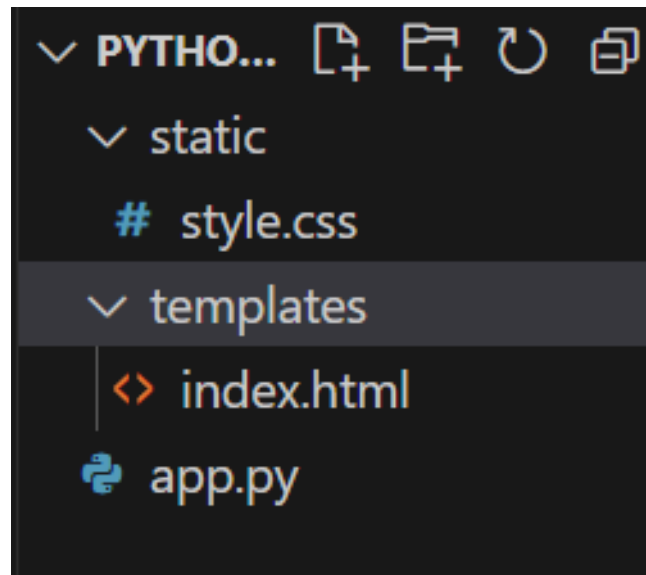
③コードを書いてみよう

このページではどのようなコードを書いたらいいかヒントが書かれていますよ

③コードを書いてみよう

ファイル、フォルダーは写真のようになっているよ

staticの中にcssコード、templatesの中にhtmlコードが入っているよ



③コードを書いてみよう

htmlのコードはまずformタグを作り、投稿のタイトル、内容とボタンを作ろう

③コードを書いてみよう

次に投稿一覧でPythonの内容から送られた変数をhtmlで表示するようにする

③コードを書いてみよう

PythonではDBとの連携をする

③コードを書いてみよう

DBの内容を読み取り、変数にしてhtmlに送る

③コードを書いてみよう

POSTで送られてきた内容からDBに保存する

③コードを書いてみよう

デザインが気になったら自分でcssのコードを書いてみよう

目次

- ① コードを書く前の準備をしよう
- ② 完成系を確認してみよう
- ③ コードを書いてみよう
- ④ 答え合わせをしてみよう

④答え合わせをしてみよう

答えのコードを確認してどのようにしているか見れるよ

④ 答え合わせをしてみよう

htmlのヘッドの部分と入力部分を書いていこう

```
templates > index.html > ...
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>掲示板</title>
7   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
8 </head>
9 <body>
10  <h1>掲示板</h1>
11
12  <form action="/add" method="POST">
13    <input type="text" name="title" placeholder="タイトル" required>
14    <textarea name="content" placeholder="内容" required></textarea>
15    <button type="submit">投稿</button>
16  </form>
17
18  <h2>投稿一覧</h2>
19  <ul>
20    {% for post in posts %}
21    <li>
22      <h3>{{ post.title }}</h3>
23      <p>{{ post.content }}</p>
24      <small>{{ post.created.strftime('%Y-%m-%d %H:%M') }}</small>
25    </li>
26    {% endfor %}
27  </ul>
28 </body>
29 </html>
```

④答え合わせをしてみよう

ライブラリを入力しよう

必要であればその都度インストールをしよう

```
app.py > ...  
1   from flask import Flask, render_template, request, redirect  
2   import pymysql  
3   from datetime import datetime  
4
```

④答え合わせをしてみよう

DBの連携をしよう

```
4
5  app = Flask(__name__)
6
7  # データベース接続設定
8  def get_db_connection():
9      connection = pymysql.connect(
10         host='localhost',
11         user='root',
12         password='', # パスワードが設定されている場合はここに記入
13         database='python_test',
14         charset='utf8mb4',
15         cursorclass=pymysql.cursors.DictCursor
16     )
17     return connection
```

④答え合わせをしてみよう

DBの内容を読み取り、変数に代入しよう

```
18
19 # 投稿一覧の表示
20 @app.route('/')
21 def index():
22     connection = get_db_connection()
23     try:
24         with connection.cursor() as cursor:
25             cursor.execute("SELECT * FROM board ORDER BY created DESC")
26             posts = cursor.fetchall()
27             return render_template('index.html', posts=posts)
28     finally:
29         connection.close()
```

④ 答え合わせをしてみよう

投稿の内容をDBに保存しよう

```
30
31 # 新規投稿の追加
32 @app.route('/add', methods=['POST'])
33 def add_post():
34     title = request.form['title']
35     content = request.form['content']
36     created = datetime.now()
37
38     connection = get_db_connection()
39     try:
40         with connection.cursor() as cursor:
41             sql = "INSERT INTO board (title, content, created) VALUES (%s, %s, %s)"
42             cursor.execute(sql, (title, content, created))
43             connection.commit()
44             return redirect('/')
45     finally:
46         connection.close()
47
48 if __name__ == '__main__':
49     app.run(debug=True)
50
```

④答え合わせをしてみよう

余力があればcssでデザインを作ろう

```
1 body {
2   font-family: Arial, sans-serif;
3   background-color: #f4f4f4;
4   margin: 0;
5   padding: 20px;
6 }
7
8 h1 {
9   text-align: center;
10  color: #333;
11 }
12
13 form {
14   background: #fff;
15   padding: 20px;
16   margin-bottom: 20px;
17   border-radius: 5px;
18   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
19 }
20
21 form input, form textarea {
22   width: 100%;
23   padding: 10px;
24   margin: 10px 0;
25   border: 1px solid #ccc;
26   border-radius: 3px;
27 }
```

```
28
29 form button {
30   background-color: #007BFF;
31   color: white;
32   border: none;
33   padding: 10px 15px;
34   border-radius: 3px;
35   cursor: pointer;
36 }
37
38 form button:hover {
39   background-color: #0056b3;
40 }
41
42 ul {
43   list-style-type: none;
44   padding: 0;
45 }
46
47 ul li {
48   background: #fff;
49   margin-bottom: 15px;
50   padding: 15px;
51   border-radius: 5px;
52   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
53 }
54
```

```
54
55 ul li h3 {
56   margin: 0 0 10px 0;
57   color: #007BFF;
58 }
59
60 ul li p {
61   margin: 0 0 10px 0;
62   color: #333;
63 }
64
65 ul li small {
66   color: #666;
67   font-size: 0.9em;
68 }
69
```

お疲れさまでした

テキストは終了です。
あとは自分なりにアレンジを付け加えていこう！