Una aplicación simple con Flask, Python y MongoDB

Extyraido y traducido de: https://www.c-sharpcorner.com

La intención de realizar este tutorial es la de aplicar todas las herramientas vistas hasta aquí con MongoDB, Python y el framework Flask para desarrollo de aplicaciones web.

Vamos a crear una aplicación web Python simple usando Flask framework y MongoDB. Es muy fácil trabajar con Flask y MongoDB.
Primero verificar que tiene instalado lo siguiente, e instalar si no lo tiene instalado.

Software o librería	Modo de instalación
Python	Version 3.7 o mayor
MongoDB	*Descargar desde* https://www.mongodb.com/download-center/community *e instalar.*
Flask	*pip install Flask*
pymongo	*pip install pymongo*
bson	*pip install bson*

Una vez iniciado spider, crear un nuevo proyecto llamado **Tareas**. debajo de la carpeta del proyecto agregue dos subcarpetas, **static** y **templates**.

Cree un nuevo archivo llamado tareapp.py. Este es el único archivo de Python que vamos a usar en esta aplicación.

Primero, vamos a importar las bibliotecas requeridas en nuestra aplicación.

Primero, vamos a importar las bibliotecas requeridas en nuestra aplicación.

```
from flask import Flask, render_template,request,redirect,url_for
from bson import ObjectId
from pymongo import MongoClient
import os
```

Ahora vamos a declarar la variable de la aplicación en nuestro programa.

```
app = Flask(__name__)
```

Esta variable de aplicación se usa en toda la aplicación.

Ahora vamos a declarar dos títulos y encabezados variables que luego se usarán en la plantilla jinja2.

```
title =  "Aplicación de muestra TODO con Flask y MongoDB"
heading =  "Recordatorio TODO con Flask y MongoDB"
```

Debemos declarar la cadena de conexión para MongoDB y seleccionar una base de datos. Además, vamos a seleccionar el nombre de nuestra colección a una variable.

```
client = MongoClient("mongodb://127.0.0.1:27017")  #host uri
db = client.mymongodb  #Seleccione la base de datos
todos = db.todo #Seleccione el nombre de la coleccion
```

Mongodb se ejecuta en el puerto 27017 de forma predeterminada. Tenga en cuenta que hemos seleccionado **mymongodb** como base de datos y **todo** como nombre de la colección. Cuando creamos nuestra primera transacción, pymongo generará la base de datos y la colección automáticamente.

Ahora se implementa el enrutamiento. Para eso, estamos usando **render_template , request, redirect** y **url_for**. Todos estos métodos nos ayudan a establecer la redirección y mostrar plantillas html en el navegador.

```
def redirect_url():
return request.args.get('next') or \
request.referrer or \
url_for('index')
```

En el código anterior definimos un método llamado redirect_url y se usa para redirigir la página a la página de index.

```
@app.route("/")
@app.route("/uncompleted")
def tasks ():
#Display the Uncompleted Tasks
todos_l = todos.find({"done":"no"})
a2="active"
return render_template('index.html',
    a2=a2,todos=todos_l,t=title,h=heading)
```

En el código anterior, definimos un método llamado tasks y se usa para dos rutas. Una para la ruta predeterminada "/" y otra para la ruta "/uncompleted". En este código definimos una variable todos_l y obtiene los documentos del filtro mongodb por condición done igual a no. Definimos una variable más llamada a2 y se usa para controlar los registros activos. Ambas variables todos_l y a2 pasaron con las otras dos variables, título y encabezado a la plantilla jinja2 index.html que debemos crear en la carpeta templates.

Ahora vamos a terminar todas las definiciones de ruta restantes para agregar y eliminar los documentos a mongodb

1. from flask import Flask, render_template,request,redirect,url_for # For flask implem entation
2. from bson import ObjectId # For ObjectId to work

```
3.  from pymongo import MongoClient
4.  import os
5.
6.  app = Flask(__name__)
7.  title = "TODO sample application with Flask and MongoDB"
8.  heading = "TODO Reminder with Flask and MongoDB"
9.
10. client = MongoClient("mongodb://127.0.0.1:27017") #host uri
11. db = client.mymongodb    #Select the database
12. todos = db.todo #Select the collection name
13.
14. def redirect_url():
15.     return request.args.get('next') or \
16.            request.referrer or \
17.            url_for('index')
18.
19. @app.route("/list")
20. def lists ():
21.     #Display the all Tasks
22.     todos_l = todos.find()
23.     a1="active"
24.     return render_template('index.html',a1=a1,todos=todos_l,t=title,h=heading)
25.
26. @app.route("/")
27. @app.route("/uncompleted")
28. def tasks ():
29.     #Display the Uncompleted Tasks
30.     todos_l = todos.find({"done":"no"})
31.     a2="active"
32.     return render_template('index.html',a2=a2,todos=todos_l,t=title,h=heading)
33.
34.
35. @app.route("/completed")
36. def completed ():
37.     #Display the Completed Tasks
38.     todos_l = todos.find({"done":"yes"})
39.     a3="active"
40.     return render_template('index.html',a3=a3,todos=todos_l,t=title,h=heading)
41.
42. @app.route("/done")
43. def done ():
44.     #Done-or-not ICON
45.     id=request.values.get("_id")
46.     task=todos.find({"_id":ObjectId(id)})
47.     if(task[0]["done"]=="yes"):
48.         todos.update({"_id":ObjectId(id)}, {"$set": {"done":"no"}})
49.     else:
```

```
50.         todos.update({"_id":ObjectId(id)}, {"$set": {"done":"yes"}})
51.     redir=redirect_url()
52.
53.     return redirect(redir)
54.
55. @app.route("/action", methods=['POST'])
56. def action ():
57.     #Adding a Task
58.     name=request.values.get("name")
59.     desc=request.values.get("desc")
60.     date=request.values.get("date")
61.     pr=request.values.get("pr")
62.     todos.insert({ "name":name, "desc":desc, "date":date, "pr":pr, "done":"no"})
63.     return redirect("/list")
64.
65. @app.route("/remove")
66. def remove ():
67.     #Deleting a Task with various references
68.     key=request.values.get("_id")
69.     todos.remove({"_id":ObjectId(key)})
70.     return redirect("/")
71.
72. @app.route("/update")
73. def update ():
74.     id=request.values.get("_id")
75.     task=todos.find({"_id":ObjectId(id)})
76.     return render_template('update.html',tasks=task,h=heading,t=title)
77.
78. @app.route("/action3", methods=['POST'])
79. def action3 ():
80.     #Updating a Task with various references
81.     name=request.values.get("name")
82.     desc=request.values.get("desc")
83.     date=request.values.get("date")
84.     pr=request.values.get("pr")
85.     id=request.values.get("_id")
86.     todos.update({"_id":ObjectId(id)}, {'$set':{ "name":name, "desc":desc, "date":dat
    e, "pr":pr }})
87.     return redirect("/")
88.
89. @app.route("/search", methods=['GET'])
90. def search():
91.     #Searching a Task with various references
92.
93.     key=request.values.get("key")
94.     refer=request.values.get("refer")
95.     if(key=="_id"):
```

```
96.         todos_l = todos.find({refer:ObjectId(key)})
97.     else:
98.         todos_l = todos.find({refer:key})
99.     return render_template('searchlist.html',todos=todos_l,t=title,h=heading)
100.
101.         if __name__ == "__main__":
102.
103.             app.run()
```

Debemos agregar los siguientes tres archivos HTML en la carpeta templates. (index.html, searchlist.html y update.html)

index.html

```
1.  <html>
2.      <head>
3.          <title>{{t}}</title>
4.          <!-- href="/static/assets/style.css"-->
5.          <link rel="stylesheet" type="text/css"  href="{{ url_for('static',filename='assets
    /style.css')}}" >
6.          <link rel="stylesheet" type="text/css"  href="{{ url_for('static',filename='assets
    /emoji.css')}}" >
7.          <script src="{{ url_for('static',filename='assets/twemoji.min.js')}}"></script>

8.          <script src="{{ url_for('static',filename='assets/emoji.js')}}"></script>
9.      </head>
10. <body>
11.     <h1>{{ h }}</h1>
12.     <ul>
13.         <li><a href="/list" class="{{ a1 }}">ALL</a></li>
14.         <li><a href="/" class="{{ a2 }}">Uncompleted</a></li>
15.         <li><a href="/completed" class="{{ a3 }}">Completed</a></li>
16.     </ul>
17.     <hr>
18.     {% if todos[0] %}
19.     <div span="right">
20.     <form action="/search"   method="GET" >
21.         <table class="none" id="close">
22.         <tr>
23.         <td></td><td></td>
24.         <td><big><b>Search Reference:</b></big></td>
25.         <td><select name="refer" required>
26.             <option value="name">Task Name</option>
27.             <option value="desc">Description</option>
28.             <option value="date">Date</option>
29.             <option value="pr">Priority</option>
```

```
30.         </select></td>
31.         <td><input type="text" name="key" placeholder="Search Task" size="15" /><
    /td>
32.         <td><button type="submit">Search</button></td>
33.         </tr>
34.         </table>
35.     </form>
36.     </div>
37.     <b><big>To-Do LIST :</big></b>
38.     <table>
39.         <tr id="row">
40.             <th class="status">Status</th>
41.             <th class="name">Task Name</th>
42.             <th class="desc">Description Name</th>
43.             <th class="date">Date</th>
44.             <th class="pr">Priority</th>
45.         <th class="func1">Remove</th>
46.         <th class="func2">Modify</th>
47.         </tr>
48.     {% for todo in todos %}
49.         <tr class="datas">
50.             <td><a href="./done?_id={{ todo['_id'] }}"><input type="image" src="stati
    c/images/{{todo['done']}}.png" alt="Submit ME"></a></td>
51.             <td class="name">{{ todo["name"] }}</td>
52.             <td class="desc">{{ todo["desc"] }}</td>
53.             <td class="date">{{ todo["date"] }}</td>
54.             <td class="pr">{{ todo["pr"] }}</td>
55.             <td class="func1"><a href="./remove?_id={{ todo['_id'] }}"><button type=
    "submit">DELETE</button></a></td>
56.             <td class="func1"><a href="./update?_id={{ todo['_id'] }}"><button type="
    submit">EDIT</button></a></td>
57.         </tr>
58.     {% endfor %}
59.     </table>
60.     {% else %}
61.     <h4>No Tasks in the List !!</h4>
62.     {% endif %}
63.     <hr/>
64.     <form action="/action" method="POST">
65.     <table class="none">
66.         <tr>
67.             <td><b><big><label>Add a Task : </label></big></b></td>
68.         </tr>
69.         <tr>
70.         <td><input type="text" name="name" placeholder="Taskname" ></td>
71.         <td><textarea name="desc" rows="1" cols="30" placeholder="Enter Descripti
    on here..." required></textarea></td>
```

```
72.         <td><input type="text" name="date" placeholder="Date" /></td>
73.         <td><input type="text" name="pr" placeholder="Priority" /></td>
74.         <td><button type="submit"> Create </button></td>
75.         </tr>
76.     </form>
77.     </table>
78.     <script>
79.
80.     </script>
81. </body>
82. </html>
```

searchlist.html

```
1.  <html>
2.      <head>
3.          <title>{{t}}</title>
4.          <link rel="stylesheet" type="text/css"  href="{{ url_for('static',filename='assets
    /style.css')}}" >
5.      </head>
6.  <body>
7.      <h1>{{h}}</h1>
8.      <hr>
9.          {% if todos[0] %}
10.     <h3>Result of the Search : ToDO List</h3>
11.     <table>
12.         <tr id="row">
13.           <td class="status">Status</td>
14.            <td class="name">Task Name</td>
15.            <td class="desc">Task Description</td>
16.            <td class="date">Date</td>
17.            <td class="pr">Project</td>
18.         <td class="func">Delete</td>
19.         <td class="func">Modify</td>
20.         </tr>
21.         {% for todo in todos %}
22.         <tr class="datas">
23.             <td><a href="./done?_id={{ todo['_id'] }}"><input type="image" src="stati
    c/images/{{todo['done']}}.png" alt="Submit ME"></a></td>
24.             <td class="name">{{ todo["name"] }}</td>
25.             <td class="desc">{{ todo["desc"] }}</td>
26.             <td class="date">{{ todo["date"] }}</td>
27.             <td class="pr">{{ todo["pr"] }}</td>
28.             <td class="func1"><a href="./remove?_id={{ todo['_id'] }}"><button type=
    "submit">DELETE</button></a></td>
```

```
29.             <td class="func1"><a href="./update?_id={{ todo['_id'] }}"><button type="
    submit">EDIT</button></a></td>
30.         </tr>
31.         {% endfor %}
32.     {% else %}
33.         <h4>No Result Found !!</h4>
34.     {% endif %}
35.     </table>
36.    <a href="/">Return to TaskList</a>
37. </body>
38. </html>
```

update.html

```
1.  <html>
2.     <head>
3.             <title>{{t}}</title>
4.         </head>
5.         <style>
6.             h1{
7.                 font-family:"Arial Black", Gadget, sans-serif;
8.             }
9.             body{
10.                background-color:white;
11.            }
12.        </style>
13.    <body>
14.        <h1>{{h}}</h1>
15.        <hr>
16.        <h3>Update tasks with a reference</h3>
17.        <form action="/action3" method="POST">
18.        {% for task in tasks %}
19.            Unique Object ID : {{ task['_id'] }}<br/>
20.        <input type="text" name="_id" value="{{ task['_id'] }}" hidden>
21.
22.        <table>
23.        <tr>
24.        <td>Task Name</td><td> : </td><td><input type="text" name="name" value
    ="{{ task['name'] }}" placeholder="{{ task['name'] }}"></td>
25.        </tr>
26.        <tr>
27.        <td>Description</td><td> : </td><td><textarea type="text" name="desc" row
    s=3 cols=30 placeholder="{{ task['desc'] }}"> {{ task['desc'] }} </textarea></td>
28.        </tr>
29.        <tr>
30.        <td>Date</td><td> : </td><td><input type="text" name="date" value="{{ tas
    k['date'] }}" placeholder="{{ task['date'] }}"></td>
```

```
        </tr>
        <tr>
        <td>Priority</td><td> : </td><td><input type="text" name="pr" value="{{ tas
    k['pr'] }}" placeholder="{{ task['pr'] }}"></td>
        </tr>
        </table>
        {% endfor %}
            <button type="submit"> Update Task </button>
            <br/>
        </form>
        <a href="/">Return to TaskList</a>
</body>
</html>
```

Además, debemos agregar archivos emoji.css, emoji.js, style.css y twemoji.min.js en la carpeta static\assets.

emoji.css

```
img.emoji {
// Anula cualquier estilo img para asegurarte de que los Emojis se muestren
en  línea
margen :  0px ! importante ;
pantalla : en  línea ! importante ;
}
```

emoji.js

```
window.onload =  function () {
      // Establece el tamaño de los Emojis renderizados
      // Esto se puede configurar en 16x16, 36x36 o 72x72
     twemoji.size =  '16x16' ;
      // Analiza el cuerpo del documento y
      // inserta etiquetas <img> en lugar de Unicode Emojis
      twemoji.parse (document.body);
}
```

style.css

```
h 1 {
/ * font-family: "Arial Black", Gadget, sans-serif; * /
    Familia tipográfica : "Times New Romans" , Gadget,  sans-serif ;
}
cuerpo{
   de color : negro ;
    color de fondo : blanco ;
margen     izquierdo : 10px ;
```

```
9.  margen     derecho : 10px ;
10. }
11. mesa{
12.     ancho : 95% ;
13.     colapso del borde : colapso ;
14.     diseño de tabla : fijo ;
15. }
16. por ejemplo,
17. {
18.     borde : rgba ( 224 ,  119 ,  70 ,  1 )  2px sólido ;
19.     ajuste de palabra: palabra de interrupción;
20. }
21. tabla #close {
22.
23.     diseño de tabla : predeterminado ;
24.     colapso del borde : separado;
25.     espacio entre bordes :  5px 5px ;
26.     borde : ninguno ;
27.     alineación vertical : arriba ;
28. }
29. mesa. ninguna
30. {
31.     borde : ninguno ;
32.     alineación vertical : arriba ;
33. }
34. mesa. ninguno  td
35. {
36.     borde :  ninguno ;
37. }
38. tr.row td {
39.     decoración de texto : cursiva ;
40. }
41. th.status {
42.     ancho : 5% ;
43. }
44. th.name{
45.     ancho : 20% ;
46. }
47. th.des {
48.     ancho : 40% ;
49.     relleno : 5px 5px 5px 5px ;
50. }
51. th.date{
52.     ancho : 8% ;
53. }
54. th.pr{
55.     ancho : 7% ;
```

```
56. }
57. th.func 1 {
58.    ancho : 6% ;
59. }
60. th.func 2 {
61.    ancho : 5% ;
62. }
63. input [type = submit] {
64.     ancho :  20em ;  altura :  2em ;
65. }
66. ul {
67.     tipo-estilo-lista :  ninguno ;
68.    margen :  0 ;
69.    relleno :  0 ;
70.    desbordamiento :  oculto ;
71.    color de fondo :  # 333 ;
72. }
73. en el {
74.     flotador :  izquierda ;
75. }
76. le a {
77.     pantalla :  bloque ;
78.     color :  blanco ;
79.    alinear texto :  centro ;
80.    acolchado :  14px 16px ;
81.    decoración de texto :  ninguno ;
82. }
83. a.active {
84.    color de fondo :  # 4CAF50 ;
85. }
86. / * Cambiar el color del enlace a # 111 (negro) al pasar el mouse * /
87. li a: hover {
88.     color de fondo :  # 111 ;
89. }
```

Hay dos imágenes, **no.png** y **si.png**, en la carpeta **static\images**.

Estamos listos para ejecutar nuestra aplicación de muestra.

Recuerde que debe estar iniciada una instancia de MongoDB. Ejecute el programa.

Ahora puede agregar un ejemplo: coloque como Task name **"Tarea de prueba para mongo con aplicación de Flask"**, descripción **"Descripción de muestra",** fecha **"05-08-2018"** y prioridad **"Alta"**

Después de hacer clic en el botón Crear, podemos ver inmediatamente los detalles de la tarea en la cuadrícula.

Tenga en cuenta que hay una nueva base de datos llamada mymongodb y la colección todo se crea ahora. Con esta aplicación, puede editar y eliminar fácilmente los documentos existentes.