Stock Monitor with Redis

Abstract

We will use flask and redis for this. Flask is a good python web micro framework which lets you focus only on things you need. There is more focus on the modularity of your code base. Redis is a key-value data store that can be used as a database. Redis is an excellent choice for caching and for constant real-time analysis of data coming in, hence redis is great tool to build a platform.

What is Redis?

Redis is an open source, BSD licensed, advanced key-value store. It is often referred to as a data structure server, since the keys can contain strings, hashes, lists, sets and sorted sets. Redis is written in C. This tutorial provides good understanding on Redis concepts, needed to create and deploy a highly scalable and performance-oriented system.

What is Flask?

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a micro framework that doesn't include an ORM (Object Relational Manager) or such features.

Configuration

Start by installing the extension with **pip install flask-redis.** Once that's done, configure it within your Flask config.

Usage

Setup

To add a Redis client to your application:

```
import redis
from flask import Flask, render_template
db = redis.StrictRedis(host='127.0.0.1', port='6379', db=0, charset="utf-8", decode_responses=True)
```

Accessing Redis

The redis client you created above from Flask Redis acts just like a regular Redis instance from the redis-py library:

```
import redis
from pydantic import EmailStr, ValidationError, PositiveInt
from redis_om import JsonModel, Field
@app. route("/stocks")
def stocks_page ():
    item = Item.find (). all ()
return render_template('stocks.html', items = item
```

Extra features in flask-redis

1. **app.py** with the following code, which adds a second route and function that you can step through in the debugger:

```
import redis
from flask import Flask, render_template
db = redis.StrictRedis(host='localhost', port='6379', db=0,
charset="utf-8", decode_responses=True)
app = Flask(__name__)
@app.route('/')
```

```
def home()
    return render_template(“base.html”)
from pydantic import PositiveInt
from redis_om import JsonModel, Field
class Item(JsonModel):
    name: str = Field(index=True)
    price: PositveInt = Field(index=True)
    barcode: PositveInt = Field(index=True)
    description: str = Field(index=True)
@app.route(“/stocks”)
def stocks_page():
    item = Item.find().all()
    return render_template(‘stocks.html’, items=item
if __name__ == “__main__”:
    app.run(debug=True)
```

2. **stocks.json** with the following code

```
[{“name”:”HDFC”,”price”:500,”barcode”:”123123123123”,”descripti
on”:”none”}]
```

3. Inside the hello_flask folder, create a folder named **templates**, which is where Flask looks for templates by default.
    - In the **templates** folder, create a file named **base.html** with the contents below.

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <!--  CSS -->
```

```
    <link rel="stylesheet" type="text/css"
href=static\css\myapp.css>

 <title>
        {% block title %}

        {% endblock %}
    </title>
  </head>
<body>
   <p style="text-align:center;"><img
src="https://wallpaperaccess.com/full/3561135.jpg" alt="stock-
market" height="320" width="450"></p>
         <div class="navbar">
          <a  href="{{url_for('home')}}"><button
style="display:block; margin: 0 auto; height: 30px; width:
400px; left: 250; top: 250; font-size:20px">Home</button></a>
          <a  href="{{url_for('stocks_page')}}"><button
style="display:block; margin: 0 auto; height: 30px; width:
400px; left: 250; top: 250; font-
size:20px">Stocks</button></a>
        </div>
        {% block content %}
        {% endblock %}
        </body>
        </html>
```

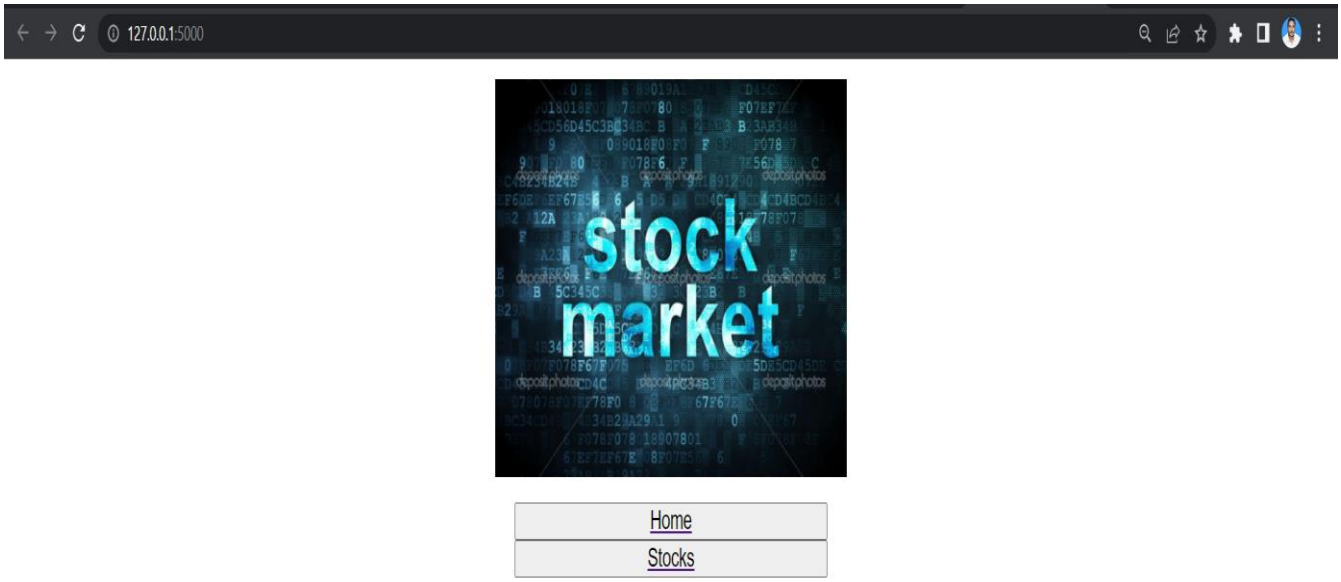
4. In the **templates** folder, create a file named **stocks.html** with the contents below.

```
{% extends 'base.html' %}
{% block title %}
   stocks
{% endblock %}

{% block content %}

<div>
<table>
   <thead>
      <tr>
       <th scope="col">ID</th>
       <th scope="col">Name</th>
       <th scope="col">Barcode</th>
       <th scope="col">Price</th>

       <th scope="col">options</th>
      </tr>
   </thead>

   <tbody>
     {% for item in items %}
        <tr>
          <td>{{ item.id }}</td>
          <td>{{ item.name }}</td>
          <td>{{ item.barcode }}</td>
          <td>{{ item.price }}</td>
          <td>
```

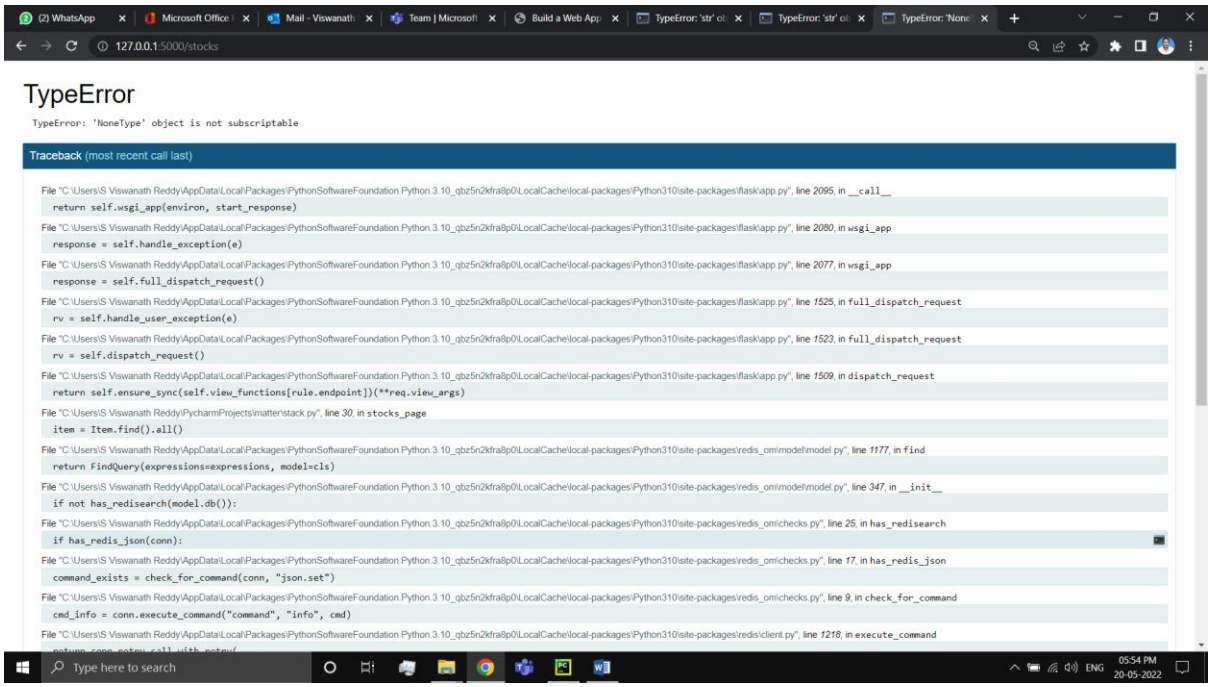
```
                <button class="btn btn-outline btn-info">More
Info</button>
                <button class="btn btn-outline btn-success">Purchase this
Item</button>
              </td>
            </tr>
        {% endfor %}
      </tbody>
    </table>
</div>

{% endblock %}
```

Output

create a route ”/stocks” and return stocks.html from the function. Then run your **main.py** file and click on the link that it provides after running. If it doesn’t show you any link, open your browser and on the url box, type http://0.0.0.0:5000/.



- Click on the Stocks button and to display the output but we didn’t get output it is showing **TypeError.**



Conclusion

Redis is an extremely popular, fast, flexible in-memory database with lots of great data structures. These features make it one of the most versatile NoSQL databases, with a superset of features of other in-memory competitors.