Penggunaan Fungsi Closure Pada Decorator Untuk Pengiriman Email Terjadwal

Irhamna Mahdi[1], Salwa Farhanatussaidah[2], Ganiya Syazwa[3], Reynaldi Rahmad[4], Syalaisha Andina Putriansyah[5]

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera

Email : *irhamna.122450049@student.itera.ac.id*[1] , *salwa.122450055@student.itera.ac.id*[2] , *ganiya.122450073@student.itera.ac.id*[3] , *reynaldi.122450088@student.itera.ac.id*[4] , *syalaisha.122450121@student.itera.ac.id*[5]

1. Pendahuluan

Informasi dalam bentuk teks didapatkan dari berbagai sumber seperti buku, surat kabar, situs web ataupun pesan email. Email merupakan suatu entitas penting yang digunakan untuk berkomunikasi digital melalui internet, selain itu digunakan untuk transfer informasi berupa file bahkan dapat digunakan untuk media iklan. Pesan Elektronik menjadi primadona untuk berkomunikasi saat ini. Hanya terhubung dengan koneksi internet, berkirim pesan elektronik dapat dengan mudah dilakukan. Semakin banyak orang yang terhubung ke internet menjadikan email sebagai salah satu alat komunikasi paling cepat dan ekonomis.

Fungsi closure sebuah konsep yang memungkinkan sebuah fungsi untuk mengakses variabel yang berada di lingkup luarannya dan sering digunakan dalam pembuatan decorator. Decorator merupakan pola desain yang memungkinkan penambahan fungsionalitas tambahan pada suatu fungsi tanpa mengubah strukturnya. Penggunaan fungsi closure pada decorator dapat memberikan kemampuan untuk menjadwalkan pengiriman email secara otomatis pada waktu yang ditentukan.

2. Metode

2.1 Metode Decoreator

Decoreator merupakan salah satu teknik yang digunakan untuk menambahkan fungsionalitas tambahan ke dalam objek tanpa mengubah sedikitpun strukturnya. Metode ini bekerja dengan membungkus objek asli atau fungsi dengan objek lain yang menyediakan perilaku tambahan tersebut[1]. Decorator sering diimplementasikan sebagai kelas, atau fungsi yang menerima objek asli sebagai parameter dan mengembalikan objek baru dengan fungsionalitas tambahan.

3. Pembahasan

3.1 Kode Program

3.1.1 Impor Modul

```
1 import time
2 import threading
3 from datetime import datetime
```

Gambar 1. Import Modul

Langkah awal dalam program ini meng-*import* modul yang diperlukan, yaitu time, threading dan datetime. Modul tersebut berfungsi untuk mengatur waktu dan membuat thread baru.

3.1.2 Fungsi Email

```
def Email(jam, menit):
    def decorator(func):
        def wrapper(*args, **kwargs):
            x = datetime.now()
            waktu = x.replace(hour=jam, minute=menit, second=0, microsecond=0)

            if waktu < x:
                waktu = waktu.replace(day=waktu.day + 1)

            waktupengiriman = (waktu - x).total_seconds()

            def send_email():
                time.sleep(waktupengiriman)
                func(*args, **kwargs)

            threading.Thread(target=send_email).start()
            print(f"Email terjadwalkan terkirim pada {waktu}")

        return wrapper
    return decorator
```

Gambar 2. Implementasi Teknik Decorator

Berikut penjelasan dari bagian-bagian kode pada gambar di atas:

1. Dekorator Email

   Fungsi Email merupakan fungsi decorator yang memiliki dua parameter , yaitu jam dan menit. fungsi ini mengambil fungsi lain sebagai argumen dan memperluas perilakunya tanpa secara eksplisit memodifikasinya.

2. Fungsi Wrapper

   Fungsi wrapper didefinisikan di dalam dekorator dan dipanggil saat fungsi yang di-*didecorate* (jadwalkan) dipanggil. Di dalam wrapper, x didefinisikan sebagai waktu saat ini dan waktu sebagai waktu target diatur. Waktu target diatur ke jam dan menit yang ditentukan, dengan detik dan mikrodetik diatur ke nol. Jika waktu target lebih awal dari waktu saat ini, hari ditambah satu (yaitu, jadwal untuk hari berikutnya). Fungsi wrapper juga mendefinisikan waktupengiriman sebagai selisih antara waktu target dan waktu saat ini dihitung dalam detik. Ini adalah waktu yang akan ditunggu sebelum mengirim email.

3. Fungsi send_email

   Fungsi ini menunggu selama waktupengiriman detik, lalu memanggil fungsi yang didekorasi.

4. Pembuatan Thread

   Thread baru dibuat dengan target send_email dan dijalankan. Ini berarti bahwa send_email akan berjalan di latar belakang,

memungkinkan program untuk melanjutkan eksekusi sementara menunggu waktu untuk mengirim email.

3.1.3 Penggunaan

```
@Email(jam=8, menit=0)
def jadwalkan():
    print("Mengirim notifikasi email...")

jadwalkan()
```

Gambar 3. Penerapan closure pada pengiriman Email terjadwal

Pada Gambar 3, Dekorator Email diterapkan ke fungsi jadwalkan dengan parameter jam = 8, dan menit = 0. Ini berarti bahwa ketika jadwalkan dipanggil, email akan dijadwalkan untuk dikirim pada jam 8:00. Proses penjadwalan email dimulai saat fungsi jadwalkan dipanggil.

3.2 Hasil Program

```
Email terjadwalkan terkirim pada 2024-04-26 08:00:00
```

Gambar 4. Hasil program

Pada Gambar 4 merupakan ouput dari penggunaan teknik decorator, dimana menyatakan bahwa email terjadwalkan telah terkirim pada tanggal 26 April 2024 jam 08.00.

4. Kesimpulan

Pembahasan diatas memberikan pengetahuan mengenai penggunaan decorator dan fungsi closure email terjadwal. Secara signifikan, dekorator email menggunakan fungsi closure untuk yang memungkinkan penjadwalan pengiriman email. Fungsi wrapper berfungsi untuk menangani logika penjadwalan dengan menghitung waktu yang harus ditunggu sebelum email dikirim. thread digunakan untuk memungkinkan program melanjutkan eksekusi sementara.

Dengan begitu, pendekatan yang dilakukan menyediakan sistem yang efektif dan modular untuk mengotomatiskan pengiriman email pada waktu tertentu. Dengan menggabungkan dekorator dan closure untuk memperluas fungsionalitas dan fleksibilitas aplikasi.

5. DAFTAR PUSTAKA

[1] V. Malcher, “DECORATOR PATTERN IN WEB APPLICATION,” *International Journal of Advanced Inforation Technology (IJAIT),* vol. 3, no. 4, pp. 13-14, 2013.