.net8 queue + background worker

Ref : <https://shadynagy.com/executing-background-tasks-dotnet/#backgroundtaskqueue>

<https://www.alter-solutions.com/articles/background-service-queue>

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-8.0&tabs=visual-studio>

พูดถึง queue มีเยอะมาก ในการจัดการ queue ของระบบ ที่เรารู้จักหรือได้ยิน ไม่ว่าจะเป้น ...

- BackgroundTaskQueue (บางที่เรียก worker) ที่เป้น lib ของ micrsoft

- RabbitMQ เปน message broker หรือ message queue ยอดนิยม

- IBMMQ เปน ตัวจัดการ queue ที่ทำไดหลายอย่าง เช่น Messaging, Queuing, Publish/subscribe, etc…

- Azure Service Bus เปน message queue ที่ Azure เอา RabbitMQ มาทำอีกที

- Azure Queue Storage ไว้เก็บข้อมูลสำหรับการจัดคิว และส่งข้อความระหว่าง app

etc .....

แต่ที่นี้จะพูดถึง BackgroundTaskQueue (lib : System.Collections.Concurrent)

หลักการ คือ

- เอา task ที่เราจะทำ Enqueue มันเข้าไปใน queue (task ที่ว่า จะเป็นอะไรก้ได้ เช่น CRUD)

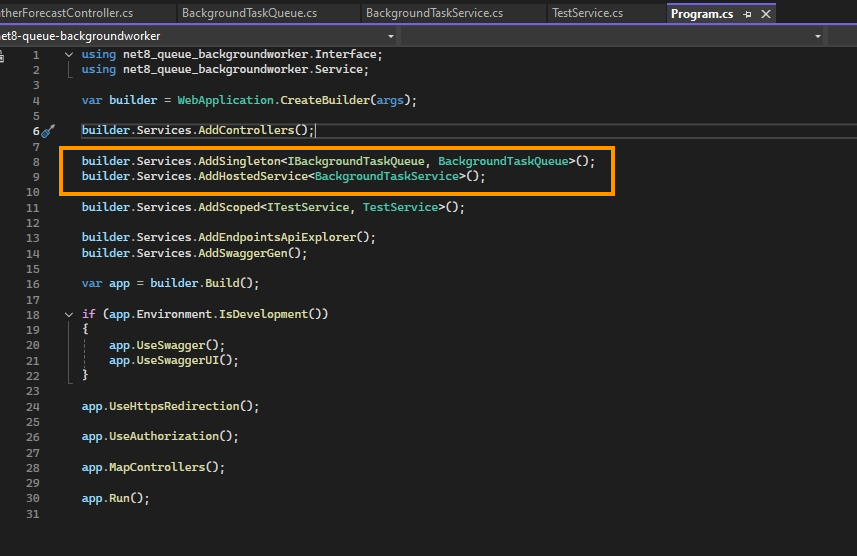
- ตัว worker จะจัดการ task ของเราตามลำดับที่ Enqueue เข้าไป

ณ ตอนที่ worker ทำงาน เราก้สามารถ get api ได้ปกติ (ทำงานคู่กันไป)

- Dequeue task และให้มันทำงาน รอจนเสด เปนอันจบครับ

อธิบายในส่วนของ code

ไฟล์ Program.cs : add Register DI มันเข้าไปซะ และ AddHostedService ของตัว BackgroundService

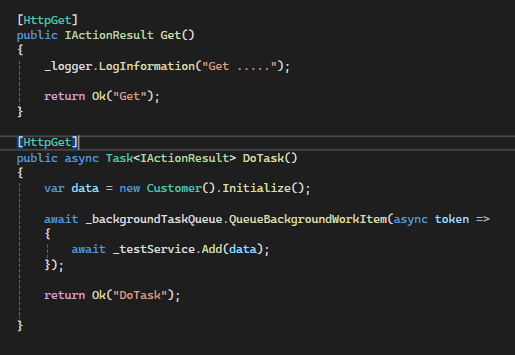


ไฟล์ Controller ทำไว้แค่ 2 api ครับ

- Get : return Ok("Get");

- DoTask : จะเอาไว้ add task เข้าไปใน queue

\*\* \_testService.Add(data) ที่เรา add task เข้าไป ในที่นี้ ต้องเป้น ValueTask นะครับ

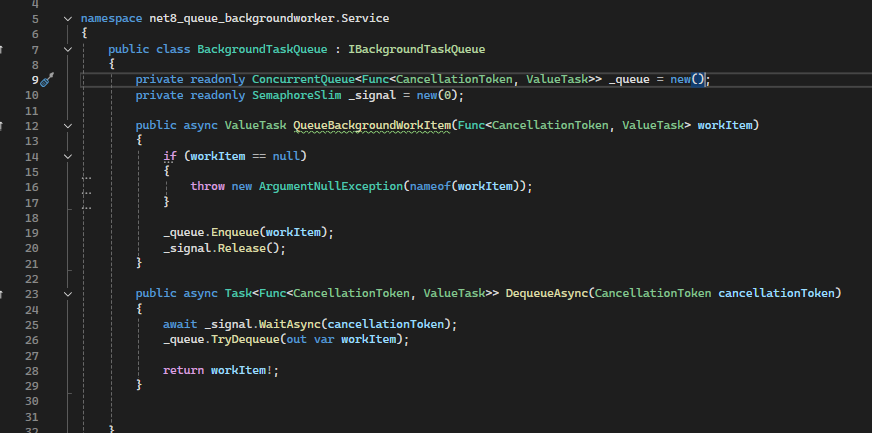


เริ่มจาก DoTask ที่เป้นตัว add task เข้าไปใน queue

โดย การ add task เข้า queue คือ ตัวนี้ครับ \_backgroundTaskQueue.QueueBackgroundWorkItem

มันเป้นตัวรับ task เข้าไปจัดการอีกทีครับ

ในนั้น ก้เป้นแบบนี้ครับ ตามรูป



มี 2 func คือ

- QueueBackgroundWorkItem : add task ของเราครับ ( task ของเรา คือ \_testService.Add(data) )

- DequeueAsync : เอาไว้ ลบ task ออกจาก queue

อธิบาย รูปก่อนหน้า

Line 9 : ประกาศประเภทตัวเก้บ queue ของเราครับ ในที่นี้จะใช้ ConcurrentQueue ตัวอื่นที่เป้ฯตัวเลือก ก้คือ channel

ความแต่กต่างของ 2 ตัวนี้ก้คือ : [Difference between Channel and ConcurrentQueue?](https://jeremybytes.blogspot.com/2021/02/whats-difference-between-channel-and.html)

ที่เราสามารถเลือกใช้ตามที่ต้องการได้ครับ การ add queue และ DeQueue ก้จะเขียนต่างกันไปด้วย

ข้างใน เปน call back func ครับ ส่ง CancellationToken เข้าไป return ValueTask ออกมา

(จะเปนอย่างอื่นก้ได้ครับ ไม่จำเปนต้อง call back func)

Line 10 : ที่ Line 9 เราใช้ ConcurrentQueue ในการจัดการ queue

ใน Line 10 นี้ก้เลยต้องใช้ SemaphoreSlim เข้ามาใช้ร่วมด้วย เพื่อ จัดการสถานะของ threading channel ครับ

Line 19 : เป้นการ Enqueue task ของเราเข้าไปใน ในที่นี้ task ก้คือ workItem

แล้วก้ให้ตัว \_signal มัน Release ชะ

Line 25 : ให้ signal มันรอเชคค่าอ่านค่า threading / channel อะไรของมันให้เสดก่อน

แล้วถึงจะ TryDequeue ของ queue นั้นๆ

เมื่อ add task เสด มันจะไปอยุ่ที่การทำงานของ BackgroundService แล้วครับ ตอนนี้

\*\* BackgroundService คือ ตัวจัดการ การทำงานต่างๆ แบบ Background

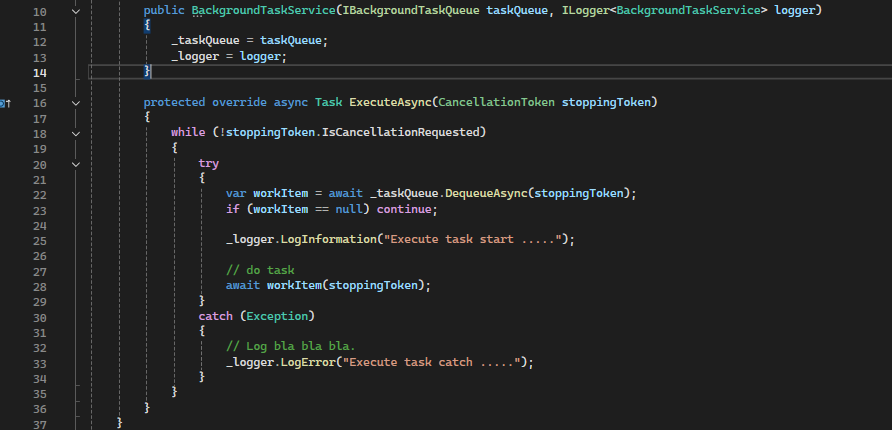
ถ้าใน กรณี ที่มี background worker หรือ background task เยอะๆ เราสามารถแยกออกมาเป้นอีก service ได้เลยครับ

แยกการทำงานเป้นเฉพาะ background worker ไปเลย (ธนาคารหรือ ระบบ CMS จะนิยมแยก เพื่อทำ transection ต่างๆ)

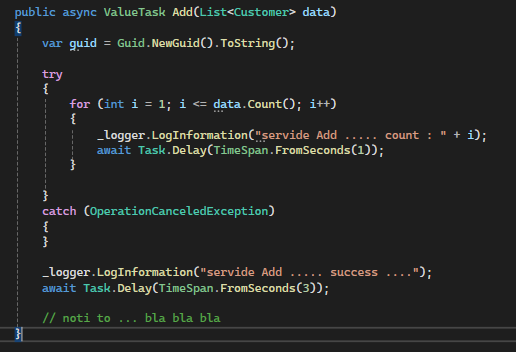
แต่ในที่นี้ เราเอาในใช้ ใน service api ตรงนี้เลยครับ

ข้างในของ BackgroundService ก้ไม่มีอะไรครับ แค่รัน task ต่อๆกันไปเรื่อยๆครับ และรอมันทำเสด

ที่เหลือ เราก้ ทำ log หรือ noti ให้รู้ ว่ามันทำงานเสดและไม่ error หรือไว้ดู error ไรพวกนี้ครับ

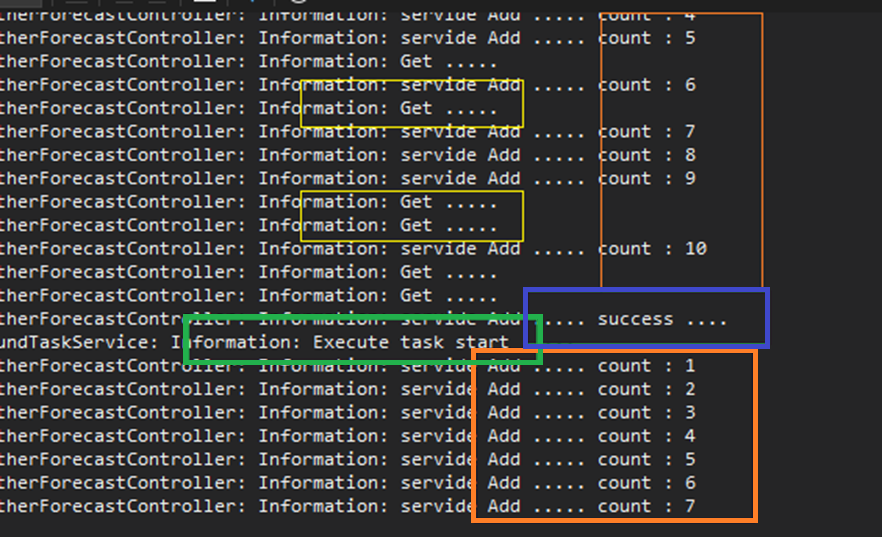


Task ของเรา คือ \_testService.Add(data) ที่ใน testService method Add จะให้ทำ ตามรูปนี้ครับ



เมื่อเรา add task มันที่ Controller แล้ว หมายความว่า task นี้ อยุ่ใน queue แล้วครับ

ผลการทดสอบ เมื่อ run และ ยิง request สลับ ไปมา ทั้ง 2 api (get / doTask)



จะเห้นว่า

**เมื่อเริ่มทำ task ตาม queue จะให้แสดง Execute task Start … (กรอบเขียว)**

**task ใน \_testService.Add(data) มันก้ทำงานของมันไปปกติครับ (กรอบส้ม)**

**แต่เมื่อเรายิง api get มันก้จะทำทันทีเลย (แยกการทำงาน เหมือนที่บอกไปครับ) (กรอบเหลือง)**

**และเมื่อ task ทำงานเสด เราจะให้แสดง log success (กรอบน้ำเงิน)**

\*\*\*\* ความสามารถเพิ่มเติมของมัน

จาก : [Efficient Background Task in .NET Core Techniques and Best Practices](https://shadynagy.com/executing-background-tasks-dotnet/#backgroundtaskqueue)

เช่น …. (อ่านเพิ่มเติมเองได้)

- สามารถทำ config ต่างๆแยกออกมาได้

- ทำ Retry Policy เพิ่มเติมได้

- Scaling Background Task Processing ได้

เช่น Use the number of processor cores as worker count

- ทำ Priority ให้ task ได้

- support load balancing