Application Pool คือ 👉 **ขอบเขตการแยกการทำงาน** ของเว็บแอปใน IIS

Application Pool Identity คือ "ตัวตน (บัญชีผู้ใช้ในระบบ Windows)" ที่ใช้รัน **Application Pool** นั้น ๆ

พูดง่าย ๆ คือ 👉 มันกำหนดว่า **เว็บแอปของคุณจะรันด้วยสิทธิ์ของใคร**

เอาไว้ทำอะไร?

เอาไว้ควบคุม **สิทธิ์การเข้าถึงทรัพยากร** ของเว็บ เช่น

- อ่าน/เขียนไฟล์ในเครื่อง
- เข้าถึง Network / Shared Folder

ถ้าตั้ง Identity ไม่ถูก เว็บอาจจะ:

- เปิดไม่ได้
- เขียนไฟล์ไม่ได้

ประเภทของ Application Pool Identity ที่พบบ่อย

1️⃣ **ApplicationPoolIdentity (ค่า default แนะนำ 👍)**

- IIS สร้าง account เสมือนให้เอง
- ปลอดภัยกว่า เพราะแต่ละ App Pool แยกสิทธิ์กัน
- ชื่อในระบบจะเป็นประมาณ:

2️⃣ **NetworkService**

- เป็น account มาตรฐานของ Windows
- สิทธิ์ค่อนข้างกว้าง
- ใช้ชื่อเครื่องเวลาติดต่อ resource ภายนอก

⚠️ ปลอดภัยน้อยกว่า AppPoolIdentity

3️⃣ **LocalSystem**

- สิทธิ์สูงมาก (เกือบ admin)
- **ไม่แนะนำ** เว้นแต่รู้ว่าทำอะไรอยู่จริง ๆ

4️⃣ **Custom Account (Domain / Local User)**

- ใช้ user ที่สร้างเอง
- เหมาะกับกรณี:
    - ต้องเข้าถึง Shared Folder / ต้องใช้สิทธิ์ใน Domain

การตั้งค่า และ การจัดการกับ Pool

Recycling Management อ่านว่า 👉 **รี-ไซ-เคิลลิ่ง แมน-เนจ-เมนต์ / รี-ไซ-คลิง แมน-เนจ-เมินท์**

เวลา App Pool ถึงรอบ Recycle จะเกิดอะไรขึ้นบ้าง

1️⃣ **IIS สร้าง Worker Process ตัวใหม่ (w3wp.exe)**

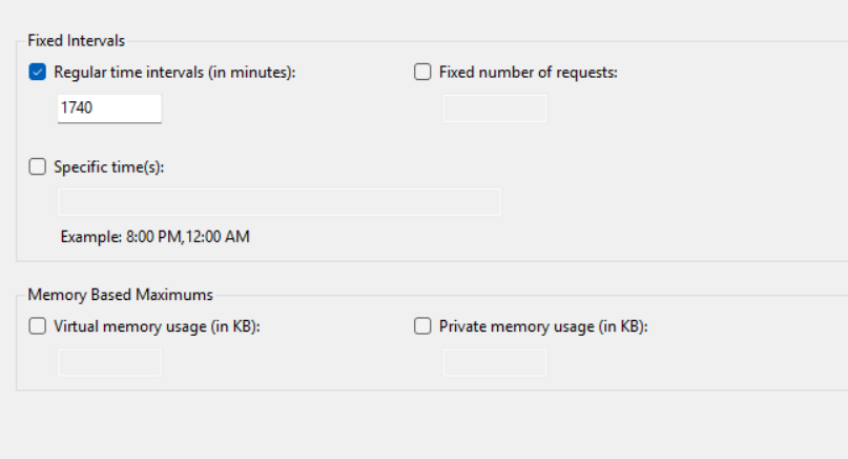
- ตัวใหม่เริ่มทำงานก่อน

2️⃣ **Request ใหม่**

- ถูกส่งไปให้ process ตัวใหม่

3️⃣ **Process ตัวเก่า**

- ยังทำงานต่อจนกว่า
    - request ที่ค้างอยู่จะเสร็จ
    - หรือถึงเวลาบังคับปิด (shutdown limit)



CPU Throttling ธรอท-ถ-ลิง คือการ **จำกัดการใช้ CPU** ของโปรแกรม / process / service
เพื่อไม่ให้ตัวใดตัวหนึ่ง "กิน CPU เกินไป" จนกระทบตัวอื่น ๆ

พูดง่าย ๆ 👉 *เป็นตัวเบรก* ไม่ให้เครื่องวิ่งแรงเกินที่ตั้งไว้

ใช้ควบคุมว่า **App Pool นี้ใช้ CPU ได้กี่ %** เช่น

- ตั้งไว้ **30%**
- ถ้า App Pool ใช้ CPU เกิน 30% ตามช่วงเวลาที่กำหนด
  → IIS จะจัดการตาม policy ที่ตั้งไว้
- **ตัวอย่างสถานการณ์จริง**
- 🔹 มีเว็บ 5 เว็บอยู่ในเครื่องเดียว
  เว็บหนึ่ง query หนักมาก
- → ตั้ง CPU Throttling ให้เว็บนั้นใช้ได้แค่ 20%
  → เว็บอื่นไม่ข้าไปด้วย

Limit Interval: default (5 นาที) ใน IIS
คือ ช่วงเวลาที่ IIS ใช้ "วัดค่า CPU" เพื่อเอาไปเทียบกับ CPU Limit

พูดง่าย ๆ 👉 *IIS จะดูย้อนหลังทีละ 5 นาที ว่า App Pool ใช้ CPU เกินที่กำหนดหรือไม่*

อธิบายแบบเห็นภาพ

สมมติคุณตั้งค่า:

- **CPU Limit = 30%**
- **Limit Interval = 5 นาที**

IIS จะคำนวณว่า

ในช่วง **5 นาทีล่าสุด**
App Pool นี้ใช้ CPU **เฉลี่ย** เกิน 30% หรือเปล่า

- ❌ ถ้าไม่เกิน → ไม่ทำอะไร
- ✅ ถ้าเกิน → ทำตาม **Limit Action**
    - Throttle
    - ThrottleUnderLoad
    - KillW3wp
    - ฯลฯ

ทำไมต้องมี Interval?

เพราะ CPU มัน **spike เป็นช่วง ๆ** ได้
IIS เลยไม่ดูแค่พริบตาเดียว แต่ดูเป็น "ช่วงเวลา"

ตัวอย่าง:

- CPU พุ่ง 90% แค่ 3 วินาที
  → **ไม่โดน throttle**
- CPU อยู่ 50–60% ต่อเนื่องหลาย ๆ นาที
  → **โดนแน่**

คำแนะนำจากประสบการณ์

- ใช้ **ค่า default 5 นาที** → ดีสุดสำหรับ 80% ของระบบ
- ไม่ควรปรับ ถ้าไม่เจอปัญหาจริง
- ปรับเฉพาะกรณีรู้ pattern ของโหลดชัดเจน

กำหนด config WebSite ด้านความปลอดภัย

SSL (Secure Sockets Layer)/TLS Configuration (Transport Layer Security) บน IIS คือ
การตั้งค่าให้เว็บของคุณ **สื่อสารแบบเข้ารหัส (HTTPS)** เพื่อความปลอดภัย

พูดง่าย ๆ 👉 *เป็นการบอก IIS ว่า จะใช้กุญแจและโปรโตคอลอะไร ในการคุยกับผู้ใช้*

ปัจจุบันใช้ **TLS** หมดแล้ว
คำว่า SSL ยังใช้เรียกรวม ๆ เฉย ๆ

SSL / TLS คืออะไร (สั้น ๆ)

- **SSL / TLS** = เทคโนโลยีเข้ารหัสข้อมูล
- ป้องกัน:
    - ขโมยข้อมูล / แก้ไขข้อมูลระหว่างทาง

Perfect Forward Secrecy คือคุณสมบัติของการเข้ารหัส TLS/SSL ที่ช่วยป้องกันข้อมูลย้อนหลัง

สร้าง **session key ใหม่ทุกการเชื่อมต่อ**

นอกจากนี้ยังมี **Certificate Pinning** ซึ่งเป็นการกำหนดให้ Client เชื่อถือเฉพาะ Certificate หรือ Public Key ที่กำหนดไว้เท่านั้น ช่วยลดความเสี่ยงจาก Certificate ปลอมหรือถูกปลอมแปลงเหมาะสำหรับระบบที่ต้องการความปลอดภัยสูงเป็นพิเศษ

```
val certificatePinner = CertificatePinner.Builder()
    .add(
        "example.com",
        "sha256/AbCdEfGhIjKlMnOpQrStUvWxYz1234567890="
    )
    // 🔥 แนะนำ pin สำรอง
    .add(
        "example.com",
        "sha256/BackupKeyHashHere="
    )
    .build()

val client = OkHttpClient.Builder()
    .certificatePinner(certificatePinner)
    .build()
```

กำหนด HTTP Response Header

ลำดับการทำงาน IIS ↔ Browser

1️⃣ Browser ส่ง Request

GET /index.html HTTP/1.1 Host: example.com

2️⃣ IIS ประมวลผล

- ประมวลผลโค้ด -- > เตรียม response

3️⃣ IIS ส่ง Response + Headers

ตัวอย่าง:

HTTP/1.1 200 OK

Content-Type: text/html

Set-Cookie: SessionId=abc123

Strict-Transport-Security: max-age=31536000

X-Frame-Options: DENY

📌 **Headers มาก่อน body เสมอ**

🛂 **Browser อ่าน Headers ก่อน**

Browser จะ:

- อ่าน header ทุกตัว
- ตัดสินใจ "กฎการใช้งาน"
- แล้วค่อย render หน้าเว็บ

Browser ทำอะไรกับ Headers บ้าง?

🔐 **Security**

Header	**Browser ทำอะไร**
Strict-Transport-Security	บังคับ HTTPS
X-Frame-Options	กันโดน iframe
Content-Security-Policy	จำกัด script / resource

ใครเป็นคน "ตัดสินใจจริง"?

🔹 **IIS**

- เป็นคน "ส่ง header"
- กำหนดนโยบาย

🔹 **Browser**

- เป็นคน "บังคับใช้"
- ถ้า header ผิด → browser ไม่สน

HSTS → HTTP Strict Transport Security

HTTP Response Header ที่บังคับให้ Browser ใช้ HTTPS เท่านั้น

พูดง่าย ๆ 👉 *สั่งเบราว์เซอร์ว่า "ห้ามคุยกับเว็บนี้ผ่าน HTTP อีก"*

มันทำงานยังไง?

เมื่อผู้ใช้เข้าเว็บผ่าน **HTTPS** แล้ว
Server จะส่ง header นี้กลับไป เช่น

Strict-Transport-Security: max-age=31536000; includeSubDomains

Browser จะจำไว้ว่า:

- ต้องเข้าเว็บนี้ด้วย **HTTPS เท่านั้น**
- ต่อให้ผู้ใช้พิมพ์ http://
  → browser จะเปลี่ยนเป็น https:// ให้เองทันที

Project Web

1️⃣ Request Filtering คืออะไร?

Request Filtering คือ
👉 การตั้งกฎให้ IIS **กรอง / ปฏิเสธ request** ที่ "หน้าตาน่าสงสัย"
ก่อนจะถึงโค้ดเว็บของคุณ

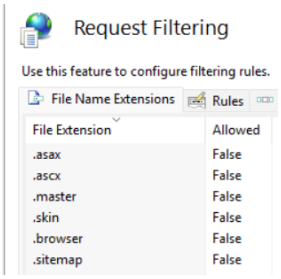
กันตั้งแต่หน้าประตู ไม่ให้ request แปลก ๆ เข้าแอป

Request Filtering กรองอะไรได้บ้าง?

- 🔹 **URL / Path**
  - ห้าม path แปลก ๆ
  - กัน ../ (path traversal)
  - กัน double encoding

- 🔹 **File Extensions**
  - บล็อกไฟล์อันตราย เช่น
    - .exe .cmd .bat .config

แม้ไฟล์จะอยู่ในโฟลเดอร์เว็บ ก็เรียกไม่ได้



- 🔹 **HTTP Verbs**
  - อนุญาตเฉพาะ: GET / POST
  - ปิด: PUT / DELETE / TRACE (อันตราย)
  -

2️⃣ IP Blocking (IP Address Restrictions) คืออะไร?

IP Blocking คือ
👉 การกำหนดว่า **IP ไหนเข้าเว็บได้ / ไม่ได้**

เป็นการควบคุมตาม "แหล่งที่มา"

IP Blocking ทำอะไรได้บ้าง?

- 🔹 **Allow / Deny**

🔹 **Dynamic IP Restrictions**

บล็อกอัตโนมัติเมื่อ:

- request ถี่เกิน
- login fail ซ้ำ ๆ / เปิด connection พร้อมกันเยอะผิดปกติ
- scan เว็บ

ช่วยกัน brute force / bot

Dynamic IP Blocking คืออะไร?

คือฟีเจอร์ของ IIS ที่
👉 **บล็อก IP อัตโนมัติ** เมื่อพบพฤติกรรมผิดปกติ เช่น

- ยิง request ถี่เกิน / พยายามเดา password brute force login / สแกนเว็บ

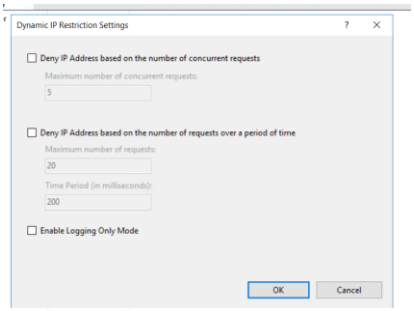
มัน block จากอะไรบ้าง?

🔹 **1) Request ต่อวินาที**

- เช่น IP เดียว ยิงเกิน 10 request / วินาที
- → block

🔹 **2) Concurrent Requests**

- เปิด connection พร้อมกันเยอะผิดปกติ
- → block



ขั้นที่ 5: ตั้งค่าที่นิยมใช้ (แนะนำ)

ตัวอย่างค่าปลอดภัยสำหรับเว็บทั่วไป:

- ✅ **Deny IP address based on the number of concurrent requests**
  - เช่น 20
- ✅ **Deny IP address based on the number of requests over a period of time**
  - เช่น 10 requests per second
- ⏱️ **Deny Action**
  - Abort request (default ดีแล้ว)

IIS Dynamic IP Restrictions กับ Imperva CDN อยู่คนละระดับกันเลย

ภาพรวมสั้น ๆ ก่อน

- **Dynamic IP Restrictions (IIS)**
  = กันแบบพื้นฐาน ระดับเว็บเซิร์ฟเวอร์
- **Imperva CDN / WAF**
  = กันระดับ enterprise อยู่ "หน้าบ้าน" ก่อนถึง IIS

เทียบแบบตรง ๆ

หัวข้อ	Dynamic IP (IIS)	Imperva CDN
ทำงานที่ไหน	บน IIS เครื่องคุณ	หน้าเว็บ (Edge / CDN)
กันอะไร	ยิงถี่ / bot ง่าย ๆ	Bot ขั้นสูง, DDoS, OWASP
วิเคราะห์พฤติกรรม	❌ แค่นับจำนวน	✅ Behavioral / ML
ป้องกัน DDoS	❌ แทบไม่ได้	✅ ระดับใหญ่
กัน SQLi / XSS	❌ ไม่ได้	✅ ได้
Geo Block	❌	✅
Rate Limit ฉลาด	❌	✅
ดู dashboard	❌	✅
ค่าใช้จ่าย	ฟรี	มีค่าใช้จ่าย