

1. Backup

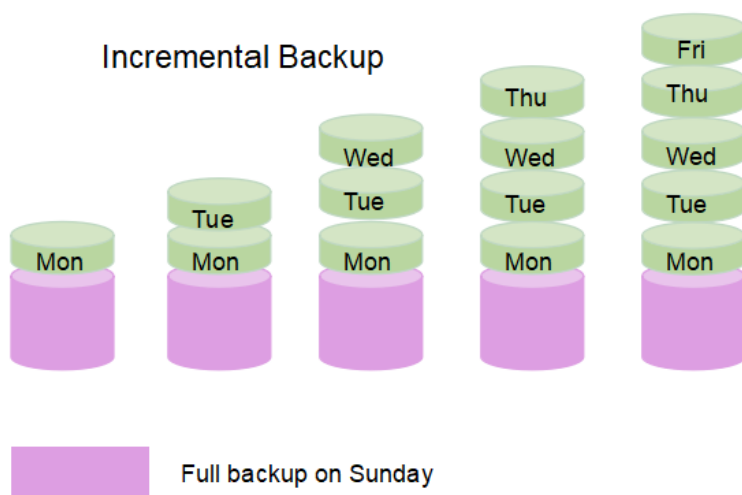
การบำรุงรักษาซอฟต์แวร์ คือการเปลี่ยนแปลงระบบภายหลังจากการส่งมอบเพื่อแก้ไขข้อผิดพลาดหรือข้อผิดพลาดเพื่อนำมาปรับปรุงให้มีประสิทธิภาพ ซึ่งหนึ่งในกระบวนการสำคัญที่ช่วยในการแก้ไขปัญหาของระบบหากมีข้อผิดพลาดเกิดขึ้นคือการสำรองข้อมูลของระบบ ซึ่งการสำรองข้อมูลมีตั้งแต่ข้อมูลของโค้ดต้นฉบับ และการสำรองข้อมูลของผู้ใช้งานที่ซึ่งมีความสำคัญมากโดย การสำรองข้อมูลต้นฉบับสามารถแบ่งวิธีการ Backup ออกเป็น 4 วิธีมีดังนี้

1) Full Backup

เป็นการ Backup ทุกไฟล์ เช่น Full Backup ของ Windows ก็ทำสำเนาเอาทุกไฟล์ในทุกไดรฟ์อย่าง C:\ , D:\ และอื่นๆ มาทั้งหมด ถ้าใน Unix หรือ Linux ก็ไล่ตั้งแต่ /home, /opt และอื่นๆ ข้อแนะนำคือควรยกเว้นการ Backup เฉพาะไฟล์ที่รู้ว่าไม่จำเป็นจริงๆ เท่านั้น เช่น ไฟล์ Config อย่าง C:\Windows\TEMP หรือ /tmp ใน Linux ทั้งหมดนี้ก็เพื่อหลีกเลี่ยงในกรณีที่คนอื่นๆ อาจนำไฟล์ที่เราไม่รู้ไปวางไว้ที่อื่นและอาจไม่ได้ถูก Backup ไปด้วย

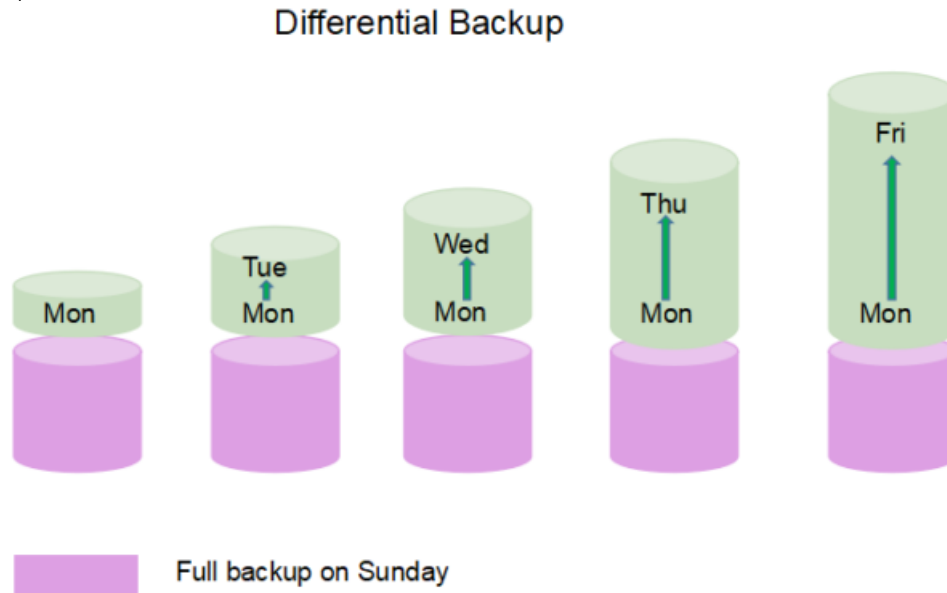
2) Incremental Backup

เป็นการ Backup เอาเฉพาะข้อมูลที่มีการเปลี่ยนแปลงจากการ Backup ครั้งล่าสุด ดูตัวอย่างตามรูปด้านล่างจะเห็นได้ว่าไฟล์ Backup จะมีขนาดเล็กนี้คือข้อดีของการไม่ไปรบกวนประสิทธิภาพของเซิร์ฟเวอร์มากนักเพราะจะให้ Backup ไฟล์ 10 GB ทั้งๆ ที่มีการเปลี่ยนแปลงข้อมูลแค่ 1 MB ซึ่งเป็นการทำงานที่ใช้พลังงานสูงเกินไป แต่หากต้องการได้ข้อมูลของวันศุกร์ก็ต้องเอาข้อมูลจากชุด Backup จากวันจันทร์ถึงพฤหัสบดี มาประกอบกันถ้าเสียไปอันหนึ่งก็จะมีปัญหา นอกจากนี้ ปัจจุบันซอฟต์แวร์ Backup สมัยใหม่ก็มีการใช้งาน Backup แบบ Block-based incremental คือการจัดการในระดับ Block โดยการใช้ API จาก VMware หรือ Hyper-V



3) Differential Backup

การ Backup เฉพาะข้อมูลที่มีการเปลี่ยนแปลงจาก Full Backup ตามรูปด้านล่าง ซึ่งวิธีการเป็นที่นิยมกับการใช้เทป Backup โดยการนำข้อมูลกลับไปใช้ Full Backup และ ไฟล์ Backup เพิ่มไปยังจุดหมายที่ต้องการซึ่งประหยัดเวลาว่าการทำ Full Backup ได้มาก



4) Forever-incremental

การมาถึงของดิสก์และ Deduplication ได้ทำให้การ Backup แบบ Full และ Differential นิยมน้อยลงเพราะไม่ต้องนั่งคิดเรื่องลดจำนวนเทปอีกต่อไป รวมถึงเวลาการ Restore ข้อมูลจาก incremental จำนวนมากกับ Full Backup แทบไม่ต่างกันเพราะว่าอันที่จริงแล้วในระบบ Backup เพียงทำการเก็บบันทึกไว้ว่า ไฟล์หรือ Block ทั้งหมดนั้นถูกเก็บอยู่ที่ไหนใน Storage และก็แค่ย้ายจาก Storage กลับไปยังเครื่อง Client เท่านั้นเอง ทั้งนี้ขั้นตอนก็ไม่ได้สัมพันธ์กันด้วยในโลกของการ Backup ยุคใหม่ๆ และวิธีการ Backup แบบ Forever incremental นั้นก็จะช่วยเรื่องของการได้ข้อมูลที่อัปเดตล่าสุดอีกด้วย

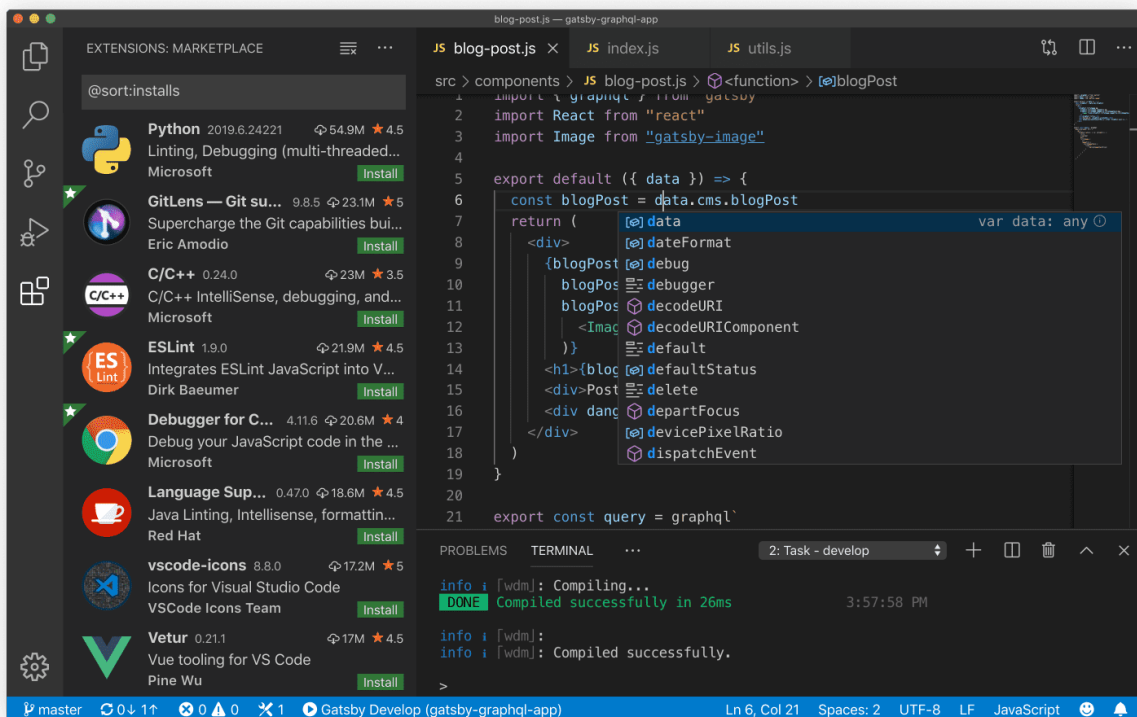
นอกจากในกระบวนการ Backup จะมีหลากหลายวิธีการในการ backup แล้วเพื่อให้ผู้พัฒนาสามารถเลือกไปใช้งานให้เหมาะสมกับระบบ ยังมีแนวคิดในการ backup เพิ่มเข้ามาด้วย ซึ่งแนวคิดที่ใช้งานกันบ่อยก็คือ แนวคิด 3-2-1 Backup Rule กฎ 3-2-1 Backup Rule นี้มีใจความหลักๆ ด้วยกัน 3 ข้อสั้นๆ ได้แก่

- 1) แนะนำให้มีการเก็บข้อมูลสำคัญเอาไว้ 3 ชุดเป็นอย่างน้อย ได้แก่ข้อมูลหลักต้นฉบับ 1 ชุด และข้อมูลสำรองอีก 2 ชุด (จะได้กระจายข้อมูลไปเก็บหลายๆ อุปกรณ์ได้)
- 2) ควรเก็บไฟล์เหล่านั้นเอาไว้บนอุปกรณ์ที่แยกขาดจากกัน 2 ประเภทเป็นอย่างน้อย (หากอุปกรณ์หนึ่งเสีย อีกอุปกรณ์จะได้ยังคงไม่เสียไปด้วยปัจจัยเดียวกัน)

- 3) ข้อมูลสำรองชุดหนึ่ง ควรนำไปเก็บไว้ที่ต่างสาขา หรือสำรองเอาไว้แบบ Offline เป็นอย่างน้อย (หากเกิดเหตุใดๆ ขึ้นกับสถานที่หนึ่ง จะได้ยังคงมีข้อมูลสำรองที่ปลอดภัยจากเหตุอื่นๆ)

หากนึกภาพตามดีๆ ในมุมมององค์กรแล้ว กฎ 3 ข้อนี้นี้ก็จะนำไปสู่การทำระบบ Local Backup และ Remote Backup ทั้งนี้แนวคิดนี้สามารถนำไปประยุกต์ใช้ได้กับทั้งการจัดการสำรองข้อมูลส่วนตัว (สำรองข้อมูลในเครื่อง + สำรองข้อมูลบน Cloud), การสำรองข้อมูลในที่ทำงาน (สำรองข้อมูลภายในสาขา + สำรองข้อมูลในต่างสาขาหรือ Cloud) ไปจนถึง Data Center ขนาดใหญ่ที่มีแนวทางในการออกแบบระบบสำรองข้อมูลที่หลากหลาย

2. Visual Studio Code



Visual Studio Code เป็นโปรแกรมแก้ไขซอร์สโค้ดที่พัฒนาโดย Microsoft สำหรับ Windows, Linux และ macOS

Visual Studio Code สามารถปรับแต่งได้หลายอย่าง ให้ผู้ใช้สามารถเปลี่ยนธีม แบนพิมพ์ลัด การตั้งค่า และติดตั้งส่วนขยายที่เพิ่มฟังก์ชันการทำงานเพิ่มเติม Visual Studio Code เป็น Open Source จึงสามารถใช้งานได้ฟรีและเผยแพร่ภายใต้สิทธิ์การใช้งาน MIT

ในการสำรวจนักพัฒนา Stack Overflow 2019 Visual Studio Code ได้รับการจัดอันดับให้เป็นเครื่องมือสำหรับนักพัฒนาที่ได้รับความนิยมมากที่สุดถึง 50.7% ของผู้ตอบแบบสอบถาม 87,317 ราย

3. KeyCloak

Keycloak คือ Open Source Project และเป็นเครื่องมือสำหรับจัดการ การยืนยันตัวตนและการเข้าถึงข้อมูล เช่น Access Management, Identity Brokering, User Federation ด้วย LDAP/AD, Event Auditing โดยมี Client Library หลากหลายทั้งบน Java EE, MicroProfile, Spring, Node.js และอื่นๆ

ความสามารถหลักของ Keycloak จะแบ่งออกเป็นส่วนๆ ดังนี้

- Authentication การยืนยันตัวตนผู้ใช้งาน
- Authorization สามารถกำหนดสิทธิ์ในการเข้าถึงทรัพยากรต่างๆ ของผู้ใช้งานแต่ละคนได้
- Clients Management รองรับ Application ได้หลาย App สำหรับผู้ใช้งานแต่ละคน
- Session Management จัดการ Session ของผู้ใช้งานแต่ละคน พร้อมทั้งตรวจสอบ Session ทั้งหมดได้
- Identity Manager (IdM) / Identity Access Manager (IAM)
- Identity Provider (IdP)
- Service Provider (SP) รองรับ SAML, OIDC และอื่นๆ
- Identity Provider Federation / Identity Broker ทำงานร่วมกับ Facebook, Google, GitHub และอื่นๆ ที่รองรับ OpenID Connector, SAML ได้

ใน Keycloak ยังมี Flow ให้เลือกกำหนดการยืนยันตัวตน การกำหนดสิทธิ์ การ Login และสิ่ง que ผู้ใช้งานสามารถทำได้ในระหว่างกระบวนการการยืนยันตัวตน เช่น การตรวจสอบ Cookie หรือ Access Token หรือ Kerberos ก่อนว่าเคยยืนยันตัวตนอยู่แล้วหรือยัง, การตรวจสอบว่ารหัสผ่านหมดอายุหรือยัง และ Reset รหัสผ่าน, การเพิ่มการทำ OTP/OOB/reCAPTCHA, การทำ MFA และอื่นๆ

โดย Add-on Flow ต่าง ๆ ที่ช่วยเสริมการทำงาน ต่าง ๆ ที่ keycloak มีประกอบไปด้วย

- Authentication Flow เช่น Registration flows, login flows, credential reset flows, etc.
- Registration Flow เช่น self-register, email validation, etc.
- Login Flow เช่น user/password ร่วมกับ MFA
- Required Actions เช่น interval or schedule password reset

4. FireWall Network security

Firewall (ไฟร์วอลล์) เป็นระบบที่ออกแบบมาเพื่อป้องกันและควบคุมการเข้าถึงข้อมูลและเครือข่ายคอมพิวเตอร์จากผู้ไม่พึงประสงค์หรือภัยคุกคามต่าง ๆ ที่อาจเข้าถึงระบบขององค์กรหรือบุคคลทั่วไปได้ หรือเพื่อควบคุมการออกจากเครือข่ายขององค์กรหรือระบบคอมพิวเตอร์ให้เป็นไปตามนโยบายและกฎระเบียบที่กำหนดไว้

Firewall ทำหน้าที่เป็นอุปกรณ์หรือซอฟต์แวร์ที่คัดกรองและควบคุมการเข้าถึงข้อมูลโดยตรวจสอบข้อมูลที่ถูกส่งมาหรือออกไปผ่านเครือข่ายคอมพิวเตอร์ โดยมีหลายวิธีดำเนินการเช่น Packet Filtering Firewall

Packet Filtering Firewall (ไฟร์วอลล์ประเภทการตัดสินใจในระดับแพ็คเก็ต) ระบบนี้ตรวจสอบแพ็คเก็ตของข้อมูลและตัดสินใจว่าจะอนุญาตให้ผ่านหรือไม่โดยดูจากข้อมูลในแพ็คเก็ตเอง เช่น ไอพีแอดเดรส (IP address), พอร์ต (Port) และโพรโทคอล (Protocol) ซึ่งเป็นข้อมูลเชิงพื้นฐานเกี่ยวกับแพ็คเก็ต

Firewall ถูกใช้เพื่อป้องกันการเข้าถึงไม่พึงประสงค์จากอินเทอร์เน็ตหรือเครือข่ายภายนอก และยังช่วยป้องกันการรั่วไหลข้อมูลที่ล้นเหลือหรือไม่เหมาะสมภายในเครือข่ายองค์กร นอกจากนี้ยังช่วยป้องกันการโจมตีแบบ Denial of Service (DoS) หรือ Distributed Denial of Service (DDoS) ที่เป็นการโจมตีที่เป้าหมายคือการเข้าถึงเครือข่ายหรือเซิร์ฟเวอร์ด้วยการก่อให้เกิดการรับข้อมูลมากเกินไปจนทำให้เครือข่ายหรือเซิร์ฟเวอร์ล้นเหลือไม่สามารถให้บริการได้อย่างปกติได้ รวมถึงช่วยในการบริหารจัดการนโยบายความปลอดภัยของระบบคอมพิวเตอร์องค์กรให้มีประสิทธิภาพและควบคุมการเข้าถึงข้อมูลเชิงบริการตามนโยบายที่กำหนดไว้

Firewall เป็นอุปกรณ์หรือซอฟต์แวร์ที่มีบทบาทสำคัญในการควบคุมและปกป้องความปลอดภัยของเครือข่ายคอมพิวเตอร์และระบบข้อมูลในโลกดิจิทัลขณะเชื่อมต่อกับอินเทอร์เน็ตหรือเครือข่ายภายนอก

5. SSL Encryption

SSL ย่อมาจาก Secure Sockets Layer มันเป็นมาตรฐานอุตสาหกรรมเพื่อความปลอดภัยที่เข้ารหัสข้อมูลส่วนบุคคลที่ปลายด้านหนึ่งโดยใช้ Public Key และเฉพาะเซิร์ฟเวอร์ปลายทางที่ต้องการซึ่งตรวจสอบ Public Key กับ Private Key ที่สอดคล้องกันบนเซิร์ฟเวอร์ต้นทางสามารถถอดรหัสข้อมูลเมื่อมาถึง เพื่อให้การจัดส่งปลอดภัยเหมือนนี้ SSL ต้องใช้ขั้นตอนจำนวนมากและการป้อนข้อมูลของบริการเพิ่มเติม

2.1 SSL Certificate Authorities (CA)

ขั้นตอนและบริการเหล่านี้เรียกรวมกันว่า Public Key Infrastructure (PKI) PKI ประกอบด้วยหน่วยงานที่ออกใบรับรองหรือที่เรียกว่าหน่วยงานรับรอง (CA) หน่วยงานผู้ลงทะเบียน (RA) ฐานข้อมูลใบรับรองและที่เก็บใบรับรอง

2.2 SSL Handshake

เมื่อผู้ใช้ที่เรียกว่า“relying party” ใน PKI ผู้เยี่ยมชมไซต์ที่ได้รับการปกป้องโดย SSL จะเห็นว่าการประกาศโปรโตคอลในแถบที่อยู่นั้นอ่าน HTTPS ส่วน HTTPS, SSL และ TLS เป็นการรวมกันของส่วนที่แข็งแกร่งที่สุดที่สามารถอนุญาตให้ใช้การเข้ารหัสบน HTTPS นอกจากนี้มีข้อบ่งชี้ในเบราว์เซอร์ Chrome แสดงการรูปแม่กุญแจสีเขียวและ Firefox จะแสดงการรูปแม่กุญแจสีเทาหน้า URL เมื่อคุณคลิกที่ล็อคเหล่านี้จะให้ข้อมูลเพิ่มเติมเกี่ยวกับใบรับรอง

ถึงแม้ว่าเราจะมี SSL ที่ช่วยให้เราส่งข้อมูลได้อย่างปลอดภัยในระดับหนึ่งแต่ในบางสิ่งที่ต้องระวังเมื่อใช้ SSL แต่ในภาพรวม SSL เป็นอนาคตของความปลอดภัยของข้อมูลบนอินเทอร์เน็ตที่ให้ประโยชน์และตัวเลือกมากมายเพื่อรักษาข้อมูลและธุรกรรมของคุณให้ปลอดภัย ในขณะที่ Google ได้ยกระดับ SSL เพื่อให้สถานะมีผลต่อการจัดอันดับผลการค้นหา ด้วยขนาดและอิทธิพลของ Google SSL คือและจะยังคงเป็นมาตรฐานอุตสาหกรรมต่อไป

6. Password encryption

Password Encryption เป็นการเข้ารหัสหรือการแปลงรหัสผ่านให้เป็นรหัสลับ เพื่อป้องกันไม่ให้ถูกอ่านหรือใช้โดยบุคคลอื่นได้ เพื่อป้องกันรหัสผ่านของคุณเมื่ออยู่ในเซิร์ฟเวอร์ โดยหนึ่งในวิธีการ encrypt password คือ

Hash

การ Hash หรือ Hashing ชื่ออย่างเป็นทางการคือ Cryptographic Hash คือการสร้างข้อมูลที่เป็นตัวแทนของข้อมูลที่ต้องการ ซึ่งอาจจะเป็นรหัสผ่าน หรือข้อมูลส่วนบุคคลอื่นๆ และนำไปจัดเก็บในฐานข้อมูลหรือใน Text file หรือในที่อื่นๆ ซึ่งข้อดีของการทำ Hash คือจะไม่สามารถถอดรหัส หรือกระทำการใดๆ เพื่อที่จะ Reverse ให้ออกมาเป็นข้อความต้นฉบับ ซึ่งในปัจจุบันมีวิธีการ Hash มากมาย เช่น MD5, SHA1, SHA256 เป็นต้น โดยจะยกตัวอย่างหนึ่งในการ hash คือ

Bcrypt algorithm

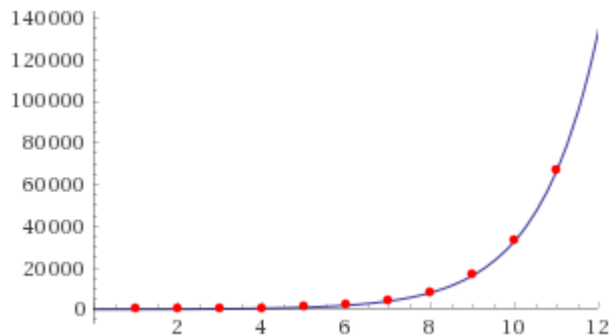
Bcrypt เป็น password hashing function ที่สร้างขึ้นจากพื้นฐานของ Blowfish cipher โดยการทำงานของ Blowfish cipher ที่การสร้าง key ใหม่ขึ้นมาจะต้องทำการ pre-process โดยใช้เวลาเทียบเท่ากับการเข้ารหัสตัวอักษรขนาด 4KB ซึ่งถือว่าช้ากว่า block cipher รูปแบบอื่นๆ ทำให้การทำ Brute-force มีความยากขึ้น และยังสามารถป้องกันการทำ Rainbow Table ได้ด้วย

Bcrypt ใช้คีย์ขนาดใหญ่ 56 บิตเพื่อเข้ารหัสข้อมูล และใช้ Salt ในการเพิ่มความซับซ้อนในการเข้ารหัส ซึ่ง Salt จะถูกสร้างขึ้นมาโดยอัลกอริทึมแบบสุ่ม แล้วนำ Key ที่ผู้ใช้ป้อนเข้ามาพร้อมกับ Salt ในการเข้ารหัส

กระบวนการเข้ารหัสแบบ Bcrypt ประกอบด้วยขั้นตอนดังนี้

- Bcrypt จะสุ่มค่า Salt ขึ้นมาเพื่อเพิ่มความซับซ้อนให้กับการเข้ารหัส โดย Salt จะเป็นข้อมูลที่ไม่วางกันและมีความยาวต่างกันไปตามการตั้งค่าของ Bcrypt ซึ่งค่า Salt จะถูกเก็บรวมกับรหัสผ่านที่ถูกเข้ารหัสแล้ว
- Bcrypt จะใช้ฟังก์ชัน Hashing เพื่อดำเนินการ Hash จากการนำรหัสผ่านที่ต้องการเข้ารหัสมาผ่านการ Hash ในรอบแรก โดยใช้ Salt ที่ถูกสุ่มมาเพื่อสร้าง Key ในการเข้ารหัส และผลลัพธ์ที่ได้จะเป็นค่า Hash ที่จะถูกนำไปใช้ในการเข้ารหัสในรอบถัดไป
- หลังจากได้ Hash มาแล้ว Bcrypt จะนำ Key ที่ถูกสร้างขึ้นจาก Salt มาใช้ในการเข้ารหัสด้วย Blowfish cipher โดยการแบ่งชุดข้อมูลออกเป็น Block ขนาด 64-bit และทำการเข้ารหัสแบบ Blowfish cipher โดยใช้ Key ที่สร้างขึ้นมา
- และสุดท้าย จะมีการทำซ้ำ โดยการนำค่า Hash ที่ได้มา สุ่ม Salt ขึ้นมาใหม่ และทำการเข้ารหัสซ้ำด้วย Salt ใหม่ตามจำนวนรอบที่กำหนด เพื่อให้ Hash ที่ได้มีความปลอดภัยและยากต่อการถอดรหัสมากขึ้น

Bcrypt จะมีการกำหนดระดับความปลอดภัย (Security Level) โดยเรียกว่า Cost factor ซึ่งจะมีค่าเริ่มต้นอยู่ที่ 10 เมื่อมีการเพิ่มระดับความปลอดภัยให้สูงขึ้น จะทำให้การคำนวณ Key และ Salt ใช้เวลานานขึ้น และจะทำให้การ Brute-force ใช้เวลานานขึ้นด้วย



Bcrypt เป็นวิธีการเข้ารหัสที่มีความปลอดภัยสูงและเหมาะสำหรับการเก็บข้อมูลสำคัญ เช่น รหัสผ่าน และข้อมูลที่ต้องการความมั่นคงปลอดภัยในระบบเครือข่าย การใช้ Bcrypt จึงเป็นที่นิยมในการพัฒนาระบบรักษาความปลอดภัยของเว็บไซต์และแอปพลิเคชันต่างๆ ในปัจจุบัน

7. Data Encryption *

8. Audit



เมื่อไม่นานที่ผ่านมาได้มีการโจมตีของ Ransomware ที่รู้จักกันในชื่อ WannaCry และ NotPetya ได้เปลี่ยนภาพลักษณ์ความปลอดภัยทางไซเบอร์ตลอดกาล WannaCry เป็นการโจมตีแบบหลายเส้นทาง ครั้ง

แรกที่มีการติดไวรัสอย่างรวดเร็วมากกว่า 200,000 เครื่องใน 150 ประเทศทำให้เกิดความเสียหายและความเสียหายหลายพันล้านดอลลาร์

การเปลี่ยนแปลงพื้นฐานที่นี้คือความจริงที่ว่านักแฮกเกอร์ ใช้เครื่องมือ แฮ็คระดับทหารที่พัฒนาโดย National Security Agency เพื่อกำหนดเป้าหมายทุกคน สำหรับวิสาหกิจขนาดกลางและขนาดย่อม (SMEs) ผลที่ตามมามีความสำคัญเนื่องจากค่าเฉลี่ยของการรั่วไหลของข้อมูลเพียงครั้งเดียวอาจสูงถึงเสียหายถึง 360,000 บาท

การโจมตีไซเบอร์ในปัจจุบันยังคงมีอย่างต่อเนื่องในองค์กรหรือผู้ประกอบการธุรกิจที่เก็บข้อมูลสำคัญเป็นจำนวนมาก จึงมีความจำเป็นต้องดำเนินการเพื่อปกป้องตัวเราเองจากการบุกรุกของแฮกเกอร์ เพื่อลดความเสียหายทางข้อมูล โดยกระบวนการนี้เริ่มต้นด้วย Security Audit Security Audit เป็นการประเมินความปลอดภัยของระบบข้อมูลของ บริษัท อย่างเป็นระบบโดยการวัดความสอดคล้องตามเกณฑ์ที่กำหนด การตรวจสอบโดยทั่วไปจะประเมินความปลอดภัยของการกำหนดค่าทางกายภาพและสภาพแวดล้อมของระบบ ซอฟต์แวร์กระบวนการจัดการข้อมูลและการปฏิบัติของผู้ใช้ การตรวจสอบความปลอดภัยโดยใช้เพื่อกำหนดความสอดคล้องของกฎระเบียบในการออกกฎหมาย (เช่น HIPAA, Sarbanes-Oxley Act และ California Information Breach Act) ซึ่งระบุว่าคุณ้องค์กรต้องจัดการกับข้อมูลอย่างไร

ก่อนที่จะทำการตรวจสอบความปลอดภัยที่รักษาความปลอดภัยจะต้องตัดสินใจเกี่ยวกับขอบเขตของการวิเคราะห์การตรวจสอบความปลอดภัยทั่วไปจะประเมินสิ่งต่อไปนี้

- 1) Bring-your-own-device initiatives
- 2) Data- and access-related items (like cards, passwords, and tokens)
- 3) Email
- 4) Hardware configurations
- 5) Information-handling processes
- 6) Network
- 7) Physical configuration of the system and environment
- 8) User practices
- 9) Smart devices
- 10) Software configurations

การตรวจสอบควรประเมินความเสี่ยงในอดีตกับความเสี่ยงในอนาคต ซึ่งหมายความว่าที่รักษาความปลอดภัยควรได้รับการปรับปรุงเกี่ยวกับแนวโน้มความปลอดภัยล่าสุดและมาตรการที่ดำเนินการโดยองค์กรอื่นๆ เพื่อตอบสนองต่อพวกเขา

ในตอนท้ายของ Security Audit จะมีการรวบรวมรายงานเชิงลึกซึ่งครอบคลุมจุดแข็งและจุดอ่อนของการจัดการความปลอดภัยในปัจจุบันของคุณ เมื่อใดก็ตามที่มีการระบุช่องโหว่ค่าใช้จ่ายในการรักษาความปลอดภัยนั้นควรได้รับการประเมินเทียบกับค่าใช้จ่ายของการละเมิด

เก็บตามที่โปรโตคอลความปลอดภัยของคุณล้มเหลว (เปรียบเทียบกับแนວໂນ້ມ ແອັກລ່າສຸດ) จำเป็นที่จะต้องดำเนินการอย่างรวดเร็วเนื่องจากช่องโหว่เดียวอาจนำไปสู่การละเมิดข้อมูลที่สำคัญ

สำหรับ SMEs โดยเฉพาะอย่างยิ่งอาจดึงดูดใจเพราะขาดบุคลากรหรือทรัพยากรที่มีขนาดใหญ่ เพื่ออุทิศให้กับความมั่นคงทางไซเบอร์ อย่างไรก็ตามนี่คือสิ่งที่ทำให้ธุรกิจเหล่านี้เป็นเป้าหมายสำคัญ เมื่อธุรกิจไม่ใช้วิธีการเชิงรุกในการรักษาความปลอดภัยทางไซเบอร์นั้นแสดงที่ไม่ดีสามารถเจาะระบบของคุณและตรวจไม่พบเป็นระยะเวลานาน

9. Monitoring & Logging

Monitoring

การ monitoring คือการเฝ้าระวังปัญหาหรือข้อผิดพลาดที่อาจขึ้นได้กับระบบ ผู้รับผิดชอบสามารถตรวจสอบ ในบทนี้จะแสดงตัวอย่างทำระบบ monitoring สำหรับ Spring Boot Application

เครื่องมือที่ใช้ประกอบไปด้วย

- Spring boot application ซึ่งจะมี 2 service คือ User service กับ Order service
- Prometheus สำหรับจัดเก็บข้อมูลของแต่ละ service ในรูปแบบ time series
- Grafana สำหรับแสดงข้อมูลในรูปแบบ graph ที่สวยงาม
- มาดูขั้นตอนการสร้างระบบ Monitoring อย่างง่ายกันดู

ขั้นตอนที่ 1 พัฒนา Service ต่าง ๆ ด้วย Spring Boot

เริ่มต้นด้วยการพัฒนา service ด้วย Spring Boot ซึ่งมีอยู่ด้วยกัน 2 service คือ

1. User Service มีอยู่ 1 api คือ ดึงข้อมูลผู้ใช้งานตาม id
2. Order Service มีอยู่ 1 api คือ ดึงข้อมูลการสั่งซื้อของผู้ใช้งานแต่ละคน

ทั้งสอง API นั้นถ้าไม่พบข้อมูลจะ return code 404 กลับมา (Not found)

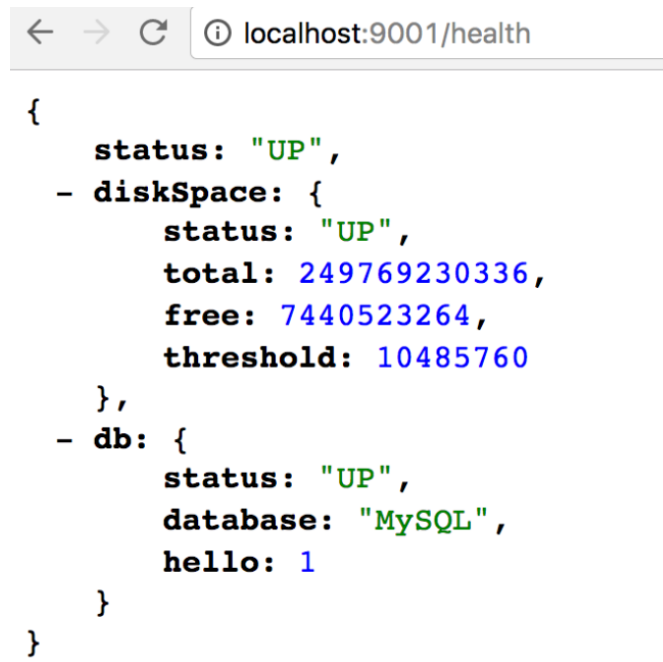
โดยปกติแล้วนั้น Service ที่พัฒนาด้วย Spring Boot จะยังไม่มีส่วนของ Metrics ต่าง ๆ ให้เราจะต้องเพิ่ม dependency ชื่อว่า Actuator ก่อนดังนี้

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-actuator</artifactId>
4 </dependency>
```

จาก Dependency นี้ทำให้ Service ของเรานั้นมี endpoint ต่าง ๆ ดังต่อไปนี้

- /health แสดงสถานะของ service เช่น uptime, disk และ database ที่ใช้งาน
- /info แสดงรายละเอียดของ service

- /metrics แสดงรายละเอียดของ server เช่นการใช้งาน CPU, Memory และพวก Heap size เป็นต้น
- /trace แสดงการ access มายัง service ว่าเข้ามาใช้งาน api/endpoint อะไรบ้าง
- ตัวอย่างของ health ดังรูป



```

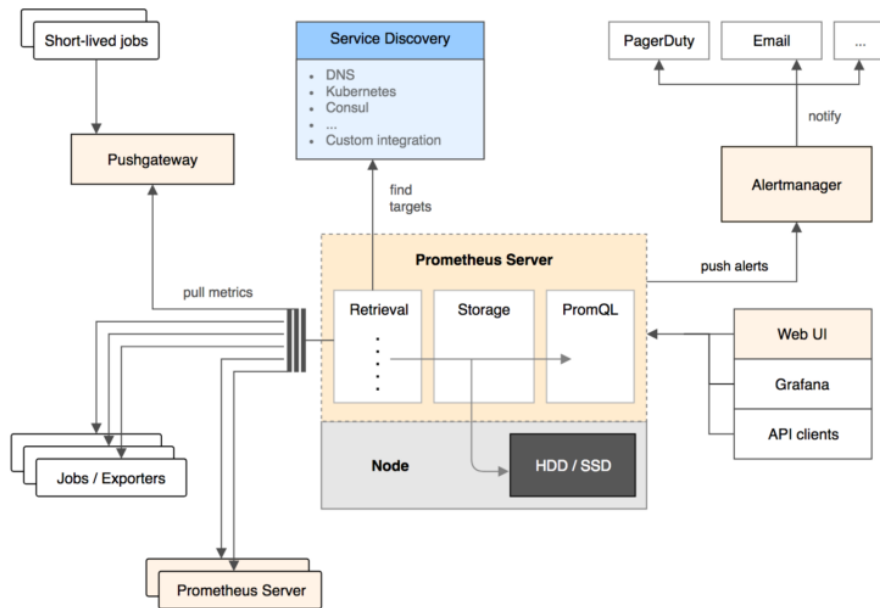
{
  status: "UP",
  - diskSpace: {
    status: "UP",
    total: 249769230336,
    free: 7440523264,
    threshold: 10485760
  },
  - db: {
    status: "UP",
    database: "MySQL",
    hello: 1
  }
}

```

ปัญหาที่ตามมาคือ ถ้าเราต้องการเก็บข้อมูลการใช้งานต่าง ๆ ของ Service ในรูปแบบ Time Series จะต้องทำอย่างไร ตัวอย่างเช่น API แต่ละตัวนั้นในแต่ละนาที่มีการใช้งานอย่างไรทั้งสำเร็จและไม่สำเร็จ

ขั้นตอนที่ 2 ทำการเก็บข้อมูลของ service จากข้อ 1 ลงใน Prometheus

หลังจากการศึกษาแบบคร่าว ๆ พบว่า สามารถจัดเก็บข้อมูลการใช้งานของแต่ละ Service แบบง่าย ๆ ลงใน Prometheus ได้ แต่ต้องสร้าง metric หรือข้อมูลให้ตรงตามรูปแบบของ Prometheus ด้วย มีสถาปัตยกรรมดังรูป



ตัวอย่างของรูปแบบข้อมูลที่จัดเก็บลงใน Prometheus มีรูปแบบตาม metric ของ Prometheus นั่นเอง ดูที่ metric url ของ Prometheus server ดังนี้

```

localhost:9090/metrics

# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.283e-05
go_gc_duration_seconds{quantile="0.25"} 4.5941e-05
go_gc_duration_seconds{quantile="0.5"} 6.1313e-05
go_gc_duration_seconds{quantile="0.75"} 8.7742e-05
go_gc_duration_seconds{quantile="1"} 0.069210986
go_gc_duration_seconds_sum 0.163082625
go_gc_duration_seconds_count 396
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 106
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.2445376e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.799990536e+09
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.524577e+06
# HELP go_memstats_frees_total Total number of frees.

```

แต่ service ที่พัฒนาด้วย Spring Boot นั้นยังมีข้อมูลการใช้งานไม่ตรงตามที่ Prometheus ต้องการ ดังนั้นจำเป็นต้องทำการ Custom Service กันนิดหน่อย โชคดีที่มีคนช่วยทำไว้ให้แบบง่าย ๆ แต่ในบทความนี้จะใช้วิธีการง่าย ๆ คือ
เพิ่ม Dependency ดังนี้เข้าไปก็จบเลย

```
1 <dependency>
2   <groupId>com.moelholm</groupId>
3   <artifactId>prometheus-spring-boot-starter</artifactId>
4   <version>1.0.1</version>
5 </dependency>
```

โดย dependency ตัวนี้จะสร้าง endpoint ใหม่ชื่อว่า /prometheus ในแต่ละ service ซึ่งมีข้อมูลดังนี้

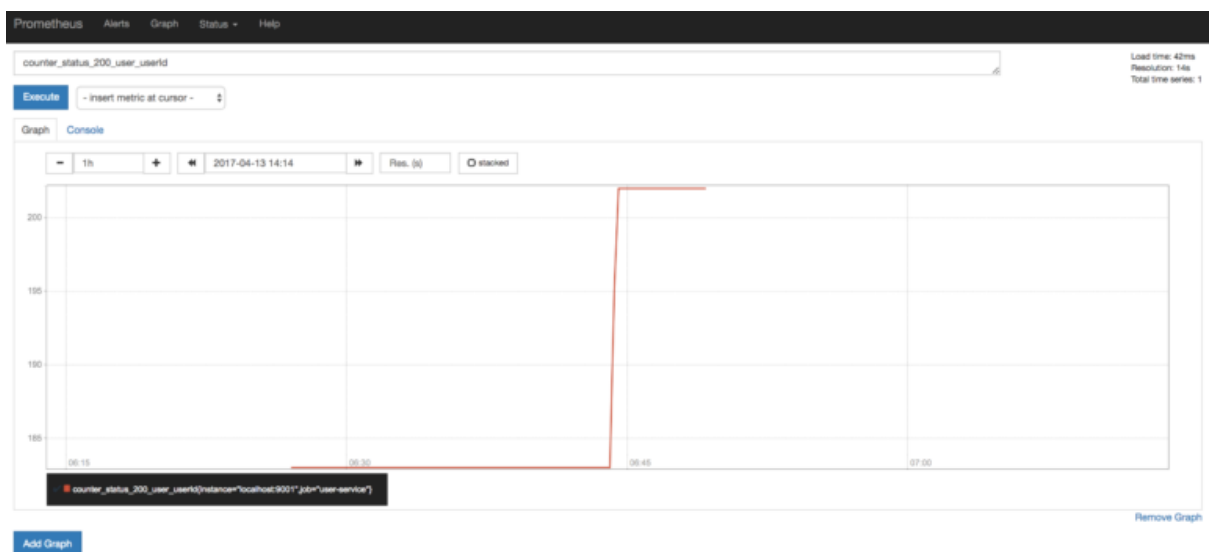
```
← → ↻ ⓘ localhost:9001/prometheus

# HELP httpsessions_max httpsessions_max
# TYPE httpsessions_max gauge
httpsessions_max -1.0
# HELP httpsessions_active httpsessions_active
# TYPE httpsessions_active gauge
httpsessions_active 0.0
# HELP datasource_primary_active datasource_primary_active
# TYPE datasource_primary_active gauge
datasource_primary_active 0.0
# HELP datasource_primary_usage datasource_primary_usage
# TYPE datasource_primary_usage gauge
datasource_primary_usage 0.0
# HELP mem mem
# TYPE mem gauge
mem 631154.0
# HELP mem_free mem_free
# TYPE mem_free gauge
mem_free 206838.0
# HELP processors processors
# TYPE processors gauge
processors 4.0
# HELP instance_uptime instance_uptime
# TYPE instance_uptime gauge
instance_uptime 5.227229E7
# HELP uptime uptime
# TYPE uptime gauge
uptime 2.5322107E7
```

จากนั้นทำการ configuration ใน Prometheus ให้ดึงข้อมูลจาก 2 service ให้ทำการดึงข้อมูลทุก ๆ 5 วินาที ด้วยการแก้ไขไฟล์ prometheus.yml ดังนี้

```
1 scrape_configs:
2   - job_name: 'user-service'
3     scrape_interval: 5s
4     metrics_path: '/prometheus'
5     static_configs:
6       - targets: ['localhost:9001']
7
8   - job_name: 'order-service'
9     scrape_interval: 5s
10    metrics_path: '/prometheus'
11    static_configs:
12      - targets: ['localhost:9002']
```

ต่อมาให้ลองทำการ query ข้อมูลที่ต้องการมาแสดงผลนิดหน่อย เพื่อตรวจสอบว่าการดึงข้อมูลถูกต้องตามที่ต้องการ เช่นต้องการดูข้อมูลการดึงข้อมูลจาก user service จาก metric ชื่อว่า counter_status_200_user_userid แสดงดังรูป



จาก graph ข้างต้นมีไว้สำหรับการทดสอบดึงข้อมูลมาแสดงผลตามที่ต้องการเท่านั้น แต่ถ้าต้องการสร้างเป็น Dashboard หรือ Graph แบบถาวรแล้ว มี 2 วิธีให้เลือกคือ

1. ใช้ Grafana
2. ใช้ Console Template

ขั้นตอนที่ 3 นำข้อมูลจาก Prometheus มาแสดงผลด้วย Grafana

มีขั้นตอนการใช้งานดังนี้

1. ทำการเพิ่ม Datasource เพื่อดึงข้อมูลจาก Prometheus

Add data source

Config Dashboards

Name: service ☐ Default

Type: Prometheus

Http settings

Url: http://localhost:9090

Access: direct

Http Auth

Basic Auth ☐ With Credentials ☐

Add Cancel

2. ทำการสร้าง Dashboard และเพิ่ม Graph ที่เราต้องการดู
ใน graph แต่ละตัวให้แสดงข้อมูลจาก metric เดียวไปเลย ดังนี้

- User service ในกรณีที่พบข้อมูลและไม่พบข้อมูล
- Order service ในกรณีที่พบข้อมูลและไม่พบข้อมูล

ตัวอย่างการ configuration ใน graph ของ User service เมื่อพบข้อมูล โดยใช้ metric
ชื่อว่า counter_status_200_user_userId แสดงข้อมูลในทุก ๆ 1 วินาที ที่สำคัญต้องกำหนด
datasource ไปยัง prometheus

แสดงดังรูป



อ้างอิงจาก: <http://www.somkiat.cc/monitoring-of-spring-boot/>

Logging

สำหรับองค์กรที่มีการลงทุนในเรื่องของระบบ Security มักจะคุ้นเคยกับสิ่งที่เรียกว่า Log และการจัดเก็บข้อมูลอย่างถูกต้องเหมาะสม เรามักมีคำถามว่าทำไมจึงจำเป็นต้องเก็บข้อมูล Log ด้วยหลายๆ ท่านอาจตั้งคำถามว่าเป็นการลงทุนในส่วนนี้เป็นการสิ้นเปลืองหรือไม่ เองบประมาณไปจัดซื้อ Security Control หรืออุปกรณ์อื่นๆ ที่จำเป็นจะดีกว่าไหม ด้วยข้อโต้แย้งต่างๆ ที่มีอยู่มากมายเกี่ยวกับการเก็บข้อมูลจราจร หรือ การเก็บ Log ตามข้อกำหนดของ พรบ. คอมพิวเตอร์ ทำให้ผมตัดสินใจเขียนบทความฉบับนี้ขึ้นเพื่อแชร์ข้อมูลในแต่ละมุมมองที่เกี่ยวข้องกับเรื่องนี้

Log คืออะไร

ในความหมายที่สามารถอธิบายให้เข้าใจได้ง่าย Log ก็คือ ข้อมูลหรือสิ่งที่อุปกรณ์แต่ละตัวในระบบคอมพิวเตอร์พยายามจะบอกจะสื่อสารกับผู้ใช้งาน หรือแม้แต่ผู้ดูแลระบบ หรือ Security Professional ซึ่งในอุปกรณ์ทุกๆ ชนิดไม่ว่าจะเป็น คอมพิวเตอร์, เครื่องแม่ข่าย, อุปกรณ์ไฟลล์วอลล์ หรืออุปกรณ์รักษาความปลอดภัยต่างๆ ในระบบคอมพิวเตอร์ ล้วนมีความสามารถในการสื่อสารกับผู้ใช้งานผ่านช่องทางของ Log ด้วยกันทั้งสิ้น นั่นหมายความว่าอุปกรณ์ในระบบคอมพิวเตอร์มี Log สำหรับสื่อสารกับผู้ใช้งาน

อุปกรณ์คอมพิวเตอร์ก็เหมือนมนุษย์คนหนึ่งทำงานตามคำสั่งที่ได้รับมอบหมาย แน่นอนว่าระหว่างที่ทำงานย่อมมีปัญหา สิ่งที่ต้องการสื่อสารกับเพื่อนร่วมงาน หรือผู้ดูแลเช่นกัน เช่น อนุมัติของอุปกรณ์สูงเกินจนจะถึงระดับที่เป็นอันตราย, อุปกรณ์ไม่สามารถเชื่อมต่ออินเทอร์เน็ตได้ด้วยเหตุผลต่างๆ หรือ มีผู้ใช้บางคนพยายามสุ่ม Password เพื่อ Login เข้าที่เครื่องแม่ข่าย เป็นต้น

เมื่อเรารู้ว่าคอมพิวเตอร์ หรือ อุปกรณ์ต่างๆ พยายามบอกอะไรเรา ก็จะทำให้เราสามารถแก้ไขปัญหา หรือป้องกันไม่ให้อุปกรณ์เกิดขึ้นได้ นอกจากนี้ Log สำหรับบางระบบยังสามารถช่วยเหลือเราในงานด้านการ Audit หรือ Review ข้อมูลการใช้งานของผู้ใช้งาน และสามารถใช้ในการงานด้านความมั่นคงปลอดภัยให้กับระบบงานได้เช่นกัน ในระบบ Security ข้อมูล Log นับเป็นสิ่งที่สำคัญมากสิ่งหนึ่งในการกำกับดูแลความมั่นคงปลอดภัยด้าน Security (Security Governance)



จะทำงานร่วมกับ Log และใช้ประโยชน์จากข้อมูลเหล่านี้ได้อย่างไร

สิ่งแรกที่เราจะต้องคำนึงถึง คือ ข้อมูล Log มีกระจายอยู่ตามอุปกรณ์ต่างๆ ทั่วทั้งองค์กรโดยอยู่ตามเครื่องคอมพิวเตอร์, เซิร์ฟเวอร์, อุปกรณ์เน็ตเวิร์คต่างๆ และอีกมากมายหลายที่ การที่เราจะ

ทำการ review แบบ one-by-one ย่อมเป็นไปได้ยากมีโอกาที่จะทำให้สำเร็จ หรือ เกิดประสิทธิภาพ (efficiency) ดังนั้นการที่จะทำให้สามารถใช้ประโยชน์จากข้อมูลที่กระจัดกระจายอยู่นี้ทำได้โดยจัดการให้ข้อมูลทุกอย่างนั้นมาอยู่ที่เดียวกัน เช่นการมีระบบศูนย์กลางในการบริหารจัดการข้อมูลจราจร หรือ ที่เรียกกันว่า Centralized Log Management (SIM/SIEM) เมื่อข้อมูลทุกอย่างอยู่ที่ศูนย์กลางแล้วก็สามารถบริหารจัดการได้โดยง่าย สามารถมองเห็นภาพรวมของสถานะ และคาดการณ์ถึงแนวโน้มด้านความปลอดภัยของระบบรวมถึงการวิเคราะห์วางแผนการป้องกันปัญหาที่คาดว่าจะเกิดขึ้น

ประโยชน์ของการนำเอา Log มาเก็บไว้ที่ศูนย์กลางโดยใช้เทคโนโลยี SIEM (Security Information and Event Management) คือ เราสามารถมองเห็นความสัมพันธ์ของเหตุการณ์ต่างๆ ที่เกิดขึ้น ทำให้สามารถมองเห็นปัญหา root cause รวมไปถึงการคาดเดาเหตุการณ์ที่จะเกิดขึ้นในอนาคตได้ การสร้างความสัมพันธ์ของเหตุการณ์ต่างๆ ในลักษณะนี้เราจะเรียกว่า Event Correlation ในระบบ SIEM หลายๆ ตัวยังสามารถกำหนดการแจ้งเตือนไปยังผู้ดูแลระบบเพื่อให้ทราบถึงเหตุการณ์สำคัญบางอย่าง ทำให้การกำกับดูแลงานด้าน Security มีความ Proactive มากขึ้น

เหตุผลด้านความปลอดภัยที่ไม่ควรเก็บ Log ไว้ที่ Local

หากไม่พูดถึงในแง่ของข้อบังคับของ พรบ. คอมพิวเตอร์ ที่ระบุว่าจะต้องทำการเก็บข้อมูล Log ไว้ที่ส่วนกลางแล้ว เหตุผลสนับสนุนในด้านของความมั่นคงปลอดภัยก็คือ หากไม่เก็บไว้ที่ระบบศูนย์กลาง หรือ SIEM จะทำให้การดูแลยาก กระทบต่อความมั่นคงปลอดภัย อาทิเช่น มีโอกาสที่ Log จะสูญหาย ถูกเปลี่ยนแปลงโดยผู้ที่ไม่หวังดีที่ต้องการบิดเบือนข้อมูล หรือพยายามลบ Log ทั้ง ในกรณีที่ต้องกรอก Hacker เจาะระบบเข้ามาสิ่งหนึ่งที่ Hacker จะทำคือ ทำลายหลักฐานที่สามารถเชื่อมโยงถึงตนเองได้ การลบ Log เป็นสิ่งหนึ่งที่จะทำให้ Archive Goal ในส่วนนี้

ที่ระบบศูนย์กลาง หรือ SIEM ข้อมูล Log จะถูกดูแลอย่างดี โดยมีระบบป้องกันข้อมูล Log ที่อยู่บนตัว SIEM ไม่ว่าจะเป็นการทำ Hashing หรือ การเข้ารหัส Log เพื่อให้ไม่สามารถถูกแก้ไขได้ มีระบบ Access Control ที่จะป้องกันการเข้าถึงข้อมูล Log/Report ดังนั้น ข้อมูลที่อยู่บน SIEM สามารถนำไปใช้ตามกฎหมายได้ หากไม่มีระบบพวกนี้ป้องกันศาลจะไม่รับรองหลักฐานเหล่านี้

เก็บ Log อย่างไรให้ถูกต้องและเหมาะสม

หลายๆ ท่านอาจมีข้อสงสัยว่าจะต้องทำอะไรให้การบริหารจัดการ Log เป็นไปอย่างถูกต้องครบถ้วน เมื่อทำงานกับระบบที่ใหญ่ และมีจำนวนอุปกรณ์ที่เกี่ยวข้องจำนวนมาก เป็นเรื่องที่ฟังดูแล้วไม่ใช่เรื่องง่าย แต่ก็ไม่ใช่เรื่องยากเช่นกัน

“เก็บเท่าที่จำเป็น ไม่ต้องเก็บทุกอย่าง” การเก็บข้อมูล Log จากทุกๆ อุปกรณ์อาจเป็นภาระที่ใหญ่หลวงทั้งในเรื่องค่าใช้จ่าย และการดูแลรักษา การเก็บทุกอย่างหมายความว่าข้อมูลในส่วนที่ไม่จำเป็นติดมาด้วยซึ่งอาจเป็นจำนวนไม่น้อย

ลบทิ้งเมื่อไม่ใช่ โดยปกติเราควรยึดเอากำหนดระยะเวลาที่ใช้ในการเก็บรักษา (Retention Period) ตาม พรบ. คอมพิวเตอร์ เป็นหลัก คือ ต้องเก็บรักษาไว้ไม่น้อยกว่า 90 วัน เป็นอย่างน้อย ในบางองค์กรจำเป็นต้องเก็บรักษาไว้นานกว่านี้ด้วยเหตุผลในด้าน Business สำหรับเวลาดังกล่าวจะอ้างอิงถึง Log ที่เป็น Mandatory หรืออุปกรณ์หลักๆ ที่จำเป็นต้องเก็บ อาทิเช่น Firewall, DHCP, Authentication Server, Proxy หรือ Internet Gateway, เครื่องแม่ข่ายที่เปิดให้บริการกับบุคคลทั่วไป เป็นต้น ดังนั้น ในส่วนของ Log ที่ไม่ได้เป็นอุปกรณ์หลักนั้นสามารถกำหนด Retention Period ได้ตามความเหมาะสม เพื่อให้ไม่เป็นภาระในด้านค่าใช้จ่ายต่างๆ ที่จะเกิดขึ้น

10. Network Log

Network log หรือ บันทึกข้อมูลเครือข่าย เป็นบันทึกข้อมูลที่สร้างขึ้นโดยระบบเครือข่าย เพื่อติดตามและบันทึกกิจกรรมที่เกิดขึ้นในระบบเครือข่ายคอมพิวเตอร์นี้ สามารถใช้ network log เพื่อตรวจสอบประวัติการเชื่อมต่อของอุปกรณ์เครือข่าย, การส่งและรับข้อมูล, การเข้าสู่ระบบ, และอีกมากมายเพื่อการดูแลรักษาและควบคุมเครือข่าย รวมถึงการตรวจจับและตรวจสอบกิจกรรมที่อาจจะมีการละเมิดความปลอดภัยของระบบเครือข่ายด้วย

Network log สามารถรวมอินฟอร์เมชันเกี่ยวกับ IP แอดเดรส, พอร์ต, เวลาที่เกิดเหตุ, ข้อมูลการสื่อสาร, และข้อมูลอื่น ๆ ที่เป็นประโยชน์ในการวิเคราะห์และปฏิบัติตามกฎหมายหรือนโยบายของระบบเครือข่าย การตรวจสอบ network log เป็นองค์การสำคัญในการรักษาความปลอดภัยของเครือข่าย และการติดตามกิจกรรมที่เกิดขึ้นในเครือข่ายขององค์กร หรือระบบเครือข่ายของบุคคลทั่วไป

การตรวจสอบการแก้ไขและการบำรุงรักษา

Network log สามารถช่วยให้ผู้ดูแลระบบติดตามการแก้ไขที่มีการดำเนินการในเครือข่าย เช่น การปรับแต่งการตั้งค่าอุปกรณ์เครือข่ายหรือการปรับปรุงระบบเพื่อให้มันสามารถทำงานอย่างมีประสิทธิภาพมากขึ้น นอกจากนี้ยังช่วยในการตรวจสอบการบำรุงรักษาอุปกรณ์เครือข่ายเพื่อป้องกันปัญหาที่เป็นไปได้

การตรวจสอบความปลอดภัย

Network log เป็นเครื่องมือสำคัญในการตรวจสอบความปลอดภัยของระบบเครือข่าย เป็นทางออกเพื่อตรวจสอบกิจกรรมที่เป็นความผิดปกติหรือภัยคุกคาม เช่น การเข้าสู่ระบบที่ไม่ถูกต้อง, การพยายามเข้าถึงข้อมูลที่ไม่ได้รับอนุญาต, หรือการโจมตีแบบจำลองการเชื่อมต่อ (simulation attacks)

การวิเคราะห์ปัญหา

เมื่อเครือข่ายมีปัญหาหรือการล่มเครือข่าย (network outage) เกิดขึ้น network log สามารถให้ข้อมูลเพิ่มเติมเกี่ยวกับสาเหตุและขอบเขตของปัญหา เป็นประโยชน์ในการวิเคราะห์และแก้ไขปัญหาในเครือข่าย

การปฏิบัติตามกฎหมายและการเก็บรักษาข้อมูล

บางกฎหมายและข้อบังคับกำหนดให้บริษัทและองค์กรต้องเก็บบันทึกกิจกรรมในเครือข่าย เป็นระยะเวลาหนึ่ง เพื่อเป็นหลักฐานในกรณีการสืบสวนหรือตามข้อบังคับเช่น HIPAA (Health Insurance Portability and Accountability Act) ในสายงานด้านสุขภาพและ GDPR (General Data Protection Regulation) ในยุโรป

11. Grails Security

Grails คือเฟรมเวิร์กแพลตฟอร์มแอปพลิเคชันเว็บแบบแบ็ค-เอนด์ที่ใช้ภาษา Groovy และฐานข้อมูลเป็นตัวขับเคลื่อนที่สร้างบนแพลตฟอร์มของภาษา Java ซึ่งได้รับแรงบันดาลใจจากเฟรมเวิร์กอื่น ๆ และออกแบบมาเพื่อเพิ่มความสะดวกในการพัฒนาแอปพลิเคชันเว็บโดยลดความซับซ้อนของการใช้ Java เปล่า ๆ ในการพัฒนา

Grails Security ช่วยให้เราสามารถจัดการเรื่องความปลอดภัยในแอปพลิเคชัน รวมถึงการรับรองความถูกต้องของผู้ใช้ (authentication) และการควบคุมสิทธิ์การเข้าถึงข้อมูล (authorization) โดยใช้เทคนิคและวิธีการที่มีอยู่แล้วใน Grails และ Groovy

Grails Security มีการรวมระบบความปลอดภัยอย่างสมบูรณ์ในการสร้างแอปพลิเคชันที่ปลอดภัย รวมถึงการจัดการการเข้าถึงข้อมูลผ่านการกำหนดสิทธิ์ให้กับผู้ใช้และบทบาท (role-based access control) และการจัดการการเข้าถึงแอปพลิเคชันของผู้ใช้ผ่านการตรวจสอบการเข้าสู่ระบบ (authentication) ที่ปลอดภัย รวมถึงการจัดการการควบคุมการเข้าถึง API และบริการอื่น ๆ ในแอปพลิเคชันด้วยการใช้ตัวกลาง (middleware) และตัวกรอง (filters) เพื่อป้องกันการโจมตีต่าง ๆ เช่น CSRF (Cross-Site Request Forgery) และ XSS (Cross-Site Scripting) อย่างมีประสิทธิภาพ

Grails Security มีและให้เครื่องมือและคำสั่งที่สะดวกในการปรับแต่งและกำหนดค่าความปลอดภัย และช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่มีระบบความปลอดภัยอย่างมีประสิทธิภาพ และได้รับการป้องกันอย่างเหมาะสมตามความต้องการ

** <https://docs.grails.org/6.0.0/guide/security.html#securityPlugins> **

12. Penetration testing **

security audits และ vulnerability assessments เป็นการพยายามเจาะระบบของคุณ เช่นเดียวกับแฮกเกอร์ ในสถานการณ์สมมตินี้ผู้เชี่ยวชาญด้านความปลอดภัยจะพยายามทำซ้ำวิธีการเดียวกันโดยผู้มีหน้าที่ไม่ดีเพื่อตรวจสอบว่าโครงสร้างพื้นฐานด้านไอทีของคุณสามารถทนต่อการโจมตีที่คล้ายกันได้หรือไม่

บ่อยครั้งการทดสอบการเจาะจะเกี่ยวข้องกับการใช้หลายวิธีร่วมกันเพื่อลองและทำลายระบบ สิ่งนี้ทำให้มีประสิทธิภาพสูงเมื่อคุณจำลองวิธีการเดียวกับที่ใช้โดยนักแสดงที่ไม่ดีในโลกแห่งความจริง เมื่อคุณมีส่วนร่วมในการทดสอบการเจาะระบบคุณจะได้รับประโยชน์จากข้อมูลเชิงลึกเชิงลึกเกี่ยวกับช่องโหว่และเรียนรู้วิธีการใช้ประโยชน์จากจุดอ่อนเหล่านี้

ตัวอย่างเช่นในบางกรณีคุณอาจพบช่องโหว่เล็กน้อยที่สามารถข้ามได้ แต่การทดสอบการเจาะระบบจะช่วยให้คุณเข้าใจถึงความจริงที่ว่าช่องโหว่เล็ก ๆ น้อย ๆ หลายช่องสามารถรวมเข้าด้วยกันเพื่อประนีประนอมเครือข่ายทั้งหมด

การทดสอบการเจาะใช้เครื่องมือเชิงพาณิชย์และโอเพนซอร์สเพื่อระบุช่องโหว่ในโมเดลความปลอดภัย นอกจากนี้ยังเกี่ยวข้องกับการโจมตีเป้าหมายบนระบบเฉพาะโดยใช้ทั้งแบบอัตโนมัติและแบบแมนนวลเพื่อให้มั่นใจว่าช่องโหว่นั้นไม่ถูกตรวจจับ มีการทดสอบการเจาะหลายประเภท แต่บ่อยครั้งที่พวกเขาไม่ได้ถูกแบ่งออกเป็นสามรูปแบบ

13. Keycloak session timeouts

Session-Timeout หมายถึง จำนวนชั่วโมงที่แต่ละอุปกรณ์สามารถใช้งานได้อัตโนมัติ หากเกินจากเวลาที่กำหนดนี้จะต้อง Login ใหม่ เช่น กำหนดไว้ 10 ชั่วโมง เราจะสามารถเชื่อมต่อผ่านอุปกรณ์ใดๆ ได้ 10 ชั่วโมงติดต่อกัน หลังจากนั้นจะต้อง Login ใหม่

Keycloak สามารถควบคุมและจัดการ session timeouts ได้ผ่านการตั้งค่าในแท็บ Sessions ผ่านเมนู Realm settings

Master

Enabled

Action

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

- <

gin

Email

Themes

Keys

Events

Localization

Security defenses

Sessions

>

SSO Session Settings

SSO Session Idle

30

Minutes

SSO Session Max

10

Hours

SSO Session Idle Remember Me

0

Minutes

SSO Session Max Remember Me

0

Minutes

Client session settings

Client Session Idle

0

Minutes

Client Session Max

0

Minutes

Offline session settings

Offline Session Idle

30

Days

Offline Session Max Limited

Disabled

Login settings

Login timeout

30

Minutes

Login action timeout

5

Minutes

Save

Revert

การกำหนดค่า	คำอธิบาย
SSO Session Idle	ใช้สำหรับ OIDC Clients เท่านั้น หากผู้ใช้ไม่มีกิจกรรมใดๆ เกินระยะเวลานี้แล้ว

	<p>Session ของผู้ใช้จะถูกยกเลิก ค่าระยะเวลานี้จะรีเซ็ตเมื่อ Client ขอการยืนยันตัวตนหรือขอ Refresh Token</p> <p>Keycloak จะแสดงหน้าต่างระยะเวลา Timeout ก่อนจะยกเลิก Session</p>
SSO Session Max	ระยะเวลานานที่สุดก่อน Session จะหมดอายุ
SSO Session Idle Remember Me	<p>การตั้งค่าคล้ายกับ SSO Session Idle แต่เจาะจงไปยังเฉพาะกรณีที่ Login และเปิดการใช้งาน “Remember Me”</p> <p>ผู้ใช้ที่มีการเปิดใช้งาน “Remember Me” สามารถมีระยะเวลาได้นานขึ้น แต่ถ้าหากค่าของการตั้งค่านี้เป็น 0, ระยะเวลาไม่มีกิจกรรม จะใช้การตั้งค่าของ SSO Session Idle</p>
SSO Session Max Remember Me	<p>การตั้งค่าคล้ายกับ SSO Session Max แต่เจาะจงไปยังเฉพาะกรณีที่ Login และเปิดการใช้งาน “Remember Me”</p> <p>ผู้ใช้ที่มีการเปิดใช้งาน “Remember Me” สามารถมีระยะเวลาได้นานขึ้น แต่ถ้าหากค่าของการตั้งค่านี้เป็น 0, ระยะเวลาไม่มีกิจกรรม จะใช้การตั้งค่าของ SSO Session Max</p>
Client Session Idle	<p>ระยะเวลา Idle Timeout สำหรับ Client Session ถ้าผู้ใช้ Inactive เป็นระยะเวลานานกว่าเวลา Timeout</p> <p>Client Session นั้นจะถูกยกเลิกและ Refresh Token จะเพิ่มระยะเวลา Idle Timeout ซึ่ง</p>

	<p>การตั้งค่านี้จะไม่มีผลกับผู้ใช้ SSO Session</p> <p>หากการตั้งค่านี้มีค่าเป็น 0 จะใช้การตั้งค่าของ SSO Session Idle</p>
Client Session Max	<p>ระยะเวลาที่นานที่สุดก่อน Session จะหมดอายุ ซึ่งการตั้งค่านี้จะไม่มีผลกับผู้ใช้ SSO Session</p> <p>หากการตั้งค่านี้มีค่าเป็น 0 จะใช้การตั้งค่าของ SSO Session Max</p>
Offline Session Idle	<p>การตั้งค่านี้ใช้สำหรับการเข้าถึงโหมดออฟไลน์ ระยะเวลาที่ Session รอการยกเลิกโทเคน ออฟไลน์โดย Keycloak</p> <p>Keycloak จะแสดงหน้าต่างระยะเวลา Timeout ก่อนจะยกเลิก Session</p>
Offline Session Max Limited	<p>การตั้งค่านี้ใช้สำหรับการเข้าถึงในโหมดออฟไลน์</p> <p>หากเปิดใช้งานจะสามารถกำหนดระยะเวลาสูงสุดที่ Offline Token ยัง active</p> <p>แต่ถ้าปิดการใช้งาน office sessions จะไม่หมดอายุอีกเลย</p> <p>ถ้าตัวเลือกนี้ถูกเปิดการใช้งาน จะมีให้ตั้งค่าสำหรับ Offline Session Max และ Client Offline Session Max</p>
Offline Session Max	<p>การตั้งค่านี้ใช้สำหรับการเข้าถึงในโหมดออฟไลน์ กำหนดระยะเวลาที่นานที่สุดก่อนที่ Keycloak จะยกเลิก Offline Token และจะไม่สนใจ Activity ที่ผู้ใช้งานกำลังทำอยู่</p>

Login timeout	ระยะเวลาทั้งหมดที่มีการเข้าสู่ระบบมาใช้งาน หากกระบวนการยืนยันตัวตนใช้เวลานานกว่าเวลาที่ตั้งค่าไว้ ผู้ใช้ต้องยืนยันตัวตนใหม่อีกครั้ง
Login action timeout	เวลานานที่สุดที่ผู้ใช้สามารถใช้ได้ต่อหน้าใดหน้าหนึ่งของการยืนยันตัวตน

14. JWT token

JSON Web Token (JWT) เป็นมาตรฐานเปิด (RFC 7519) ที่เข้ามาแก้ปัญหาการส่งข้อมูลอย่างปลอดภัยระหว่างกัน โดยที่ถูกออกแบบไว้ว่า จะต้องมีความกระชับ (Compact) และครบถ้วนในตัวเอง (Self-contained) สำหรับการส่งข้อมูลระหว่างบุคคล อย่างปลอดภัยในรูปแบบของ JSON object ซึ่งข้อมูลนี้สามารถตรวจสอบและเชื่อถือได้เนื่องจากการเซ็นแบบดิจิทัล โดยใช้ secret (HMAC algorithm) หรือ public/private key (RSA/ECDSA) ซึ่งเหมาะจะใช้งาน ดังต่อไปนี้

Authorization

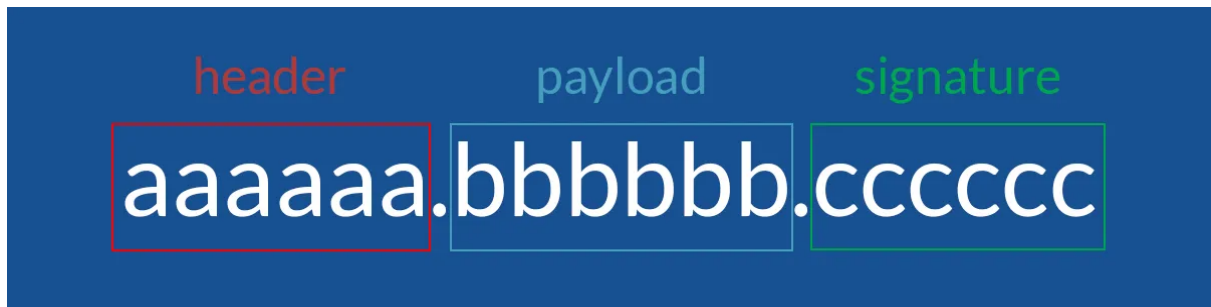
เป็นสถานการณ์ที่พบบ่อยที่สุดในการใช้ JWT เมื่อผู้ใช้เข้าสู่ระบบแล้ว ทุกคำขอติดตามแต่ละครั้งจะรวม JWT ด้วย เพื่อให้ผู้ใช้สามารถเข้าถึงเส้นทาง บริการ และทรัพยากรที่อนุญาตด้วยโทเค็นนี้ ซึ่งใช้กันอย่างแพร่หลายในปัจจุบันเนื่องจากมีค่าใช้จ่ายเพียงเล็กน้อยและสามารถใช้งานได้ง่ายแม้ในโดเมนที่แตกต่างกัน

Information Exchange

JSON Web Token เป็นวิธีที่ดีในการส่งข้อมูลระหว่างบุคคลอย่างปลอดภัย เนื่องจาก JWT สามารถเซ็นดิจิทัลได้ เช่น ใช้ public/private key คุณสามารถมั่นใจได้ว่าผู้ส่งคือคนๆนั้นจริง นอกจากนี้ เนื่องจากลายเซ็นถูกคำนวณโดยใช้ header และ payload จึงสามารถตรวจสอบได้ว่าเนื้อหาถูกดัดแปลงหรือไม่

JWT Structure

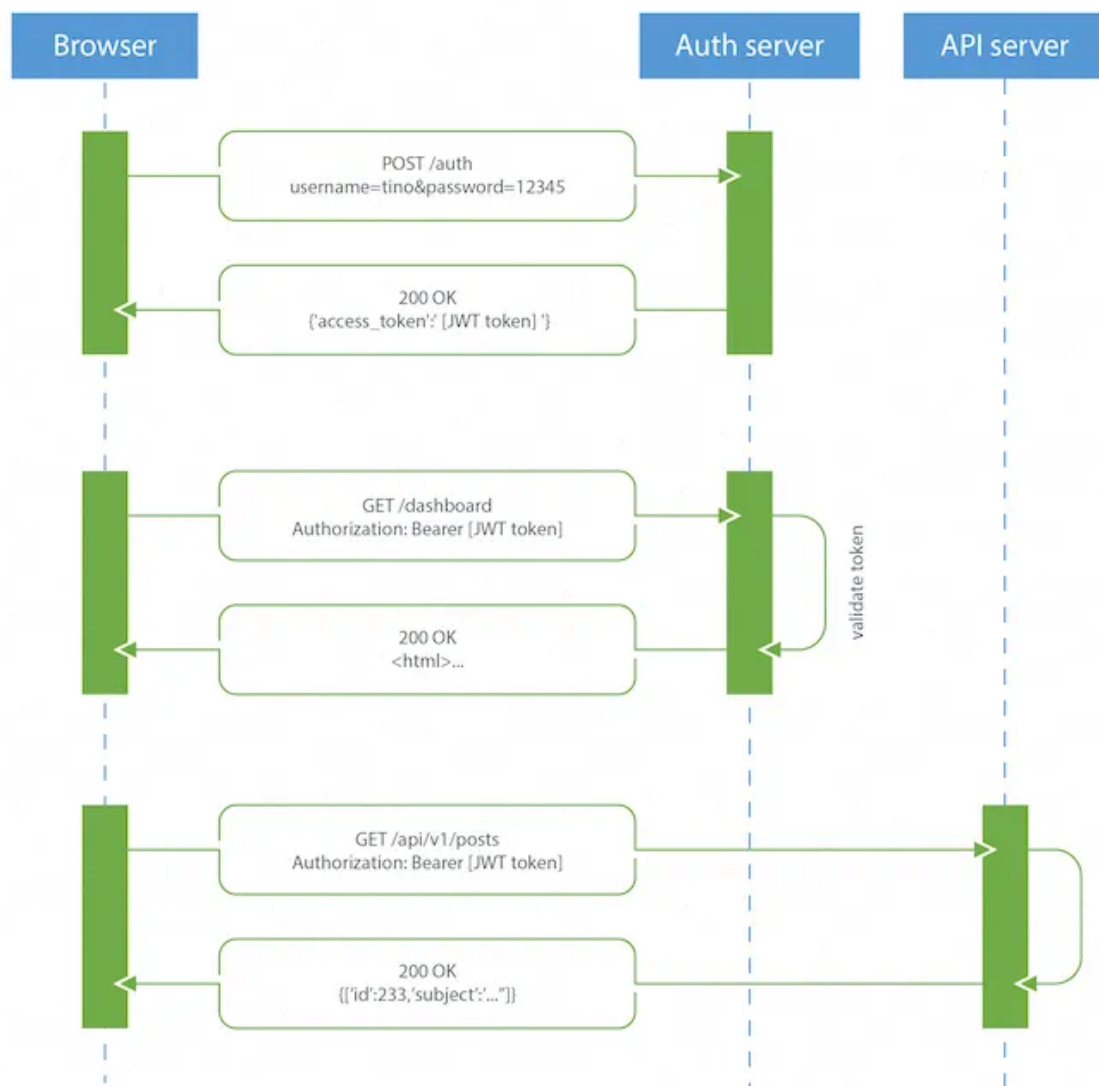
JWT เป็น Token หรือ ชุดตัวอักษรชุดหนึ่ง โดยมีโครงสร้างแบ่งออกเป็น 3 ส่วน ได้แก่



- 1.**Header:** ไว้เก็บว่า ข้อความชุดนี้ ใช้รหัสแบบไหนอยู่ (เช่น SHA256, RSA)
- 2.**Payload:** เก็บข้อมูลจริงๆ เช่น User ID, Roles ของผู้ใช้, E-mail ผู้ใช้ เป็นต้น
- 3.**Signature:** ส่วน Digital Signed ซึ่งเหมือนลายเซ็นทั้งท้าย ไว้ใช้ว่า เป็น Token ที่ถูกสร้างอย่างถูกต้องหรือไม่ เพราะหากมี Hacker ใจเปลี่ยนข้อมูลใน Payload ทำให้ Signature ไม่ตรง Token นั้นก็จะไม่นำมาใช้ เพราะเชื่อถือไม่ได้

JWT ทำงานอย่างไร?

เนื่องจาก JWT ถูกออกแบบมาให้มีขนาดกะทัดรัด เวลาใช้งาน เราสามารถพ่วง Token นี้ไปกับ HTTP Request ได้เลย ผ่าน Header หรือแม้กระทั่ง GET Parameter เมื่อ Server ได้รับ Token นี้ ก็จัดการเช็ค Signature ว่าถูกต้องไหม และเริ่มการถอดรหัส (Decrypt) เพื่อเอาข้อมูลออกมา เช่น User ID และ Roles ของ User ซึ่งการเก็บข้อมูลแบบนี้ Server จะไม่ต้องเก็บรายการ Session ID ทำให้ลดการทำงานลง เพราะไม่ต้อง Query หา User และ Permission เพิ่มประสิทธิภาพให้กับการทำงาน



toptal

จากภาพด้านบน Request แรกที่ทำการ Authentication, Server จะทำการสร้าง JWT แล้วส่งกลับ
ไปให้ Browser ทำการเซฟลง Localstorage ซึ่ง Server จะไม่เซฟ JWT ไว้ที่ Server เมื่อ User ทำการเรียก
Request ถัดไป เพื่อดึงข้อมูลส่วนตัว Browser ก็จะส่ง JWT กลับมาให้ Server ทำการ Decrypt ข้อมูล
จัดการเช็คสิทธิ แล้วส่งข้อมูลกลับไปให้ User ในกรณีที่มิสิทธิการใช้งาน ซึ่ง Server ไม่ต้อง Query หาข้อมูล
เพิ่มเติม สามารถนำ Payload มาใช้งานได้ทันที หรือหาก User ต้องการติดต่อ Server ตัวอื่น ก็สามารถใช้
JWT ชุดเดิมส่งไปให้

Server ตัวนั้น ก็เช็ค Signature และ Decrypt ข้อมูล เมื่อเช็คแล้ว พบว่าทุกอย่างถูกต้องก็เปิดข้อมูลส่วนตัว
นั้นให้ User ดูได้นั่นเอง

ข้อดีในการใช้ JWT

ประสิทธิภาพ (Performance)

เนื่องจาก JWT มีการเก็บข้อมูลในตัว (Self-contained) ทำให้ลดการทำงานซ้ำๆ ไปได้ เช่น การ Query หาข้อมูลผู้ใช้ หรือ Permission ของผู้ใช้ ซึ่งข้อมูลเหล่านี้ไม่ได้อันตรายและเสี่ยงต่อความปลอดภัย จึงสามารถเก็บลง Payload ได้ ช่วยลดการทำงาน เพิ่มประสิทธิภาพ เพราะไม่ต้อง Query หา Database ซ้ำๆ

การขยายตัว (Scalability)

ไม่ส่งผลกระทบต่อขยาย Server เพราะไม่มีการเก็บ Token ไว้บน Server และแยก Authentication ออกมาเป็น Service ไม่ต้องขึ้นอยู่กับโปรเจกต์ใดๆ สนับสนุนเรื่อง Modularity ทำให้ Scale App ในอนาคตได้ง่าย

CORS (Cross-origin Resource Sharing)

ไม่ต้องกังวลเรื่อง CORS เพราะไม่ต้อง Request ข้ามโปรเจกต์ เพื่อทำการ Authentication เพียงส่ง Token ไป Verify Signature หากทุกอย่างถูกต้อง ก็สามารถนำไปใช้ได้เลย

Cross-Site Request Forgery (CSRF, XSRF)

ไม่มีการเก็บ Token ไว้ใน Cookie แต่แนบไปกับ Header หรือ GET / POST Parameters ทำให้ไม่สามารถโดนโจมตีแบบ CSRF ได้

15. GitLab

GitLab เป็น Git Repository Manager ที่มีให้เลือกทั้งแบบ Software as a Service (SaaS) และ Self-Managed Host (On-Premises) ซึ่งช่วยในการบริหารจัดการ Source Code ของแต่ละโปรเจกต์ (Git Repository) และจัดการ CI/CD (Continuous Integration and Continuous Delivery) และยังมีมาพร้อม Feature ต่างๆ เช่น

- จัดการ Project หรือ Repository
- Pipeline, Jobs, Schedules, Environments สำหรับ CI/CD

- Graph, Charts สำหรับ Project หรือ Repository
- สร้าง Issues เพื่อแจ้งปัญหาต่างๆ
- การเขียน Wiki เพื่อเก็บเป็นความรู้ไว้สำหรับโปรเจกต์นั้นๆ
- เปิดตัวข้อสำหรับการพัฒนาความสามารถใหม่ๆ

ไปจนถึงด้านความปลอดภัยแบบเต็มรูปแบบของ DevSecOps เหมาะสำหรับทีม Developer ที่ต้องการพัฒนา workflow ของการพัฒนา Software ให้มีความรวดเร็วและคล่องตัว

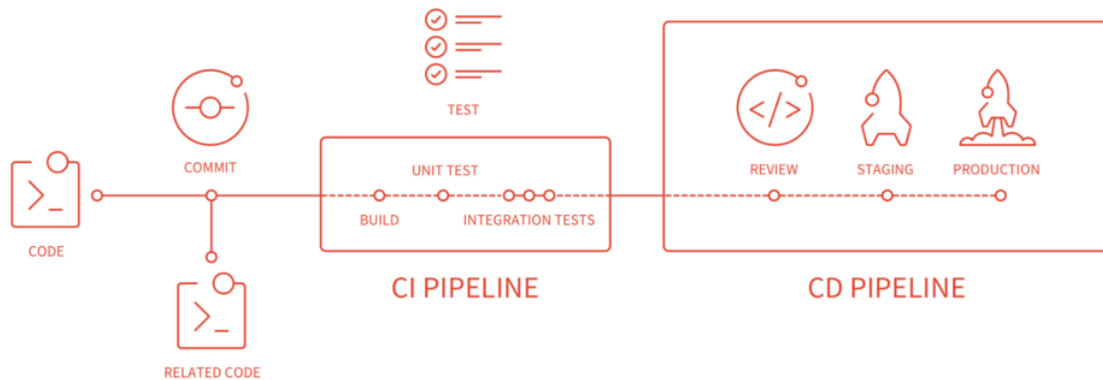


หนึ่งใน highlight สำคัญของ GitLab คือระบบควบคุมเวอร์ชัน ด้วย GitLab ทีมสามารถจัดการซอร์สโค้ด ติดตามการเปลี่ยนแปลง และทำงานร่วมกันบนโค้ดได้อย่างง่ายดาย GitLab ยังมีตัวติดตามปัญหาและเครื่องมือการจัดการโครงการในตัว ทำให้ง่ายต่อการติดตามข้อบกพร่อง งาน และโครงการ

Feature ยอดเยี่ยมอีกอย่างหนึ่งของ GitLab คือความสามารถในการผสานรวมและการปรับใช้อย่างต่อเนื่อง (CI/CD) ด้วย GitLab CI/CD ทีมต่างๆ สามารถทำให้ Workflow การพัฒนาซอฟต์แวร์เป็นแบบอัตโนมัติได้ ตั้งแต่การสร้างและทดสอบโค้ด การ monitor ไปจนถึงการเพิ่มความปลอดภัยให้กับแอปพลิเคชันที่ได้พัฒนาด้วยการ Scan ช่องโหว่ในส่วนต่างๆ ทำให้ Deploy Application ได้อย่างรวดเร็ว ประหยัดเวลา ทำให้มั่นใจได้ว่าแอปพลิเคชันจะทันสมัยและปลอดภัยอยู่เสมอ

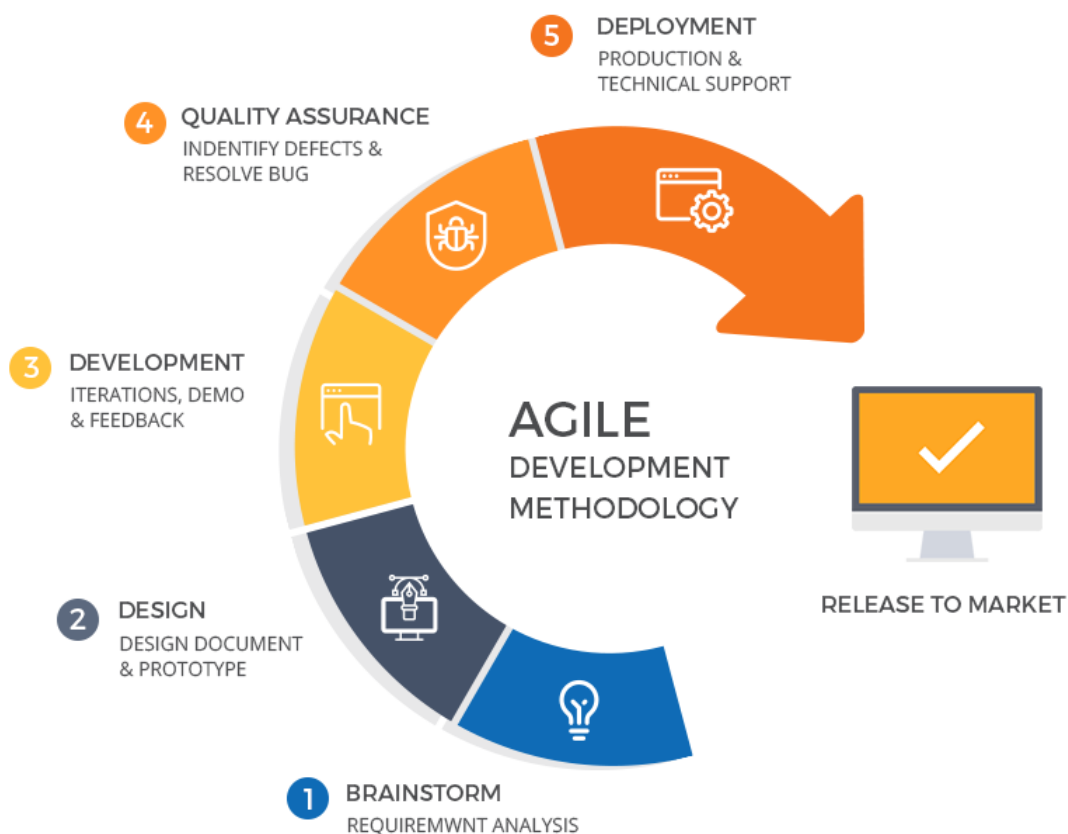


GitLab CI



นอกเหนือจากการควบคุมเวอร์ชันและความสามารถ CI/CD แล้ว GitLab ยังมีเครื่องมือสำหรับการทำ Code Quality Scan และ Code Security Scan ด้วยฟีเจอร์ต่างๆ เช่น การวิเคราะห์คุณภาพโค้ด การสแกนความปลอดภัย และการตรวจสอบโค้ด ทีมจึงมั่นใจได้ว่าโค้ดของตนมีคุณภาพสูงและปลอดภัย สามารถ Integrate เข้ากับเครื่องมือและบริการต่างๆ เช่น Jira, Slack, Microsoft Teams และอื่นๆ อีกมากมาย ทำให้ง่ายต่อการเชื่อมต่อ GitLab กับเครื่องมือและบริการอื่นๆ ที่ทีมใช้เป็นประจำทุกวัน พร้อมด้วยเครื่องมือสำหรับ Planning, Creating, Verifying, Packaging, Securing, Releasing, Configuring, และ monitoring สอบแอปพลิเคชันของคุณ สิ่งนี้ทำให้ทีมสามารถจัดการแอปพลิเคชันของตนได้อย่างง่ายดาย ตั้งแต่การพัฒนาไปจนถึงการนำขึ้น Production ในแพลตฟอร์มเดียว

16. Agile



Agile แนวทางในการพัฒนาซอฟต์แวร์ที่เน้นเรื่องการให้ความสำคัญกับตัวผู้ทำงานและการมีปฏิสัมพันธ์ระหว่างผู้ทำงาน มากกว่าขั้นตอนวิธีการหรือเครื่องมือ (Individuals and interactions over processes and tools) การสร้างซอฟต์แวร์ที่สามารถนำไปใช้งานได้จริง มากกว่าเอกสาร (Working software over comprehensive documentation) การทำงานร่วมกับลูกค้า มากกว่าการต่อรองสัญญา กับลูกค้า (Customer collaboration over contract negotiation) การยอมรับปรับซอฟต์แวร์ตามความเปลี่ยนแปลงของความต้องการของลูกค้า มากกว่าการทำตามแผนการเพียงอย่างเดียว (Responding to change over following a plan) ทั้งนี้ เรายังคงให้ความสำคัญต่อขั้นตอนวิธีการ เอกสาร สัญญา และแผนการดำเนินงานอยู่ เพียงแต่ ปฏิสัมพันธ์ระหว่างผู้ร่วมงาน ซอฟต์แวร์ที่ใช้ได้จริง การติดต่อกับลูกค้า และการปรับตามความเปลี่ยนแปลงนั้นมีความสำคัญยิ่งกว่า (Agile Alliance, 2001)

2.1 Agile Manifesto (แนวทางของ Agile)

จากหัวข้อที่แล้ว เราทราบว่าหัวใจหลักของ Agile นั้นเกิดจากแนวทาง 4 ข้อด้วยกัน ในหัวข้อนี้จึงจะกล่าวถึงความหมายของแนวทางแต่ละข้อนั้น

1) Individuals and interactions over processes and tools

ในองค์กรหรือหน่วยงานส่วนใหญ่นั้น มักสรรหากระบวนการหรือเครื่องมือราคาแพงที่จะคอยควบคุมคนให้ทำงานได้อย่างที่ต้องการ ซึ่งบ่อยครั้งเครื่องมือหรือกระบวนการเหล่านี้อาจไม่ได้มีความเหมาะสมต่อการนำมาใช้งานเลย เช่น หากต้องการแก้ไขโค้ดโปรแกรมชุดหนึ่งที่ทำอยู่กับนักพัฒนาอีกคน แทนที่จะใช้เครื่องมือสำหรับ review โค้ดโปรแกรมที่ถูกออกแบบมาให้สามารถดูได้ในทุกสถานที่ผ่านอินเทอร์เน็ต (ซึ่งไม่มีประโยชน์เลยหากนักพัฒนาที่ทำงานด้วยกันนั่งอยู่ในห้องเดียวกัน) เพียงแค่คุยกันหรือโทรศัพท์ติดต่อกัน (interact) แทนการใช้เครื่องมือยุ่งยากดังกล่าว (tool) ก็สามารถแก้ปัญหาได้อย่างรวดเร็วแล้ว

อีกหนึ่งปัจจัยที่จะทำให้ซอฟต์แวร์นั้นออกมาดี คือ แรงจูงใจในการทำงานของนักพัฒนา เมื่อใดที่นักพัฒนามีความสุขกับงานที่ได้ทำ เมื่อนั้นผลงานก็จะออกมาดี แต่เมื่อใดที่นักพัฒนาไม่สามารถมีความสุขกับงานที่ได้ทำ คุณภาพของงานที่ออกมา ก็มักจะลดลงไปด้วย ดังนั้น แทนที่จะใช้กระบวนการในการบังคับให้นักพัฒนาทำตามขั้นตอน ก็ควรจะหาวิธีให้นักพัฒนาได้ตระหนักถึงหน้าที่ของตน และอยากจะทำมันออกมาให้ดีด้วยตนเอง

2) Working software over comprehensive documentation

ลูกค้านั้นต้องการซอฟต์แวร์ที่สามารถใช้งานได้ มากกว่าเอกสารหรือคู่มือที่สมบูรณ์ (ที่ลูกค้ามักจะไม่ได้อ่าน) นั่นคือ ต่อให้สร้างเอกสารที่อ่านเข้าใจง่าย ครบถ้วนและสมบูรณ์แบบขึ้นมาได้ แต่ซอฟต์แวร์กลับทำงานไม่ได้ตามที่ลูกค้าต้องการ เอกสารเหล่านี้ก็จะหมดค่าลงทันที

แต่เอกสารนั้นยังคงมีความจำเป็นอยู่ ทั้งสำหรับการพัฒนาและดูแลระบบต่อไปในอนาคต รวมไปถึงการใช้เป็นหลักฐานในการติดต่อกับลูกค้า ดังนั้น จึงต้องพัฒนาซอฟต์แวร์ที่สามารถตอบสนองความต้องการของลูกค้าได้ขึ้นมาก่อน แล้วจึงค่อยสร้างเอกสารที่มีเนื้อหาเพียงพอต่อการพัฒนาระบบต่อไปในอนาคตขึ้นมา

3) Customer collaboration over contract negotiation

ในการพัฒนาซอฟต์แวร์ดั้งเดิมนั้น ลูกค้าจะมีบทบาทอยู่สองช่วง ได้แก่ช่วงแรกที่ทำหน้าที่เป็นผู้ให้

requirement และช่วงสุดท้ายที่จะรับส่งมอบงาน โดยที่ในระหว่างขั้นตอนการพัฒนาซอฟต์แวร์นั้น ลูกค้าแทบจะไม่ได้มีบทบาทใดๆเลย และเนื่องจากความต้องการของลูกค้านั้นมีการเปลี่ยนแปลงอยู่เสมอ ประกอบกับการที่บางครั้งลูกค้าเองยังไม่ทราบด้วยซ้ำว่าตนเองต้องการอะไร ซึ่งหากเราดันทำตาม requirement ที่ได้รับมาในตอนต้น ผลลัพธ์ที่ได้คือ ซอฟต์แวร์ที่ไม่สามารถทำงานได้ตรงตามความต้องการของลูกค้า

การแก้ปัญหานี้คือ ให้ลูกค้ามีส่วนร่วมในทุกๆขั้นตอนของการพัฒนา พนักงานพัฒนาเริ่มพัฒนาโปรแกรมที่สามารถทำงานได้แล้วในระดับหนึ่ง อาจจะเพียง 5% หรือ 10% ของความสามารถทั้งหมด ก็ให้เริ่มนำเสนอต่อลูกค้าในทันที เพื่อให้ลูกค้าเกิดโอเคเสียว่านี่คือสิ่งที่ตนต้องการ

จริงหรือเปล่า หากไม่ใช่ ทางทีมพัฒนาจะได้เตรียมรับมือหาแนวทางแก้ไขกันไป โดยในท้ายที่สุดแล้ว ซอฟต์แวร์ที่เกิดขึ้นมาจะทำงานได้ตรงตามความต้องการของลูกค้ามากที่สุด

4) Responding to change over following a plan

สืบเนื่องจากข้อที่แล้ว คือ ความต้องการที่เปลี่ยนแปลงอยู่ตลอดเวลาของลูกค้า ที่บางครั้งอาจทำให้เกิดความขัดแย้งกับแผนงานที่ได้เตรียมไว้ การยอมให้เกิดความเปลี่ยนแปลงในแผนที่ได้วางไว้ถือเป็นอีกสิ่งหนึ่งที่จะช่วยให้ซอฟต์แวร์ที่ออกมา นั้น ไม่กลายเป็นชิ้นงานที่ไร้คุณค่า เนื่องจากลูกค้าไม่สามารถนำไปใช้งานได้

2.2 ตำแหน่งที่สำคัญใน Agile

- **Stakeholders** ผู้มีส่วนเกี่ยวข้องกับผลิตภัณฑ์, ผู้ใช้งาน, เจ้าของบริษัท
- **Product Owner (PO)** ผู้ออกแบบผลิตภัณฑ์เพื่อตอบสนอง Stakeholders
- **Developer (Dev)** ผู้พัฒนาผลิตภัณฑ์ตามที่ออกแบบไว้ให้เกิดขึ้นจริง

2.3 วิธีการทำงานแบบ Agile

- เริ่มจาก Stakeholders มี Requirement (ปัญหาหรือความต้องการ) บางอย่าง
- PO อยากทำการแก้ไขปัญหาและตอบสนองความต้องการนั้น
- PO แปลง Requirement เป็น User Story เพื่อให้ Dev นำไปพัฒนาผลิตภัณฑ์
- Dev ออกแบบและพัฒนาผลิตภัณฑ์ตาม User Story ที่ได้รับ โดยมีตัววัดว่าสำเร็จคือ Acceptance Criteria
- Dev จะส่งมอบผลิตภัณฑ์ให้กับ Stakeholders นำไปใช้งาน
- Stakeholders อาจมี Feedback หรือความต้องการเพิ่มเติม
- วัดผลสิ่งที่ทำ จาก Feedback ที่ได้รับมา
- PO แปลงผลที่ได้เป็น User Story ใหม่ เพื่อให้ Dev นำไปปรับปรุงหรือพัฒนาผลิตภัณฑ์เพิ่มเติม

การทำอย่างไ้คือการลด Feedback Loop ดังกล่าวให้สั้นที่สุดเพื่อจะได้นำมาปรับปรุงได้อย่างรวดเร็ว นอกจากนี้ในความเป็นจริง Stakeholders มีความต้องการมากมายทำให้ User Story มีจำนวนมาก ในขณะที่ Dev มีความสามารถในการพัฒนาผลิตภัณฑ์จำกัด ทำให้ไม่สามารถตอบสนองความต้องการทั้งหมดนั้นได้ PO จึงต้องเป็นคนคอยกำหนดว่าจะทำอะไรก่อนทำอะไรหลัง หรือจะไม่ทำอะไรเพราะประโยชน์ที่จะได้รับไม่คุ้มค่ากับการลงมือทำ

2.4 ทีมงาน Agile

Stakeholders

ผู้มีส่วนเกี่ยวข้องกับผลิตภัณฑ์ ผู้ใช้งาน(End user), ผู้บริหารของบริษัท, บริษัทคู่สัญญาที่มาจ้างงานบริษัทเรา หลายบริษัทที่รับงาน Outsource อาจมีปัญหาว่าผู้ใช้งานกับคนตรวจรับงานต้องการผลิตภัณฑ์ที่ต่างกัน ทางที่ถูกต้องเราควรยึดผู้ใช้งานจริงเป็นหลักแต่ต้องหาข้อมูลมาสนับสนุนให้ได้ ว่าถ้าทำแบบนี้แล้วผู้ใช้งานจะใช้งานได้เร็วขึ้นกว่าแบบที่คนตรวจงานอยากได้เท่าไร ผลงานรวมต่อวันได้มากขึ้นเท่าไร มีประโยชน์อะไรจะได้รับมากขึ้น เป็นต้น

Product Owner

- เป็นคนอยากทำบางอย่างเพื่อแก้ไขปัญหาหรือตอบสนองความต้องการของ Stakeholders (Requirement)
- เปลี่ยน Requirement เป็น User Story และ Acceptance Criteria
- ทำให้ทุกฝ่ายเห็นภาพของ User Story ตรงกัน
- จัดลำดับความสำคัญของงานโดยคำนึงถึง 1. User Impact 2. Business Impact 3. Development Cost สำหรับข้อ 1 และ ข้อ 2 สามารถรู้ได้จากการทำวิจัย ส่วนข้อ 3 นั้นต้องถามทีมพัฒนาว่าพัฒนายากหรือง่าย ใช้เวลาในการพัฒนาเท่าไร
- แปลง User Story ที่มีขนาดใหญ่ให้มีขนาดเล็กลง และชัดเจนมากขึ้น
- ต้องปฏิเสธ Requirement บางอย่าง ไม่ปล่อยให้ Backlog ค้างมากเกินไป
- รักษาสมดุลระหว่างการพัฒนาฟีเจอร์ใหม่ การแก้ไขปัญหาค้าง (Bug) การอัปเดตระบบเดิมให้ดียิ่งขึ้น(Optimize)
- ในกรณีที่มีผลิตภัณฑ์หลายตัวแต่ทีมพัฒนามีจำกัด ต้องรักษาสมดุลในการให้ความสำคัญระหว่างการดูแลผลิตภัณฑ์เก่าและผลิตภัณฑ์ใหม่
- ลดความเสี่ยงทางธุรกิจ เทคโนโลยี ต้นทุนในการพัฒนา และเวลา โดยคำนึงถึงโอกาสที่จะเกิดขึ้น และผลกระทบที่จะได้รับหากเกิดขึ้น
- วางแผนการพัฒนาทั้งระยะสั้นและระยะยาว
- แต่ไม่มีอำนาจในการกำหนดว่าทีมพัฒนาต้องพัฒนาอย่างไร

Developer

- ประกอบไปด้วยตำแหน่งย่อย ดังนี้ Business Analyst, UX Designer, UI Designer, Developer, Quality Assurance, Operation โดยหนึ่งคนอาจมีบทบาทได้มากกว่า 1 ตำแหน่ง
- พัฒนาผลิตภัณฑ์โดยยึดตาม User Story
- งานจะเสร็จเมื่อ User Story นั้น ผ่าน Acceptance Criteria
- พัฒนา Unit Test และ Automate Test สำหรับ Acceptance Criteria

- พัฒนาระบบ Continuous Integration และ Continuous Delivery เพื่อให้สามารถรวบรวมผลิตภัณฑ์จากทีมต่างๆและส่งมอบให้ผู้ใช้งานได้ง่ายและรวดเร็ว
- พัฒนา Internal Tools เพื่อช่วยในการทำงานให้ง่ายและเร็วมากขึ้น
- พัฒนาทักษะเทคนิคคอล โดยทำวิจัยหรือ Proof of Concept (POC) ก่อนลงมือพัฒนาจริง
- ออกแบบและพัฒนา Architecture ให้เหมาะสมและดีขึ้น
- รู้ขีดจำกัดของตัวเองและทีม เช่น Story Point Limited per Sprint

2.5 ผลงานของตำแหน่งในทีมพัฒนาผลิตภัณฑ์แบบ Agile

- **Stakeholders** แสดงความต้องการ (Requirement)
- **Product Owner (PO)** นำ Requirement มาแปลงเป็น User Story
- **Business Analyst (BA)** นำ User Story มาขยายเป็น Behavior Driven Development (BDD) เพื่อให้เข้าใจพฤติกรรมของผู้ใช้งาน และหา Solution ให้กับ User Story นั้น โดยคำนึงถึงการใช้งานส่วนใหญ่ทั้งแบบปกติและไม่ปกติ (Happy and Unhappy Case)
- **UX Designer (UX)** นำ BDD มาออกแบบพฤติกรรมการใช้งาน ความรู้สึกในการใช้งาน User Journey และ Wireframe
- **UI Designer (UI)** นำ Wireframe มาออกแบบหน้าตาการใช้งานที่ผู้ใช้งานจะเห็น
- **Developer (Dev)** นำ Design ที่ออกแบบไว้มาพัฒนาเป็นผลิตภัณฑ์จริง
- **Quality Assurance (QA)** นำ BDD มาขยาย Happy และ Unhappy Case ให้ครอบคลุมพฤติกรรมการใช้งานทุกรูปแบบที่เป็นไปได้ให้ได้มากที่สุด นำผลิตภัณฑ์มาทดสอบให้เป็นไปตาม BDD / Test Case ที่ออกแบบไว้
- **Operation (Ops)** นำผลิตภัณฑ์ที่ผ่านการทดสอบแล้วไปส่งมอบให้ผู้ใช้งานสามารถเข้ามาใช้งานได้

2.6 Agile ที่ประกอบด้วยหลายทีม

ในแต่ละทีมพัฒนาจะต้องมี PO เป็นของทีมตัวเอง (PO 1 คน อาจดูแลหลายทีมได้) PO ของแต่ละทีมจะต้องมีการคุยเพื่อแลกเปลี่ยนข้อมูลกัน เพื่อลดการทำงานที่ซ้ำซ้อน หรือถ้ามีหลายทีมมากอาจมีหัวหน้า PO ขึ้นมาอีกคนเพื่อประสานงานระหว่าง PO ทีมต่างๆ

2.7 การจัดลำดับความสำคัญของการพัฒนาแบบ Agile

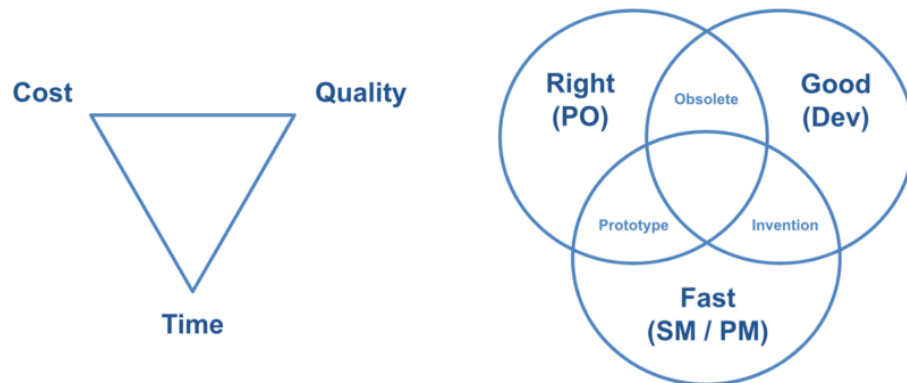
งานสำคัญและด่วน งานที่ต้องทำทันที ถ้าไม่เสร็จอาจเกิดปัญหาใหญ่ เช่น แก้ไขข้อผิดพลาดที่ทำให้ผู้ใช้งานไม่สามารถใช้งานได้ แก้ไขระบบล่ม

งานสำคัญและไม่ด่วน งานที่ต้องหาเวลาทำ เช่น วางแผนสปรินท์ พัฒนาฟีเจอร์ใหม่ ขยายหรืออัปเดตระบบเดิม

งานไม่สำคัญและด่วน งานที่ให้คนอื่นทำแทนได้ เช่น ตอบข้อสงสัยของผู้ใช้งาน

งานไม่สำคัญและไม่ด่วน งานที่ไม่ควรมี หรือทำในเวลาว่าง เช่น การประชุมที่เราไม่จำเป็นต้องเข้า
เมื่อพัฒนาในส่วนที่ 2 มากๆ จะทำให้งานในส่วนที่ 1 และส่วนอื่นๆ ลดน้อยลงไปด้วย

2.8 สมดุลของการพัฒนาผลิตภัณฑ์แบบ Agile

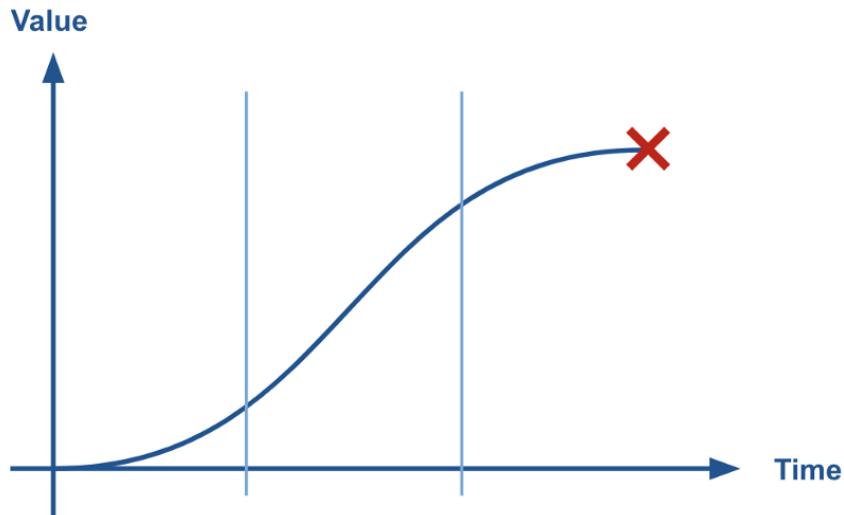


ทรัพยากรในการพัฒนาผลิตภัณฑ์มีจำกัดจึงต้องคำนึงถึงปัจจัยต่างๆ ทั้งต้นทุน คุณภาพ และเวลา ให้ออกมาตอบสนองผู้ใช้และตลาดได้อย่างทันเวลา โดย

- **Developer (Dev)** จะพยายามสร้างผลิตภัณฑ์ที่ดีที่สุด
- **Product Owner (PO)** จะพยายามทำผลิตภัณฑ์เพื่อตอบสนองผู้ใช้ให้มากที่สุด
- **Project Manager (PM)** จะพยายามส่งมอบผลิตภัณฑ์ให้ทันในเวลาที่กำหนด หรือ **Scrum Master (SM)** จะพยายามควบคุมให้ทีมทำงานได้ตามขั้นตอนในเวลาที่วางแผนไว้

หากทำผลิตภัณฑ์ดีและเสร็จเร็วแต่ไม่มีคุณภาพ ก็จะเป็นได้เพียงต้นแบบ ไม่สามารถใช้งานจริงได้ หากทำผลิตภัณฑ์มีคุณภาพและเสร็จเร็วแต่ไม่มีคนอยากใช้ ก็จะเป็นเพียงสิ่งประดิษฐ์ที่ไม่มีใครต้องการ หากทำผลิตภัณฑ์ดีและมีคุณภาพแต่ออกมาช้า ก็ไม่มีคนอยากใช้งานแล้ว กลายเป็นผลิตภัณฑ์ล้าสมัย ทั้งสามฝ่ายจึงต้องร่วมมือกัน หาจุดที่เหมาะสมที่ส่งผลกระทบต่อผู้ใช้ ต่อธุรกิจ และต้นทุนในการพัฒนาได้ลงตัวที่สุด

2.9 จุดสิ้นสุดของการพัฒนาผลิตภัณฑ์แบบ Agile



กราฟแสดงความสัมพันธ์ระหว่าง Value ในช่วงเวลาต่างๆของการพัฒนาผลิตภัณฑ์ด้วยไจล์ โดย Value ในที่นี้ประกอบไปด้วย Team Knowledge และ Customer and Business Value การพัฒนาผลิตภัณฑ์ด้วยไจล์จะแบ่งเป็น 3 ช่วงหลักๆ

ช่วงเริ่มต้นของการพัฒนา จะโฟกัสไปที่ความรู้(Knowledge) ของทีมเป็นหลัก เนื่องจากเป็นผลิตภัณฑ์ใหม่ ทีมอาจจะยังประเมินขนาดของงานและเวลาที่ใช้พัฒนาได้ไม่แม่นยำ รวมถึงอาจต้องใช้เทคโนโลยีใหม่ที่ไม่คุ้นเคยในการพัฒนาผลิตภัณฑ์ด้วย จึงเน้นให้ทีมมีความสามารถมากขึ้นในช่วงนี้ อาจมีการทำ Prototype หรือ Proof of Concept เพื่อหาเทคโนโลยีที่เหมาะสมที่จะนำไปพัฒนาผลิตภัณฑ์ รวมถึงการหาความต้องการของผู้ใช้งานและตลาดที่แท้จริง

ช่วงกลางของการพัฒนา จะโฟกัสไปที่ Customer and Business Value คือหลังจากที่ทีมมีความรู้ในการพัฒนาผลิตภัณฑ์ในระดับหนึ่งแล้ว และเข้าใจความต้องการของผู้ใช้และตลาดมากขึ้นแล้ว จะมุ่งพัฒนาผลิตภัณฑ์ให้ตอบสนองให้เกิดคุณค่ากับผู้ใช้ให้มากที่สุด

ช่วงปลายของการพัฒนา จะเป็นการพัฒนาผลิตภัณฑ์เพิ่มเติมที่อาจไม่ได้เพิ่มคุณค่าต่อผู้ใช้มากนัก เปรียบเสมือนเป็นพีแอร์แอม จนถึงจุดหนึ่งที่ไม่คุ้มค่าที่จะพัฒนาเพิ่มเติมก็จะทำการตัดจบเพื่อนำทรัพยากรไปพัฒนาผลิตภัณฑ์อื่นต่อไป แต่จะต้องยังคงดูแลระบบหรืออัปเดตระบบเพิ่มเติมจนกว่าจะปิดการใช้งานผลิตภัณฑ์นี้

2.10 การใช้งาน Agile ให้มีประสิทธิภาพ

- รับ Feedback ให้เร็วที่สุด เพื่อนำมาปรับปรุงผลิตภัณฑ์และการทำงาน
- ใช้ต้นทุนให้น้อยที่สุด เพื่อให้ได้คุณค่ามากที่สุด
- ยอมรับความจริงและข้อผิดพลาดที่เกิดขึ้น เพื่อให้สามารถนำไปปรับปรุงได้
- มีการพัฒนากระบวนการการทำงานและระบบเดิมด้วย ไม่ใช่ทำแต่ของใหม่

- คนในทีมต้องรู้ว่าทำส่วนนี้เพื่ออะไร ไม่ใช่ทำเพราะ Guideline บอกให้ทำ

17. Restful API

RESTful API

RESTful API เป็นอินเทอร์เฟซที่ระบบคอมพิวเตอร์สองระบบใช้เพื่อแลกเปลี่ยนข้อมูลผ่านอินเทอร์เน็ตได้อย่างปลอดภัย แอปพลิเคชันทางธุรกิจส่วนใหญ่ต้องสื่อสารกับแอปพลิเคชันภายในอื่นๆ และของบุคคลที่สามเพื่อทำงานต่างๆ ตัวอย่างเช่น หากต้องการสร้างสลิปเงินเดือน ระบบบัญชีภายในของคุณต้องแบ่งปันข้อมูลกับระบบธนาคารของลูกค้าเพื่อออกใบแจ้งหนี้และสื่อสารกับแอปพลิเคชันบันทึกเวลาปฏิบัติงานภายในโดยอัตโนมัติ RESTful API ให้การสนับสนุนการแลกเปลี่ยนข้อมูลนี้ เพราะเป็นระบบที่มีมาตรฐานการสื่อสารระหว่างซอฟต์แวร์ที่ปลอดภัย เสถียร และมีประสิทธิภาพ

API

ส่วนต่อประสานโปรแกรมประยุกต์ (Application Programming Interface หรือ API) กำหนดกฎที่คุณต้องปฏิบัติตามเพื่อสื่อสารกับระบบซอฟต์แวร์อื่น โดยนักพัฒนาเปิดเผยหรือสร้าง API เพื่อให้แอปพลิเคชันอื่นสามารถสื่อสารกับแอปพลิเคชันของตนได้ทางโปรแกรม ตัวอย่างเช่น แอปพลิเคชันบันทึกเวลาปฏิบัติงานแสดง API ที่ขอชื่อเต็มของพนักงานและช่วงวันที่ เมื่อได้รับข้อมูลนี้แล้ว ระบบจะประมวลผลบันทึกเวลาปฏิบัติงานของพนักงานเป็นการภายใน และส่งกลับจำนวนชั่วโมงที่ทำงานในช่วงวันดังกล่าว

ทั้งนี้คุณสามารถมองได้ว่า API เว็บเป็นเกตเวย์ระหว่างไคลเอ็นต์และทรัพยากรบนเว็บ

ไคลเอ็นต์

ไคลเอ็นต์คือผู้ใช้ที่ต้องการเข้าถึงข้อมูลจากเว็บ โดยไคลเอ็นต์อาจเป็นบุคคลหรือระบบซอฟต์แวร์ที่ใช้ API ก็ได้ ตัวอย่างเช่น นักพัฒนาสามารถเขียนโปรแกรมที่เข้าถึงข้อมูลสภาพอากาศจากระบบสภาพอากาศ หรือคุณสามารถเข้าถึงข้อมูลเดียวกันจากเบราว์เซอร์เมื่อคุณเยี่ยมชมเว็บไซต์รายงานสภาพอากาศได้โดยตรง

ทรัพยากร

ทรัพยากรคือข้อมูลที่แอปพลิเคชันต่างๆ มอบให้แก่ไคลเอนต์ โดยทรัพยากรอาจเป็นรูปภาพ วิดีโอ ข้อความ ตัวเลข หรือข้อมูลประเภทใดก็ได้ ทั้งนี้เครื่องคอมพิวเตอร์ที่มอบทรัพยากรให้แก่ไคลเอนต์นั้นเรียกอีกอย่างว่าเซิร์ฟเวอร์ องค์กรต่างๆ ใช้ API เพื่อแบ่งปันทรัพยากรและให้บริการเว็บในขณะที่ยังคงดูแลรักษาความปลอดภัย การควบคุม และการรับรองความถูกต้องไปพร้อมกัน นอกจากนี้ API ยังช่วยให้ลูกค้าระบุได้ว่าไคลเอนต์ใดสามารถเข้าถึงทรัพยากรภายในที่เฉพาะเจาะจงได้

REST

Representational State Transfer (REST) เป็นสถาปัตยกรรมซอฟต์แวร์ที่กำหนดเงื่อนไขว่า API ควรทำงานอย่างไร โดยแต่แรกเริ่มนั้น มีการสร้าง REST ขึ้นเพื่อเป็นแนวทางในการจัดการการสื่อสารบนเครือข่ายที่ซับซ้อน เช่น อินเทอร์เน็ต คุณสามารถใช้สถาปัตยกรรม REST เพื่อรองรับการสื่อสารที่มีประสิทธิภาพสูงและเชื่อถือได้ในทุกระดับ คุณยังสามารถใช้และปรับเปลี่ยนสถาปัตยกรรมได้อย่างง่ายดาย โดยนำความสามารถในการมองเห็นและการเคลื่อนย้ายข้ามแพลตฟอร์มมาสู่ทุกระบบ API

นักพัฒนา API สามารถออกแบบ API ได้โดยใช้สถาปัตยกรรมต่างๆ โดย API ที่เป็นไปตามรูปแบบสถาปัตยกรรม REST เรียกว่า REST API บริการเว็บที่ใช้สถาปัตยกรรม REST เรียกว่าบริการเว็บ RESTful คำว่า RESTful API โดยทั่วไปหมายถึง API เว็บแบบ RESTful อย่างไรก็ตาม คุณสามารถใช้คำว่า REST API และ RESTful API แทนกันได้

โดยหลักการบางประการของรูปแบบสถาปัตยกรรม REST มีดังต่อไปนี้:

อินเทอร์เฟซรูปแบบเดียวกัน

อินเทอร์เฟซรูปแบบเดียวกันถือเป็นพื้นฐานในการออกแบบบริการเว็บ RESTful ทุกประเภท ซึ่งระบุว่าเซิร์ฟเวอร์ถ่ายโอนข้อมูลในรูปแบบมาตรฐาน ทรัพยากรที่จัดรูปแบบเรียกว่าการแทนข้อมูลใน REST โดยรูปแบบนี้อาจแตกต่างจากการแทนข้อมูลภายในของทรัพยากรบนแอปพลิเคชันเซิร์ฟเวอร์ ตัวอย่างเช่น เซิร์ฟเวอร์สามารถจัดเก็บข้อมูลเป็นข้อความ แต่ส่งข้อมูลในรูปแบบการแทนข้อมูลด้วย HTML

โดยอินเทอร์เฟซรูปแบบเดียวกันกำหนดข้อจำกัดทางสถาปัตยกรรมไว้ 4 ประการ ได้แก่

- 1.คำขอควรระบุทรัพยากร ซึ่งสามารถทำเช่นนั้นได้โดยใช้ตัวระบุทรัพยากรรูปแบบเดียวกัน
- 2.ไคลเอนต์มีข้อมูลเพียงพอในการแทนข้อมูลทรัพยากรเพื่อแก้ไขหรือลบทรัพยากรดังกล่าวหากต้องการ เซิร์ฟเวอร์เป็นไปตามเงื่อนไขนี้โดยการส่งข้อมูลเมตาที่อธิบายทรัพยากรเพิ่มเติม
- 3.ไคลเอนต์ได้รับข้อมูลเกี่ยวกับวิธีการประมวลผลการแทนข้อมูลเพิ่มเติม เซิร์ฟเวอร์ดำเนินการเช่นนี้ได้โดยการส่งข้อความอธิบายตนเองที่มีข้อมูลเมตาเกี่ยวกับวิธีการที่ไคลเอนต์สามารถใช้งานข้อมูลดังกล่าวได้ดีที่สุด
- 4.ไคลเอนต์ได้รับข้อมูลเกี่ยวกับทรัพยากรอื่นๆ ที่เกี่ยวข้องทั้งหมดที่จำเป็นสำหรับการทำงานให้เสร็จสมบูรณ์ เซิร์ฟเวอร์ดำเนินการเช่นนี้ได้โดยการส่งไฮเปอร์ลิงก์ในการแทนข้อมูลเพื่อให้ลูกค้าสามารถค้นพบทรัพยากรเพิ่มเติมได้แบบไดนามิก

ความไร้สถานะ

ในสถาปัตยกรรม REST ความไร้สถานะหมายถึงวิธีการสื่อสารที่เซิร์ฟเวอร์ดำเนินการตามคำขอของไคลเอนต์ทั้งหมดโดยไม่ขึ้นกับคำขอก่อนหน้าทั้งหมด โดยไคลเอนต์สามารถร้องขอทรัพยากรในลำดับใดก็ได้ และทุกคำขอจะไร้สถานะหรือแยกออกจากคำขออื่นๆ ข้อจำกัดในการออกแบบ REST API นี้บ่งบอกว่าเซิร์ฟเวอร์สามารถเข้าใจและดำเนินการตามคำขอได้อย่างสมบูรณ์ทุกครั้ง

ระบบที่แบ่งออกเป็นชั้น

ในสถาปัตยกรรมระบบที่แบ่งออกเป็นชั้น ไคลเอนต์สามารถเชื่อมต่อกับตัวกลางอื่นๆ ที่ได้รับอนุญาตระหว่างไคลเอนต์และเซิร์ฟเวอร์ได้ และจะยังคงได้รับการตอบสนองจากเซิร์ฟเวอร์ เซิร์ฟเวอร์ยังสามารถส่งต่อคำขอไปยังเซิร์ฟเวอร์อื่นได้อีกด้วย คุณสามารถออกแบบบริการเว็บ RESTful ให้ทำงานบนเซิร์ฟเวอร์หลายตัวที่มีหลายชั้นได้ เช่น ความปลอดภัย แอปพลิเคชัน และตรรกะทางธุรกิจ โดยทำงานร่วมกันเพื่อตอบสนองคำขอของไคลเอนต์ โดยไคลเอนต์ไม่สามารถมองเห็นชั้นต่างๆ เหล่านี้ได้

ความสามารถในการแคช

บริการเว็บ RESTful รองรับการแคช ซึ่งเป็นกระบวนการจัดเก็บการตอบสนองบางส่วนบนไคลเอ็นต์หรือตัวกลางเพื่อปรับปรุงเวลาตอบสนองของเซิร์ฟเวอร์ ตัวอย่างเช่น สมมติว่าคุณเยี่ยมชมเว็บไซต์ที่มีรูปภาพส่วนหัวและส่วนท้ายทั่วไปในทุกหน้า ทุกครั้งที่คุณเยี่ยมชมหน้าเว็บไซต์ใหม่ เซิร์ฟเวอร์จะต้องส่งภาพเดิมอีกครั้ง เพื่อหลีกเลี่ยงปัญหานี้ ไคลเอ็นต์จะแคชหรือจัดเก็บรูปภาพเหล่านี้หลังจากการตอบสนองครั้งแรก จากนั้นจึงใช้รูปภาพโดยตรงจากแคช ทั้งนี้บริการเว็บ RESTful ควบคุมการแคชโดยใช้การตอบสนอง API ที่ระบุว่าสามารถแคชได้หรือไม่สามารถแคชได้

การปรับแต่งโค้ดได้ตามความต้องการ

ในรูปแบบสถาปัตยกรรม REST เซิร์ฟเวอร์สามารถขยายหรือปรับแต่งฟังก์ชันการทำงานของไคลเอ็นต์ได้ชั่วคราวโดยการถ่ายโอนโค้ดการเขียนโปรแกรมซอฟต์แวร์ไปยังไคลเอ็นต์ ตัวอย่างเช่น เมื่อคุณกรอกแบบฟอร์มลงทะเบียนบนเว็บไซต์ เบราวเซอร์ของคุณจะเน้นย้ำให้เห็นถึงข้อผิดพลาดของคุณทันที เช่น หมายเลขโทรศัพท์ที่ไม่ถูกต้อง ซึ่งระบบสามารถทำเช่นนี้ได้เนื่องจากโค้ดที่เซิร์ฟเวอร์ส่งไป

RESTful API มีข้อดีอะไรบ้าง

RESTful API มีข้อดีต่างๆ ดังต่อไปนี้:

ความสามารถในการปรับขนาด

ระบบที่ใช้ REST API สามารถปรับขนาดได้อย่างมีประสิทธิภาพเนื่องจาก REST ปรับการโต้ตอบระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ให้เหมาะสม ความไร้สถานะช่วยจัดการโหลดเซิร์ฟเวอร์เนื่องจากเซิร์ฟเวอร์ไม่จำเป็นต้องเก็บข้อมูลค่าขอของไคลเอ็นต์ในอดีต การแคชที่มีการจัดการเป็นอย่างดีบางส่วนหรือทั้งหมดจะช่วยลดการโต้ตอบระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์บางส่วน คุณสมบัติเหล่านี้ทั้งหมดจะสนับสนุนความสามารถในการปรับขนาดโดยไม่ทำให้เกิดปัญหาคอขวดในการสื่อสารซึ่งลดประสิทธิภาพการทำงาน

ความยืดหยุ่น

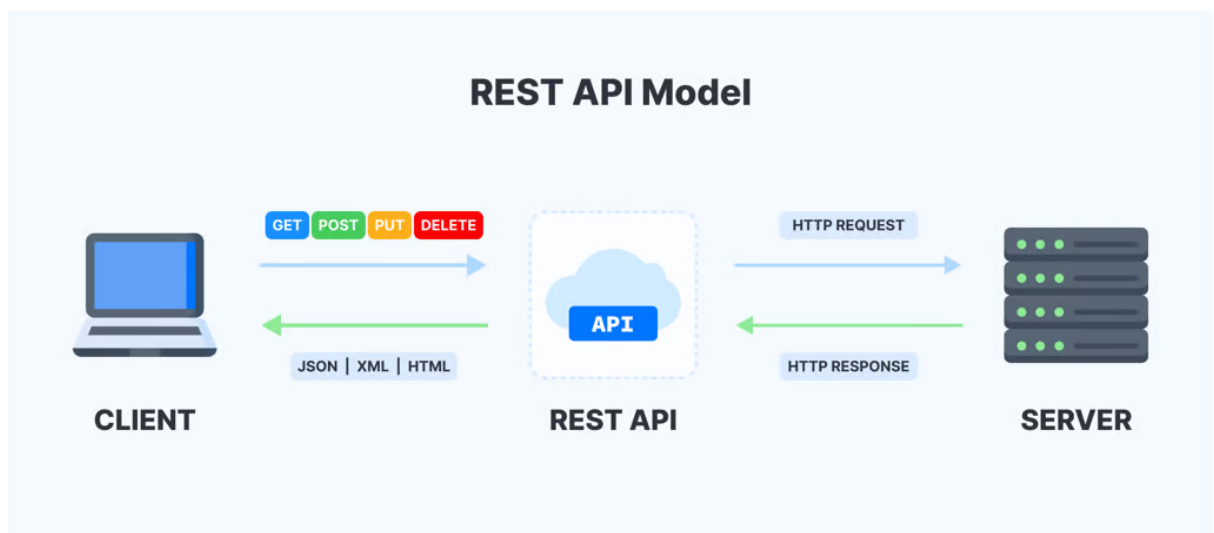
บริการเว็บ RESTful รองรับการแยกไคลเอนต์และเซิร์ฟเวอร์โดยสมบูรณ์ ซึ่งลดความซับซ้อนและแยกส่วนประกอบเซิร์ฟเวอร์ต่างๆ เพื่อให้แต่ละส่วนสามารถพัฒนาได้อย่างอิสระ ทั้งนี้การเปลี่ยนแปลงแพลตฟอร์มหรือเทคโนโลยีที่แอปพลิเคชันเซิร์ฟเวอร์ไม่ส่งผลกระทบต่อแอปพลิเคชันไคลเอนต์ ความสามารถในการแบ่งชิ้นการทำงานของแอปพลิเคชันช่วยเพิ่มความยืดหยุ่นยิ่งขึ้นไปอีก ตัวอย่างเช่น นักพัฒนาสามารถเปลี่ยนแปลงพื้นฐานข้อมูลได้โดยไม่ต้องเขียนตรรกะของแอปพลิเคชันขึ้นใหม่

ความไม่ขึ้นกับระบบใด

REST API ไม่ขึ้นอยู่กับเทคโนโลยีที่ใช้ คุณจึงสามารถเขียนแอปพลิเคชันไคลเอนต์และเซิร์ฟเวอร์ในภาษาการเขียนโปรแกรมต่างๆ ได้โดยไม่กระทบต่อการออกแบบ API นอกจากนี้ คุณยังสามารถเปลี่ยนเทคโนโลยีพื้นฐานในทั้งสองฝั่งได้โดยไม่มีผลกระทบต่อการสื่อสารอีกด้วย

RESTful API ทำงานอย่างไร

ฟังก์ชันพื้นฐานของ RESTful API จะเหมือนกับการท่องอินเทอร์เน็ต ไคลเอนต์จะติดต่อกับเซิร์ฟเวอร์โดยใช้ API เมื่อต้องใช้ทรัพยากร นักพัฒนา API อธิบายวิธีการที่ไคลเอนต์ควรใช้ REST API ในเอกสารประกอบ API ของแอปพลิเคชันเซิร์ฟเวอร์ โดยการเรียกใช้ REST API มีขั้นตอนทั่วไปดังนี้:



1. ไคลเอนต์ส่งคำขอไปยังเซิร์ฟเวอร์ ไคลเอนต์ปฏิบัติตามเอกสารประกอบ API เพื่อจัดรูปแบบคำขอในลักษณะที่เซิร์ฟเวอร์เข้าใจได้
2. เซิร์ฟเวอร์รับรองความถูกต้องของไคลเอนต์ และยืนยันว่าไคลเอนต์มีสิทธิ์ส่งคำขอดังกล่าว

3.เซิร์ฟเวอร์รับคำขอและประมวลผลเป็นการภายใน

4.เซิร์ฟเวอร์ส่งคืนการตอบสนองกลับไปยังไคลเอนต์ การตอบสนองมีข้อมูลที่บอกให้ลูกค้าทราบว่าคำขอดังกล่าวสำเร็จหรือไม่ การตอบสนองยังรวมถึงข้อมูลใดๆ ที่ไคลเอนต์ร้องขออีกด้วย

รายละเอียดคำขอและการตอบสนอง REST API จะแตกต่างกันเล็กน้อยโดยขึ้นอยู่กับวิธีการที่นักพัฒนา API ออกแบบ API

คำขอของไคลเอนต์ RESTful API มีอะไรบ้าง

RESTful API กำหนดให้คำขอมีส่วนประกอบหลักดังต่อไปนี้:

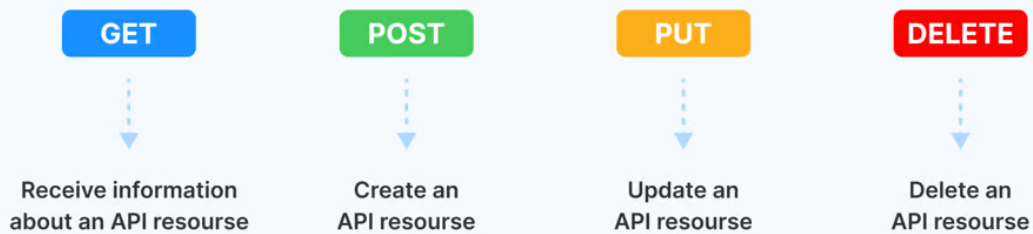
ตัวระบุทรัพยากรที่ไม่ซ้ำกัน

เซิร์ฟเวอร์ระบุทรัพยากรแต่ละรายการด้วยตัวระบุทรัพยากรที่ไม่ซ้ำกัน สำหรับบริการ REST โดยปกติแล้วเซิร์ฟเวอร์จะดำเนินการระบุทรัพยากรโดยใช้ตัวชี้แหล่งในอินเทอร์เน็ต (Uniform Resource Locator หรือ URL) โดย URL ระบุเส้นทางไปยังทรัพยากร ทั้งนี้ URL คล้ายคลึงกับที่อยู่เว็บไซต์ที่คุณป้อนลงในเบราว์เซอร์เพื่อเข้าชมหน้าเว็บต่างๆ นอกจากนี้ URL ยังเรียกอีกอย่างว่าตำแหน่งข้อมูลคำขอ และระบุอย่างชัดเจนต่อเซิร์ฟเวอร์ถึงสิ่งที่ลูกค้าต้องการ

วิธีการ

นักพัฒนามักใช้ RESTful API โดยใช้เกณฑ์วิธีขนส่งข้อความหลายมิติ (Hypertext Transfer Protocol หรือ HTTP) วิธีการ HTTP จะบอกให้เซิร์ฟเวอร์ทราบถึงสิ่งที่ต้องทำกับทรัพยากร โดยวิธีการ HTTP ทั่วไปมี 4 วิธีดังต่อไปนี้:

HTTP Methods



GET

ไคลเอ็นต์ใช้ GET เพื่อเข้าถึงทรัพยากรที่อยู่ URL ที่ระบุบนเซิร์ฟเวอร์ ซึ่งสามารถแคชคำขอ GET และส่งพารามิเตอร์ในคำขอ RESTful API เพื่อสั่งให้เซิร์ฟเวอร์กรองข้อมูลก่อนส่ง

POST

ไคลเอ็นต์ใช้ POST เพื่อส่งข้อมูลไปยังเซิร์ฟเวอร์ ซึ่งรวมถึงการแทนข้อมูลพร้อมกับคำขอ การส่งคำขอ POST เดียวกันหลายครั้งมีผลข้างเคียงเหมือนกับการสร้างทรัพยากรเดียวกันหลายครั้ง

PUT

ไคลเอ็นต์ใช้ PUT เพื่ออัปเดตทรัพยากรที่มีอยู่บนเซิร์ฟเวอร์ การส่งคำขอ PUT เดียวกันหลายครั้งในบริการเว็บ RESTful จะให้ผลลัพธ์เหมือนกัน ซึ่งแตกต่างจาก POST

DELETE

ไคลเอ็นต์ใช้คำขอ DELETE เพื่อลบทรัพยากรออก โดยคำขอ DELETE สามารถเปลี่ยนสถานะเซิร์ฟเวอร์ได้ อย่างไรก็ตาม หากผู้ใช้ไม่มีการรับรองความถูกต้องที่เหมาะสม คำขอก็จะล้มเหลว

ส่วนหัว HTTP

ส่วนหัวของคำขอคือข้อมูลเมตาที่แลกเปลี่ยนระหว่างไคลเอนต์และเซิร์ฟเวอร์ ตัวอย่างเช่น ส่วนหัวของคำขอจะระบุรูปแบบของคำขอและการตอบกลับ ให้ข้อมูลเกี่ยวกับสถานะคำขอ และอื่นๆ

ข้อมูล

คำขอ REST API อาจรวมถึงข้อมูลสำหรับวิธีการ POST, PUT และ HTTP อื่นๆ เพื่อให้ทำงานได้สำเร็จ

พารามิเตอร์

คำขอ RESTful API อาจรวมถึงพารามิเตอร์ที่ให้รายละเอียดเพิ่มเติมกับเซิร์ฟเวอร์เกี่ยวกับสิ่งที่ต้องดำเนินการ โดยพารามิเตอร์ประเภทต่างๆ มีดังต่อไปนี้:

- พารามิเตอร์พาธที่ระบุรายละเอียด URL
- พารามิเตอร์การสอบถามที่ขอข้อมูลเพิ่มเติมเกี่ยวกับทรัพยากร
- พารามิเตอร์คูกี้ที่รับรองความถูกต้องของไคลเอนต์ได้อย่างรวดเร็ว

วิธีการรับรองความถูกต้อง RESTful API มีอะไรบ้าง

บริการเว็บ RESTful ต้องรับรองความถูกต้องของคำขอก่อนที่จะส่งการตอบสนอง การรับรองความถูกต้องเป็นกระบวนการยืนยันตัวตน ตัวอย่างเช่น คุณสามารถพิสูจน์ตัวตนของคุณได้โดยแสดงบัตรประจำตัวประชาชนหรือใบขับขี่ ในทำนองเดียวกัน ไคลเอนต์บริการ RESTful เองก็จะต้องพิสูจน์ตัวตนต่อเซิร์ฟเวอร์เพื่อสร้างความไว้วางใจ

โดย RESTful API มีวิธีการรับรองความถูกต้องทั่วไป 4 วิธี ได้แก่

การรับรองความถูกต้อง HTTP

HTTP กำหนดรูปแบบการรับรองความถูกต้องบางส่วนที่คุณสามารถใช้ได้โดยตรงเมื่อคุณนำ REST API ไปใช้ โดยทั้งสองรูปแบบมีดังต่อไปนี้:

การรับรองความถูกต้องพื้นฐาน

ในการรับรองความถูกต้องพื้นฐาน ไคลเอ็นต์จะส่งชื่อผู้ใช้และรหัสผ่านในส่วนหัวของคำขอ ซึ่งจะเข้ารหัสด้วย base64 ซึ่งเป็นเทคนิคการเข้ารหัสที่แปลงทั้งคู่เป็นชุดอักขระ 64 ตัวเพื่อการรับส่งข้อมูลที่ปลอดภัย

การรับรองความถูกต้องของแบเรอร์

คำว่ารับรองความถูกต้องของแบเรอร์หมายถึงกระบวนการให้การควบคุมการเข้าถึงแก่แบเรอร์ของโทเค็น โดยโทเค็นแบบแบเรอร์มักจะเป็นสตริงอักขระที่เข้ารหัสซึ่งเซิร์ฟเวอร์สร้างขึ้นเพื่อตอบสนองต่อคำขอเข้าสู่ระบบ จากนั้นไคลเอ็นต์จะส่งโทเค็นในส่วนหัวของคำขอเพื่อเข้าถึงทรัพยากร

คีย์ API

คีย์ API เป็นอีกตัวเลือกหนึ่งสำหรับการรับรองความถูกต้องของ REST API ด้วยวิธีการนี้ เซิร์ฟเวอร์จะกำหนดค่าที่สร้างขึ้นเฉพาะให้กับไคลเอ็นต์ที่ใช้งานครั้งแรก เมื่อใดก็ตามที่ไคลเอ็นต์พยายามเข้าถึงทรัพยากร ก็จะใช้คีย์ API เฉพาะเพื่อยืนยันตัวเอง ทั้งนี้คีย์ API มีความปลอดภัยน้อยกว่า เนื่องจากไคลเอ็นต์ต้องส่งคีย์ ซึ่งทำให้เสี่ยงต่อการถูกขโมยผ่านเครือข่าย

OAuth

OAuth รวมรหัสผ่านและโทเค็นเข้าด้วยกันเพื่อการเข้าถึงการเข้าสู่ระบบที่มีความปลอดภัยสูงในทุกระบบ โดยเซิร์ฟเวอร์จะขอรหัสผ่านก่อนแล้วจึงขอโทเค็นเพิ่มเติมเพื่อให้กระบวนการรับรองความถูกต้องเสร็จสมบูรณ์ ซึ่งสามารถตรวจสอบโทเค็นได้ตลอดเวลาและเมื่อเวลาผ่านไป ด้วยขอบเขตและระยะเวลาที่เฉพาะเจาะจง

การตอบสนองของเซิร์ฟเวอร์ RESTful API มีอะไรบ้าง

หลักการ REST กำหนดให้การตอบสนองของเซิร์ฟเวอร์มีองค์ประกอบหลักดังต่อไปนี้:

บรรทัดสถานะ

บรรทัดสถานะประกอบด้วยรหัสสถานะสามหลักที่แจ้งว่าคำขอสำเร็จหรือล้มเหลว ตัวอย่างเช่น รหัส 2XX ระบุถึงความสำเร็จ แต่รหัส 4XX และ 5XX ระบุถึงข้อผิดพลาด รหัส 3XX ระบุถึงการเปลี่ยนเส้นทาง URL

โดยรหัสสถานะทั่วไปบางส่วนมีดังต่อไปนี้:

- 200: การตอบสนองเพื่อระบุถึงความสำเร็จทั่วไป
- 201: การตอบสนองเพื่อระบุถึงความสำเร็จของวิธีการ POST
- 400: คำขอที่ไม่ถูกต้องที่เซิร์ฟเวอร์ไม่สามารถประมวลผลได้
- 404: ไม่พบทรัพยากร

เนื้อหา

เนื้อหาการตอบสนองประกอบด้วยการแทนข้อมูลทรัพยากร เซิร์ฟเวอร์จะเลือกรูปแบบการแทนข้อมูลที่เหมาะสมตามสิ่งที่อยู่ในส่วนหัวของคำขอ ไคลเอนต์สามารถขอข้อมูลในรูปแบบ XML หรือ JSON ได้ ซึ่งกำหนดวิธีการเขียนข้อมูลในรูปแบบข้อความธรรมดา ตัวอย่างเช่น หากไคลเอนต์ร้องขอชื่อและอายุของบุคคลชื่อว่า John เซิร์ฟเวอร์จะส่งคืนการแทนข้อมูล JSON ดังนี้:

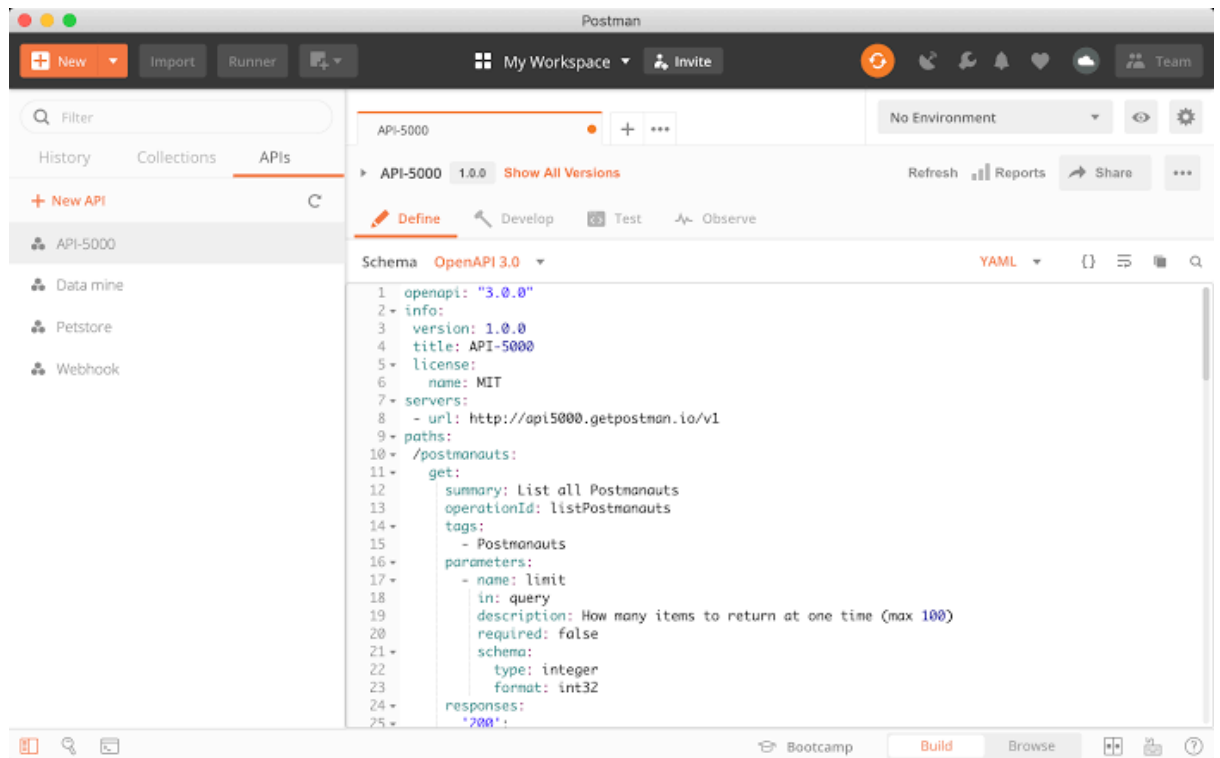
```
'{"name":"John", "age":30}'
```

ส่วนหัว

การตอบสนองยังมีส่วนหัวหรือข้อมูลเมตาเกี่ยวกับการตอบสนองอีกด้วย ซึ่งให้บริบทเพิ่มเติมเกี่ยวกับการตอบสนองและมีข้อมูลต่างๆ เช่น เซิร์ฟเวอร์ การเข้ารหัส วันที่ และประเภทเนื้อหา

ตัวอย่างเครื่องมือที่ใช้ทดสอบ RESTful API

- Postman 



การทำ RESTful API ก็ทำได้ในหลายภาษามาก ไม่ว่าจะเป็น php java node ซึ่งแน่นอนว่าในแต่ละภาษาก็มีจุดเด่นที่แตกต่างกันไปในแต่ละภาษา

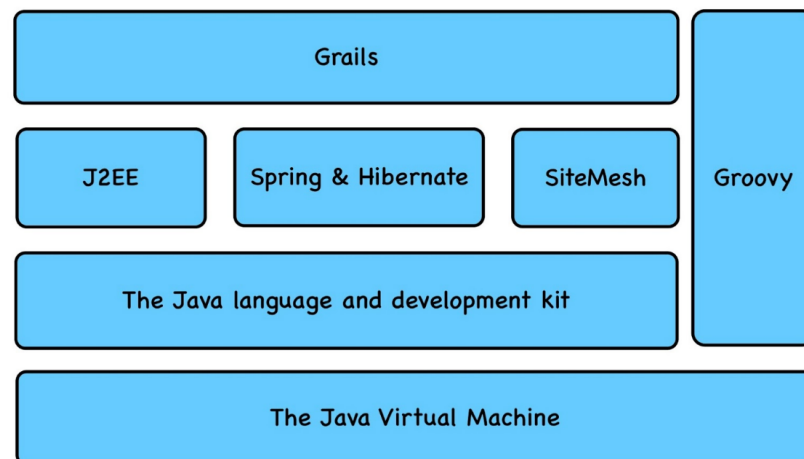
18. Grails



อ้างอิง <https://grails.org/>

Grails ถูกสร้างขึ้นมาให้เราสามารถสร้าง Web Application ในระบบองค์กรขนาดเล็ก ไปจนถึงขนาดใหญ่ ซึ่งทำงานบนแพลตฟอร์ม JAVA ทำให้สามารถเขียนโปรแกรมได้อย่างสนุกสนานยิ่งขึ้น ผู้พัฒนาสามารถจดจ่ออยู่กับ Business Domain ได้ดี โดยไม่จำเป็นต้องกังวล หรือเหนื่อย ในการตั้งค่า ติดตั้ง Library ที่ยุ่งยาก และภาษาหลักในการพัฒนาบน Framework นี้ คือภาษา Groovy นอกจากนี้ยังมี Domain-Specific Language (DSL) ที่ช่วยให้เพิ่มความสามารถในการ

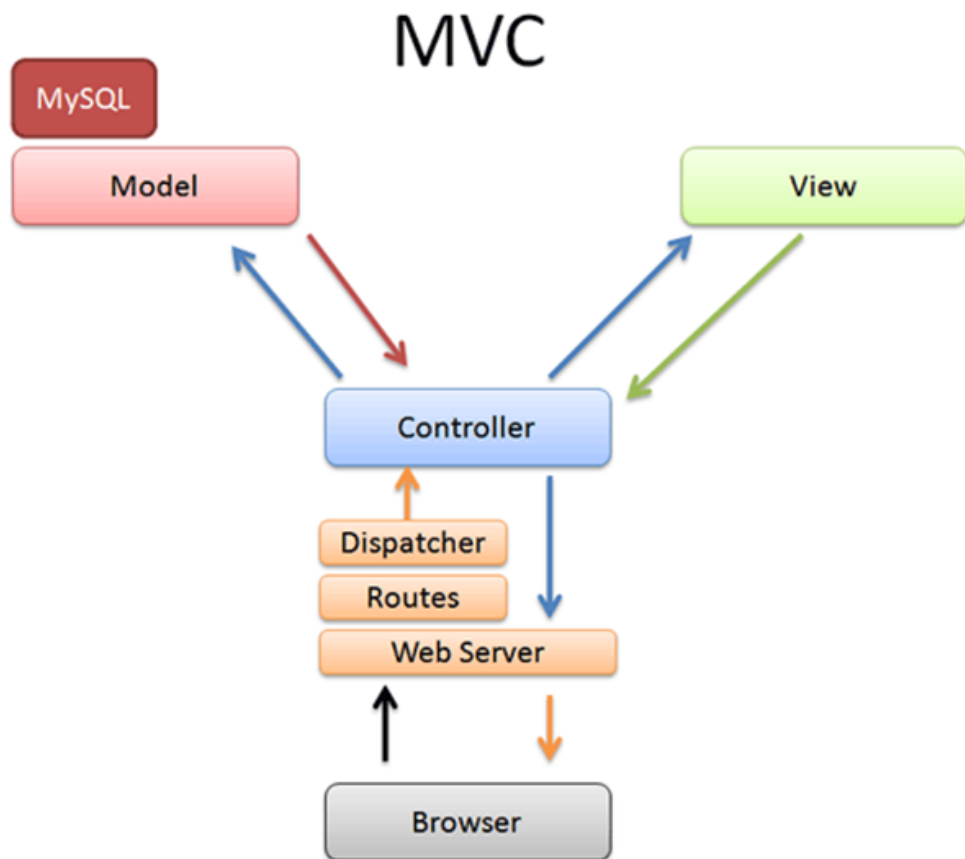
พัฒนาให้ง่ายยิ่งขึ้น และหากมองลึกลงไปในระดับโครงสร้างของ Grails แล้ว เราจะเห็นว่า Grails ถูกสร้างขึ้นมาจากเทคโนโลยีที่ได้รับการยอมรับอย่างกว้างขวางว่าเป็นสิ่งที่ดีที่สุดคือ Spring, Hibernate และ JAVA แพลตฟอร์ม นอกจากนั้น Grails ยังนำเอาข้อดีต่างๆ ของ Dynamic Language Framework อย่าง Ruby on rails, Django เข้ามารวมกันได้ เพื่อให้เกิดประโยชน์กับผู้ใช้งานมากที่สุด



Grails ถูกสร้างขึ้นมาจากเฟรมเวิร์คที่มีอยู่แล้วโดยที่ Grails จะช่วยให้เราสร้างแอปพลิเคชันขึ้นมาได้อย่างรวดเร็ว และยังไม่สูญเสียความยืดหยุ่นของระบบไป โดยสิ่งที่นำมาประกอบกันเพื่อสร้าง Grails มีดังต่อไปนี้

- hibernate: ORM persistence
- Spring: inversion of control container และ wrapper framework ของ JAVA ที่
- Site mesh: layout-rendering framework
- Jetty: embeddable servlet container
- hsqldb: relational database management system ระบบฐานข้อมูลคุณภาพสูงที่ถูกเขียนด้วย JAVA
- ORM หรือ object relational mapping
- IOC หรือ Inversion of control

อ้างอิง <https://grails.org/>



MVC คือสถาปัตยกรรมซอฟต์แวร์ (Software Architecture) ที่ Grails ยึดหลักโดยมีการแบ่งแยกระบบออกเป็น 3 ส่วนหลักๆ ดังแสดงในรูปที่ 2 ได้แก่ Data Model, User Interface, and Control Logic โดยมีรายละเอียดการทำงานดังนี้

- Model จะประกอบด้วยคลาสที่เชื่อมต่อกับ DBMS ประกอบด้วยฟังก์ชันที่เกี่ยวข้องกับการเพิ่มและการเปลี่ยนแปลงข้อมูลในฐานข้อมูลเพื่อช่วยจัดการงานด้านฐานข้อมูล
- Controller เป็นส่วนที่ทำงานเป็นอันดับแรกเมื่อมีโปรแกรมถูกเรียกจากเว็บเบราว์เซอร์เป็นส่วนที่ติดต่อการทำงานระหว่างผู้ใช้และโปรแกรมมีการติดต่อกับฐานข้อมูลด้วย Model และแสดงผลข้อมูลผ่านทาง View และเป็นส่วนที่มีการประมวลผลหลักของโปรแกรม
- Views เป็นส่วนของข้อมูลที่ใช้แสดงผลซึ่งในส่วนของ view จะสามารถแบ่งเว็บเพจเป็นส่วนย่อยได้

ทั้งนี้ในการพัฒนาโปรแกรมประยุกต์ผ่านเว็บ ที่มีลักษณะการพัฒนาเป็นไปตามหลักการ MVC และง่ายต่อการนำไปรันบน Cloud ที่สำคัญตัวหนึ่งคือ Groovy on Grails ประกอบกับความสามารถสำคัญข้างล่าง เว็บเฟรมเวิร์กนี้จึงเหมาะแก่การใช้เป็นเครื่องมือในการพัฒนาโปรแกรมประยุกต์ที่พร้อมทำงานบน Cloud

- ใช้ภาษาแบบไดนามิก ที่ทำงานอยู่บน Java Virtual Machine (JVM) ซึ่งทำให้เข้าถึงไลบรารีของจาวาที่มีได้ทั้งหมด
- สนับสนุน DSL (Dynamic Specific Languages) ทำให้โค้ดที่เขียนขึ้นอ่านง่าย และง่ายต่อการบำรุงรักษา
- ทำให้การทดสอบง่ายขึ้น ด้วยเครื่องมือ Unit Testing และ Mocking มาให้ในตัวเฟรมเวิร์ค
- คอมไพล์ออกมาเป็นจาวาไบต์โค้ดโดยตรง ทำให้ทำงานได้บน JVM ปกติ (ง่ายต่อการทำงานบนกลุ่มเมฆที่สนับสนุนจาวา)

