การใช้งาน MongoDB เบื้องต้น

โดย แอดมินโฮ โอน้อยออก

ประวัติการแก้ไข

ครั้งที่	วันที่	รายละเอียดการแก้ไข
1	9 ม.ค. 2559	เริ่มสร้าง และเผยแพร่ผลงาน
2	4 เม.ย. 2559	แก้ไขเล็กน้อย

ถ้าท่านดาวน์โหลดทิ้งไว้นาน แล้วเพิ่งมาเปิดอ่าน ก็ขอรบกวนให้โหลดใหม่อีกครั้งที่

http://www.patanasongsivilai.com/itebook_form.html

เผื่อผมอัพเดตแก้ไข pdf ตัวใหม่เข้าไป หรือใครไปดาวน์โหลดมาจากที่อื่นก็อาจพลาดเวอร์ชั่นใหม่ล่าสุดได้ครับ

และรบกวนช่วยกรอกแบบสอบถาม ตามลิงค์ข้างบนด้วยนะครับ

แอดมินโฮ โอน้อยยอก

(จตุรพัชร์ พัฒนทรงศิวิไล)

9 ธันวาคม 2558

ถ้าสนใจเกี่ยวกับเพจด้านไอที ก็ติดตามได้ที่ https://www.facebook.com/programmerthai/

เกริ่นนำ

เริ่มแรกเดิมทีผมได้เขียนหนังสือ “เสียดายไม่ได้อ่านจาวาสคริปต์ ฝั่งเซิร์ฟเวอร์ (Node.js ฉบับย่อ)” เล่มที่ 2 ก็กะจะเขียนเรื่องการใช้ Node.js ติดต่อฐานข้อมูล MongoDB

แต่เมื่อเขียนไปเขียนมา ก็กลัวคนอ่านไม่เข้าใจว่า ...MongoDB คืออะไร ผมเลยแยกเขียนออกมา จนกลายมาเป็นเล่มนี้นั้นเอง ซึ่งจุดประสงค์หลัก ก็เพื่อแนะนำให้เห็นภาพว่า ถ้าใช้ Node.js ติดต่อกับ MongoDB ต้องทำอย่างไรมากกว่า (เล่มนี้จึงเหมือนเป็นซีรีย์ภาคต่อมากกว่า)

สุดท้ายนี้หากเนื้อหามีอะไรผิดพลาดไป เช่น ให้ข้อมูลผิด สะกดอะไรผิดไป มุมแป็กบ้าง ขำบ้าง หรืออ่านแล้วมึนงงไป 7 วัน เป็นต้น ผมก็ขออภัยมา ณ โอกาสนี้ด้วย และถ้าคุณเข้าใจ ไม่เข้าใจยังไง ก็สามารถชี้แนะผมได้ตลอดเวลา

...อีกทั้งผมก็ตั้งใจจะหมั่นอัพเดตเนื้อหา ขึ้นอยู่กับเวลา โอกาส และความสามารถจะอำนวย

อธิบาย MongoDB

ถ้าจะอธิบายความหมายของ MongoDB สั้น ๆ มันคือฐานข้อมูลประเภท NoSQL ซึ่งชื่อมันก็ชัดเจนดีนะครับ ...มันจะปราศจากการใช้ภาษา SQL คุยกับฐานข้อมูล เพราะไม่ได้เก็บข้อมูลเป็นตาราง (Table)

id	col	col2	col3

ซึ่งคอนเซปท์ของมัน จะไม่เหมือนฐานข้อมูลประเภท Relational Database เช่น MySQL, SQL Server, Oracle และ DB2 เป็นต้น ที่ใช้ภาษา SQL ในการสร้างตาราง หรือค้นหาข้อมูล (Query) หรืออัพเดตตาราง เป็นต้น

ถ้าถาม ถ้าไม่ใช่ตาราง แล้ว MongoDB มันเก็บข้อมูลเป็นอะไร?

คำตอบ เก็บข้อมูลในรูปแบบ JSON

ตัวอย่าง JSON

```
{
"name": "webapp"
, "version": "1.0.0"
}
```

***หมายเหตุ** คำว่า NoSQL มันไม่ใช่มาตรฐานอะไรเลย เพียงแค่บอกว่าไม่ใช้ภาษา SQL คุยกับฐานข้อมูล ก็เท่านั้นเอง ด้วยเหตุนี้ฐานข้อมูล NoSQL ประเภทอื่น ๆ จึงอาจเก็บข้อมูลแบบอื่นได้เช่นกัน (ไม่ใช่ JSON)

สำหรับฐานข้อมูลแบบ NoSQL อาจแบ่งได้ดังนี้ [1]

ประเภท	ตัวอย่างชื่อฐานข้อมูล
Document databases	MongoDB, CouchDB, Elasticsearch
Graph stores	Neo4J, Infinite Graph, InfoGrid
Key-value stores	DynamoDB, Redis, MemcacheDB
Wide-column stores	Cassandra, Amazon SimpleDB, Hadoop HBase

*** สำหรับวิธีการจัดเก็บของแต่ละฐานข้อมูล ก็ยังแตกต่างกันอีกด้วยครับ ขึ้นอยู่กับผู้ผลิตแต่ละเจ้า

...และในหนังสือเล่มนี้ ก็ได้เลือก MongoDB มาเป็นตัวอย่างในการศึกษาเท่านั้น

สำหรับจุดเด่นเฉพาะของ MongoDB (ไม่ได้พูดถึงตัวอื่น) ก็จะแสดงให้ดูเอาที่สำคัญแล้วกันดังนี้ [2]

- รองรับจำนวนข้อมูลที่เพิ่มขึ้นอย่างมหาศาล และรองรับการทำงานหนักได้ ๆ
- ค้นหาจากข้อมูลที่มีปริมาณมากมายมหาศาล ได้อย่างรวดเร็ว (รองรับการทำ Full Index)
- เก็บข้อมูลได้ทั้งกว้าง และลึก (ไม่ใช่แบบตารางที่มีมิติเดียว ซึ่งเก็บข้อมูลได้แค่ 1 แถว)
- เรียกข้อมูลมาแสดงผลได้ง่าย (ไม่ต้องใช้ภาษา SQL ทำการ Join ตารางโน่นนี่นั่น)
- แก้ไขข้อมูลได้รวดเร็ว
- สำรองข้อมูลได้ง่าย ไม่ต้องตั้งค่าอะไรเยอะ
- เขียนชุดคำสั่งเป็นสคริปต์ แล้วสั่งรันทีเดียวได้
- เก็บข้อมูลด้วยระบบ GridFS (เก็บข้อมูลเป็นก้อน ๆ บนฮาร์ดดิสก์ ซึ่งสามารถรองรับการเพิ่มขึ้น หรือลดลงของปริมาณข้อมูล)
- มีบริการสอบถามและดูแลเป็นพิเศษ (เสียเงิน)

ต้องบอกอย่างนี้นะครับ จุดประสงค์ของ NoSQL (ไม่ใช่แค่ MongoDB) มันเกิดขึ้นมาเพื่อลดความยุ่งยาก และซับซ้อน เวลาต้องเข้าไปจัดการกับข้อมูลที่มีปริมาณมาก โดยไม่เน้นความถูกต้องของการทำงาน แต่เน้นให้ทำงานเร็ว ซึ่งโดยทั่วไปแล้ว NoSQL จะเร็วกว่าฐานข้อมูลแบบ SQL

ด้วยเหตุนี้ NoSQL ทั้งหลายแหล่ จึงเหมาะกับ **BigData** (ข้อมูลปริมาณมหาศาลมาก ๆ)

สรรพสิ่งทุกอย่างในใต้หล้า ย่อมมีเหรียญสองด้าน มีทั้งข้อดีและข้อเสีย ไม่เว้นแม้แต่ NoSQL ก็ย่อมมีข้อเสียเช่นเดียวกัน ดังตัวอย่าง

- ถ้าจะเน้นความถูกต้อง จะผิดพลาดไม่ได้เลย ...NoSQL จะไม่เหมาะ เพราะมีโอกาสที่ข้อมูลจะเกิดการ loss (สูญหาย) ได้มาก (เน้นเร็ว ไม่เน้นความถูกต้อง)
- เนื่องจาก NoSQL ไม่มีมาตรฐานกลาง ดังนั้นเมื่อเปลี่ยนไปใช้ฐานข้อมูลค่ายใหม่ ก็ต้องเสียเวลาศึกษาใหม่
- ผู้เชี่ยวชาญด้าน NoSQL สำหรับองค์กรในไทย ยังมีไม่มาก ซึ่งสวนทางกับเทรนด์ BigData ที่กำลังมาแรงในปัจจุบัน (นับตั้งแต่ผู้เขียนแต่งหนังสือ)

ตามลัทธิเต๋า จะ**มีหยิน-หยาง**

หมายถึงสรรพสิ่งต้องคู่กัน

มีพระอาทิตย์ ต้องมีพระจันทร์

มีชาย ต้องมีหญิง

มีกลางวัน ต้องมีกลางคืน

ดังนั้น ...มี SQL ก็ต้องมี NoSQL

เสริมนิดหน่อย

ผมขอยกนิยาม MongoDB อย่างเป็นทางการตามลิงค์ https://docs.mongodb.org/getting-started/node/introduction/ มาให้คุณดูดังนี้

> Introduction to MongoDB
>
> MongoDB is an open-source *document database* that provides high performance, high availability, and automatic scaling. MongoDB obviates the need for an Object Relational Mapping (ORM) to facilitate development.

ผมจะลองแปลนิยาม อาจไม่สละสลวยเท่าไรนัก

> "MongoDB เป็นฐานข้อมูลโอเพ่นซอร์สที่เก็บเอกสาร (document database) ที่มาพร้อมกับประสิทธิภาพที่สูง รองรับการใช้งานตลอดเวลาได้สูง ปรับตัวขยายขนาดได้อย่างอัตโนมัติ ซึ่งตัว MongoDB ได้กำจัดแนวคิดเรื่อง Object Relational Mapping (ORM) จึงทำให้มันมีความสะดวกในการพัฒนาซอฟต์แวร์"

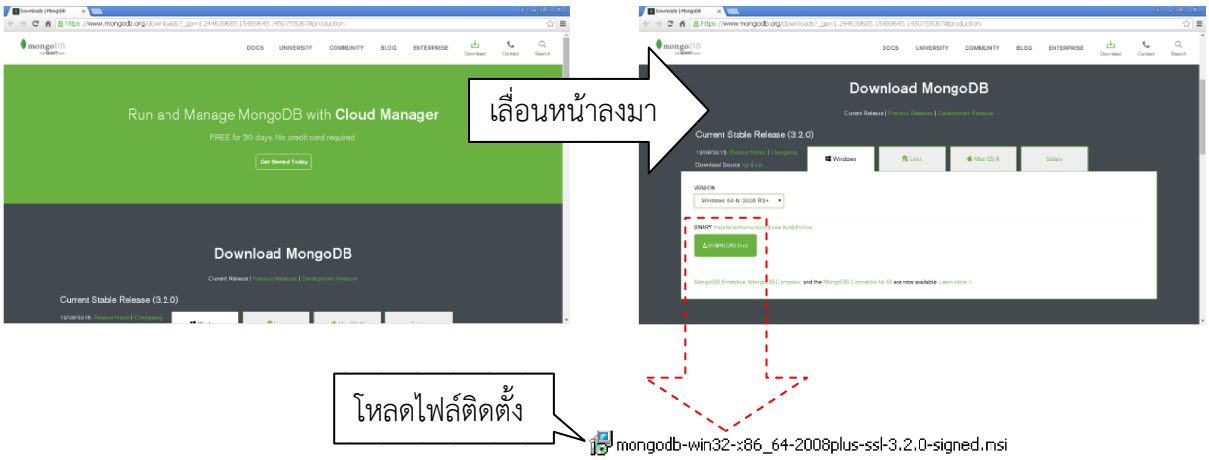
....ยังไม่ต้องซีเรียสกับคำนิยามมากครับ แค่ยกมาให้อ่านเฉย ๆ อีกทั้งผมก็เกลามาจาก Google translate ซึ่งมันแปลได้แค่นี้แหละ อิ ๆ ๆ

วิธีติดตั้ง MongoDB

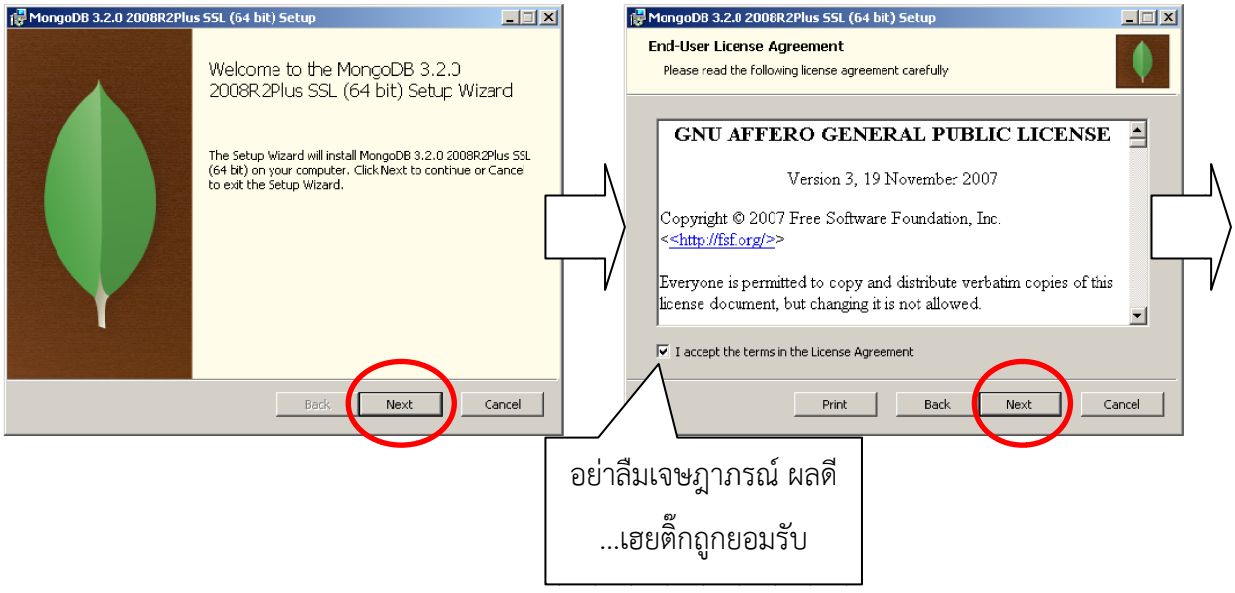
1) ให้คุณไปที่ลิงค์ข้างล่าง เพื่อดาวน์โหลดไฟล์ติดตั้ง

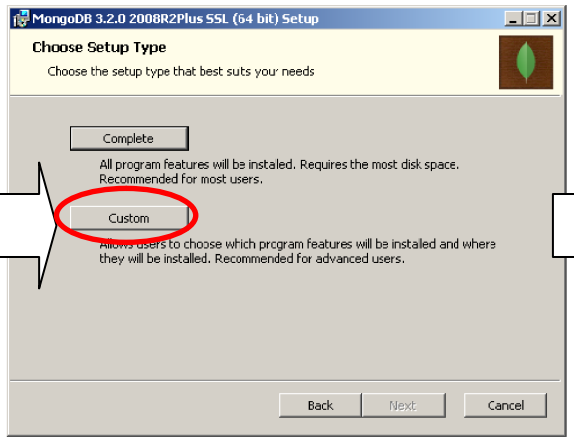
https://www.mongodb.org/downloads?_ga=1.244639685.15489645.1450755087#production

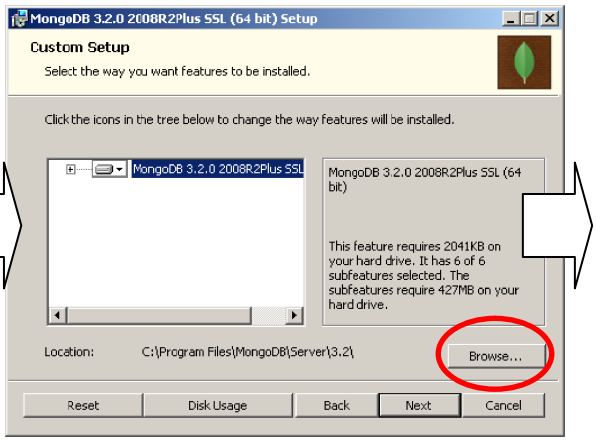
2) คุณจะเห็นหน้าเว็บ และให้เลื่อนเพจลงมาเรื่อย ๆ จนเจอปุ่มให้กดดาวน์โหลดไฟล์ติดตั้ง ซึ่งในที่นี้ผมจะเลือกลงบนวินโดวส์ครับ (เครื่องผู้เขียนเป็น64 บิต)

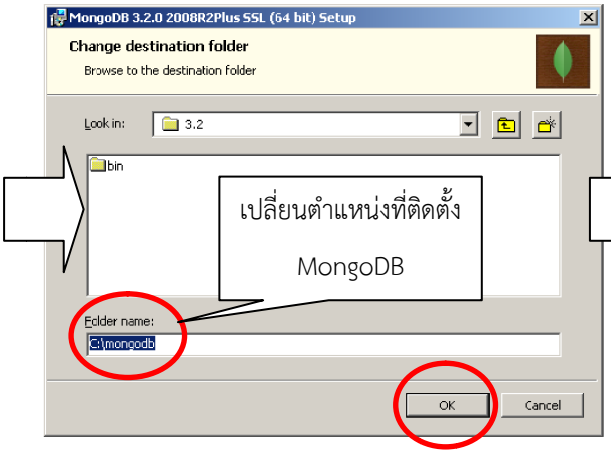


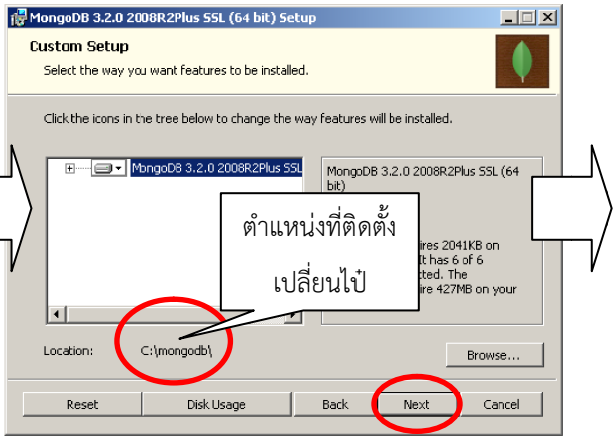
3) เมื่อดาวน์โหลดเสร็จแล้ว ก็ให้ดับเบิลคลิกที่ไฟล์ติดตั้ง เหมือนคุณลงโปรแกรมปกติ ซึ่งรูปข้างล่างจะเป็นแค่ไอเดียเฉย ๆ ...เพราะถ้ามีเวอร์ชั่นใหม่ออกมา ก็ให้ติดตั้งทำนองเดียวกัน (หวังว่านะครับ)

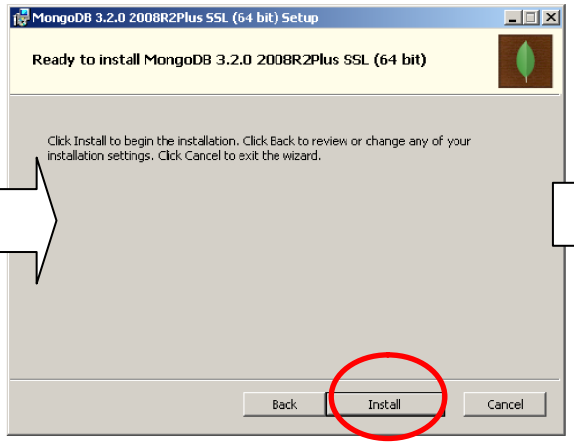


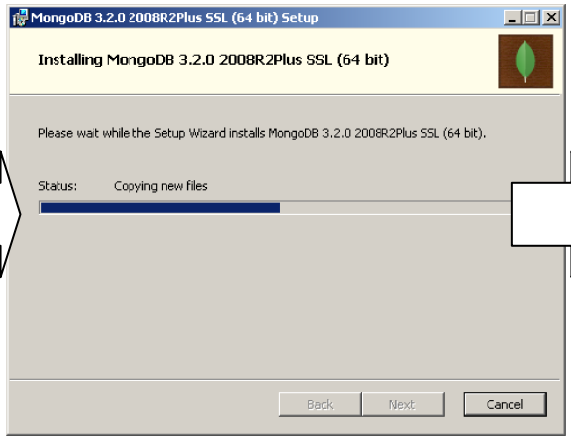


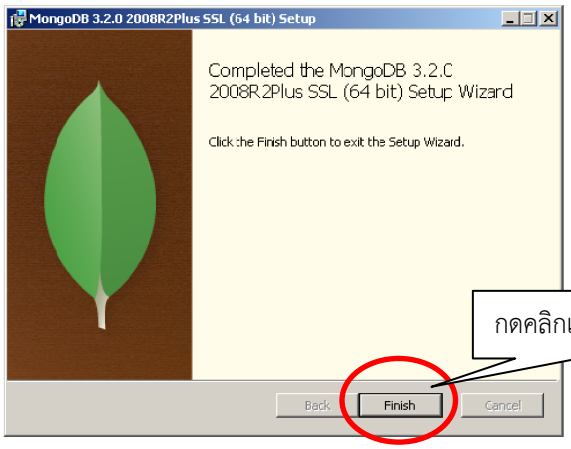












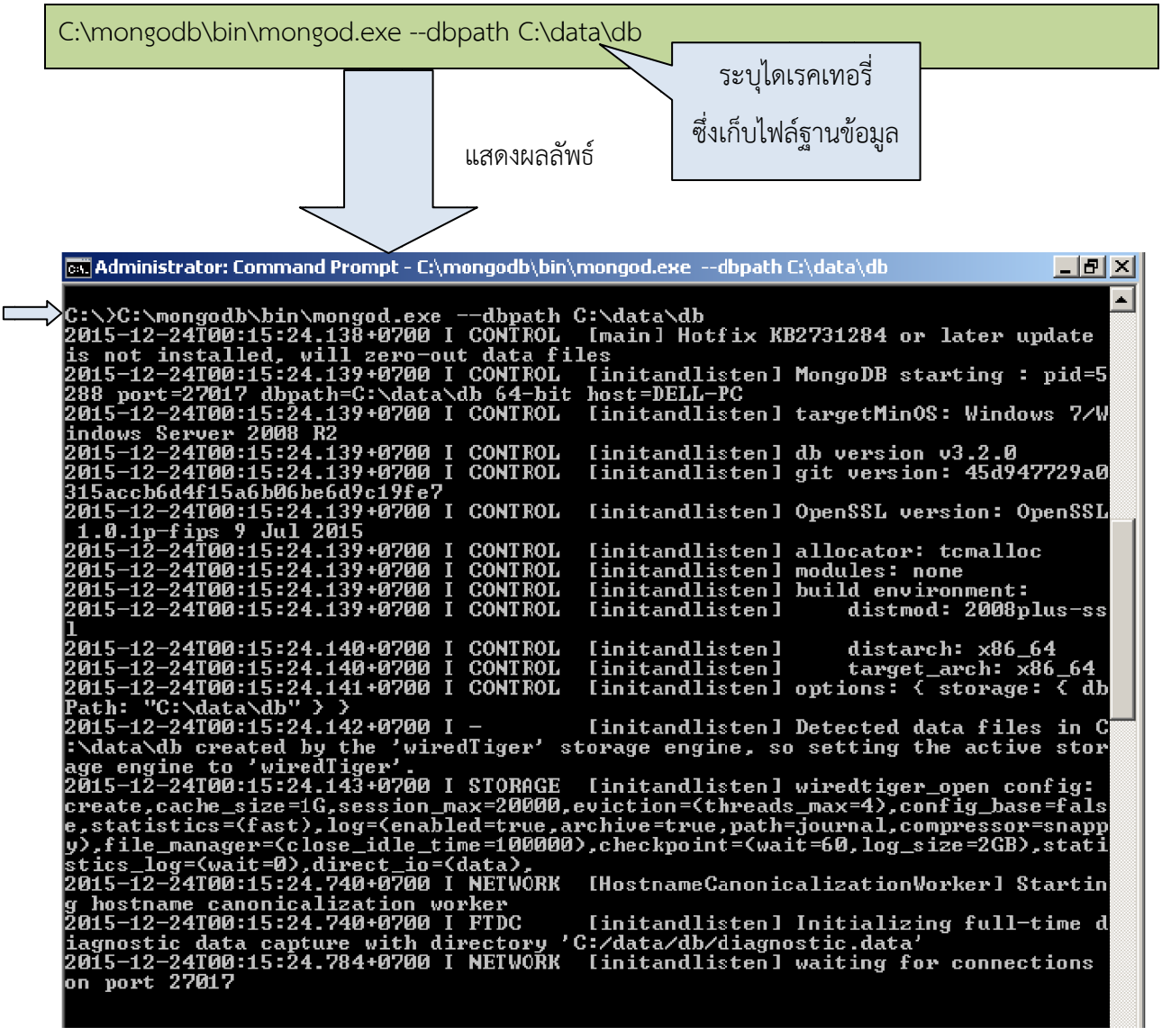
หมายเหตุ

ถ้าติดตั้งบน Windows Server 2008 R2 หรือ Windows 7 อาจต้องติดตั้ง hotfix ก่อน ตามลิงค์ข้างล่างก่อน (กรณีมีปัญหาใช้งาน MongoDB ไม่ได้)

http://support.microsoft.com/kb/2731284

4) เตรียมสร้างโฟลเดอร์ใหม่ สำหรับเก็บไฟล์ฐานข้อมูลต่าง ๆ และไฟล์ log ด้วยการพิมพ์คำสั่งข้างล่าง

5) ให้ลองรัน MongoDB ตามคำสั่งข้างล่าง



ในภาพนี้ MongoDB กำลังทำงานอยู่ (MongoDB ของผู้เขียนจะถูกติดตั้งอยู่ที่ C:\mongodb) ซึ่งหน้าจอนี้ห้ามปิดเด็ดขาด ปล่อยค้างไว้อย่างนี้แหละ

...และถ้าคุณแอบไปเปิดโฟลเดอร์ C:\data\db ก็จะเห็นว่ามีไฟล์ต่าง ๆ ของ MongoDB ได้ถูกสร้างเอาไว้ให้

วิธีตั้งค่า MongoDB ให้เป็นเซอร์วิส

ในบทก่อนหน้านี้ จะเห็นว่าเวลารัน MongoDB ผ่านทางคอมมานไลน์ มันจะยุ่งยากเกินไป แต่เราสามารถเปลี่ยนให้มารันอยู่เบื้องหลังได้ ด้วยการทำเป็นเซอร์วิส (Service) ในวินโดวส์ (เมื่อเปิดคอมขึ้นมา มันก็จะรันให้ทันที) ซึ่งจะมีวิธีการตั้งค่าตามขั้นตอนต่อไปนี้

1) สร้างไฟล์ “mongod.cfg” ที่โฟลเดอร์ดังต่อไปนี้

C:\mongodb\mongod.cfg

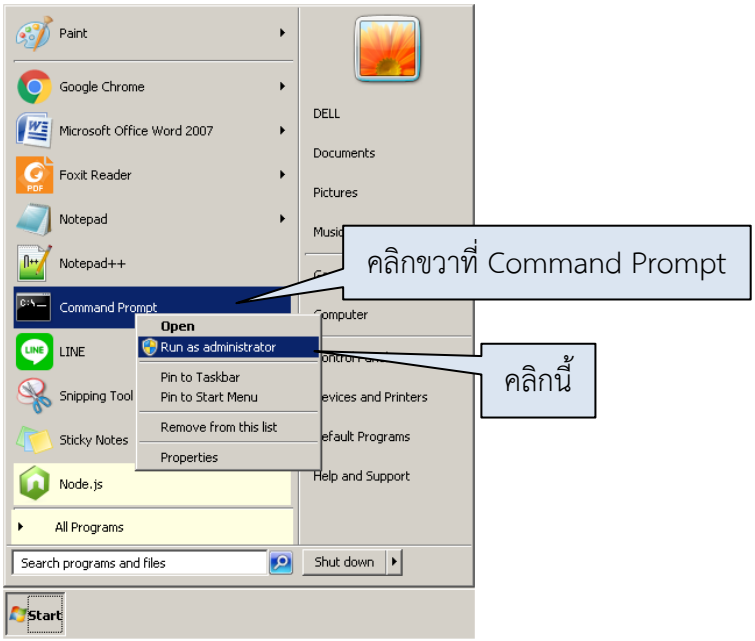
ซึ่งเนื้อหาภายในไฟล์จะเขียนดังนี้

```
systemLog:
    destination: file
    path: c:\data\log\mongod.log
storage:
    dbPath: c:\data\db
```

ระบุชื่อไฟล์ log

ระบุชื่อโฟลเดอร์สำหรับเก็บไฟล์ฐานข้อมูลต่าง ๆ

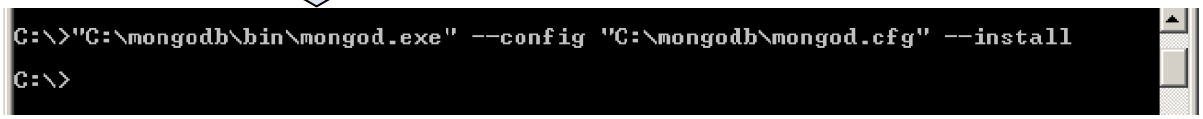
สำหรับขั้นตอนต่อจากนี้ไป ควรเปิดคอมมานไลน์บนวินโดวส์ โดยได้รับสิทธิเป็น “Administrative” ดังภาพ



2) ติดตั้ง MongoDB ให้กลายมาเป็นเซอร์วิส ด้วยคำสั่งบนคอมมานไลน์ ดังนี้

```
"C:\mongodb\bin\mongod.exe" --config "C:\mongodb\mongod.cfg" --install
```

ไฟล์ที่สร้างในข้อ 1



3) ถ้าจะสั่งให้เซอร์วิส MongoDB ทำงาน ก็ให้ใช้คำสั่งดังนี้

```
net start MongoDB
```

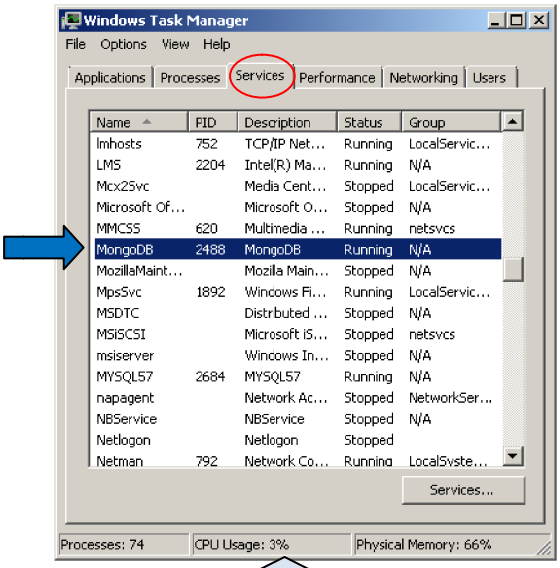
4) ถ้าจะสั่งให้เซอร์วิส MongoDB หยุดทำงาน ก็ให้ใช้คำสั่งดังนี้

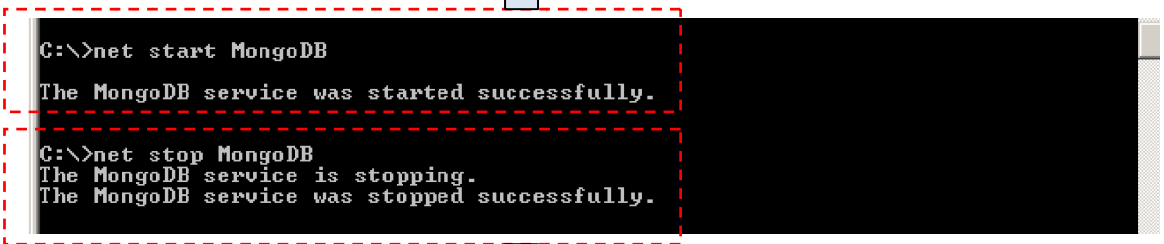
```
net stop MongoDB
```

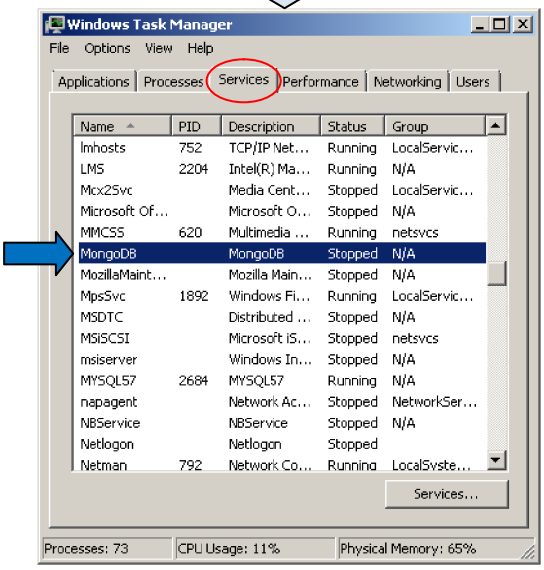
5) ถ้าจะลบเซอร์วิส MongoDB ก็ให้ใช้คำสั่งดังนี้

```
"C:\mongodb\bin\mongod.exe" --remove
```

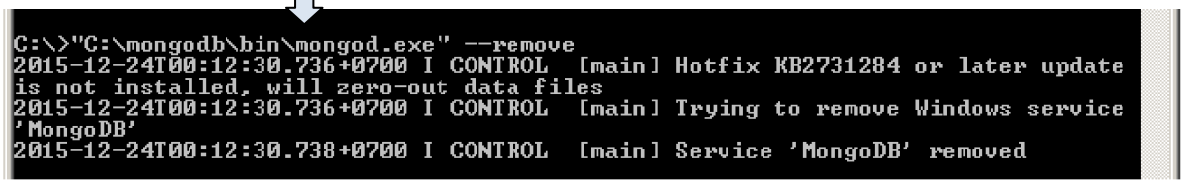
สำหรับตัวอย่างการทำงานของคำสั่งในข้อ 3, 4 และ 5 ก็ให้ดูรูปในหน้าถัดไปได้เลยครับ







เมื่อจะลบเซอร์วิส ก็ให้พิมพ์คำสั่งดังนี้



สำหรับ OS อื่น ๆ นอกจากวินโดวส์ ☆

ถ้าจะติดตั้ง และรัน mongoDB บน Linux ก็ให้ไปอ่านเอกสารได้ที่

https://docs.mongodb.org/v3.0/administration/install-on-linux/

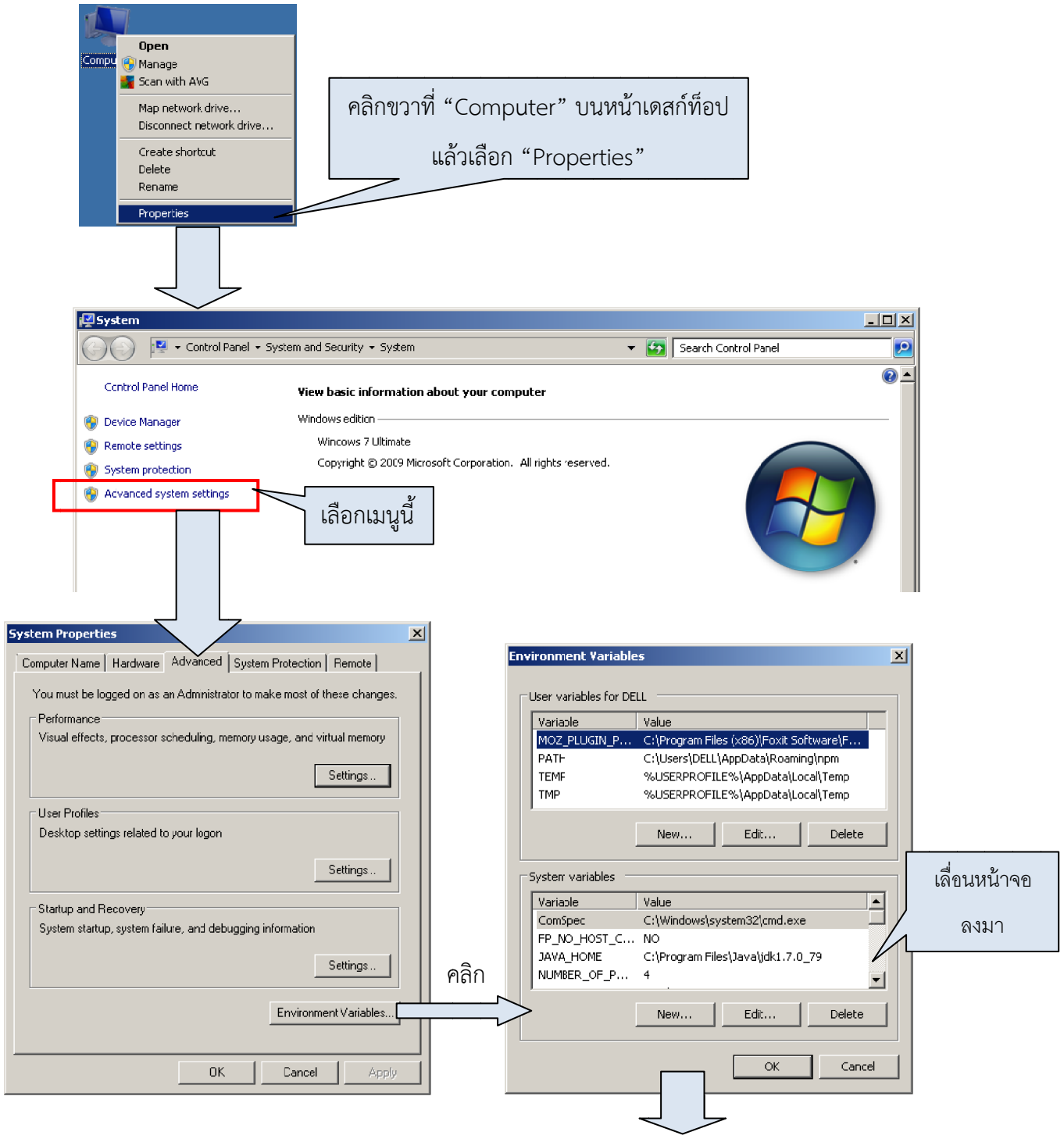
หรือถ้าติดตั้ง และรัน mongoDB บน OS X ก็ให้ไปอ่านเอกสารได้ที่

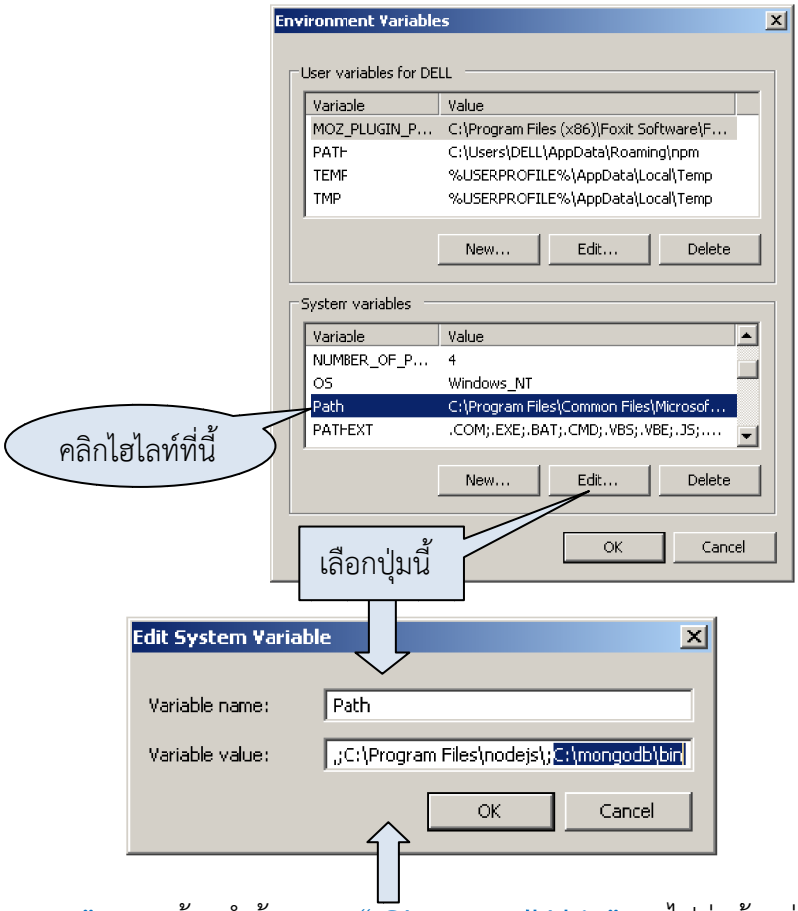
https://docs.mongodb.org/v3.0/tutorial/install-mongodb-on-os-x/

เซทพาธบนวินโดวส์

เนื่องจากคำสั่งต่าง ๆ ของ MongoDB บนคอมมานไลน์ของผู้เขียน จะเก็บอยู่ที่ C:\mongodb\bin ซึ่งทำให้ยุ่งยาก เพราะต้องเรียกคำสั่งผ่านพาธ (pat) เต็ม เช่น "C:\mongodb\bin\mongod.exe" เป็นต้น

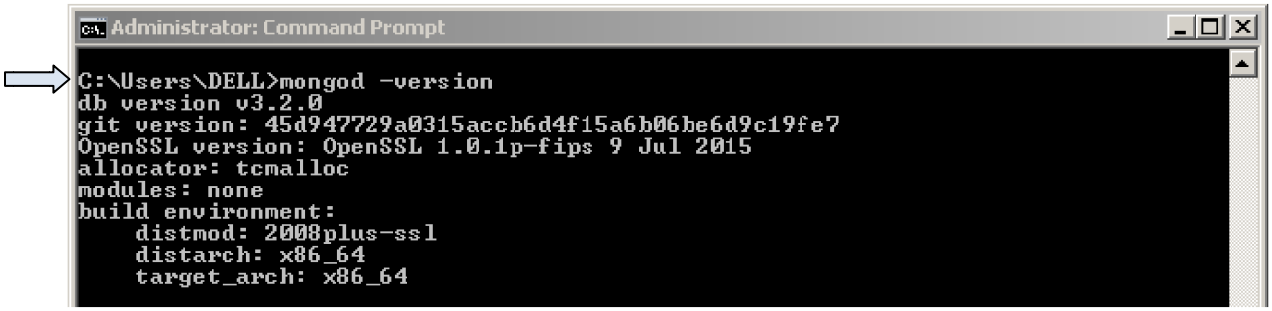
...แต่เราไม่ต้องระบุพาธเต็มก็ได้ ด้วยการเข้าไปแก้ Environment variable (ตัวแปรของระบบ) ดังภาพ





***ในภาพนี้ ...ค่า “Variable name:” จะต้องนำข้อความ “;C:\mongodb\bin” ไปต่อท้ายค่า “Variable value:” ตัวเก่า (ห้ามทับค่าเดิม) หลังจากนั้นก็กด “OK”→ “OK” → “OK” เป็นอันสิ้นสุด

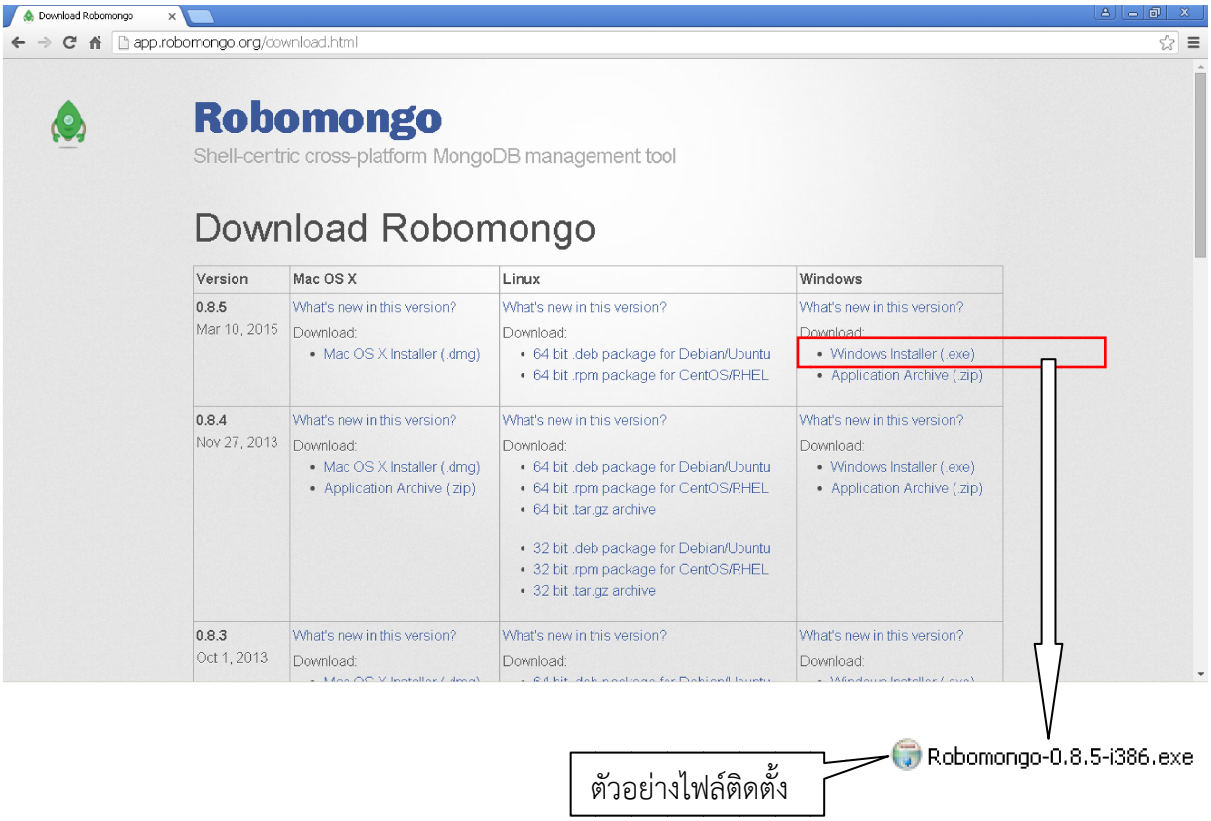
...เราสามารถทดสอบได้ว่า ตั้งพาธเสร็จเรียบร้อยแล้วหรือไม่ ด้วยการพิมพ์ “mongod -version” บนคอมมานไลน์ ที่ไดเรคทอรี่ไหนก็ได้ ซึ่งควรจะเห็นเลขเวอร์ชั่นตามภาพ (เลขเวอร์ชั่นจะเปลี่ยนไปตามไฟล์ติดตั้ง ที่เราดาวน์โหลดมา)



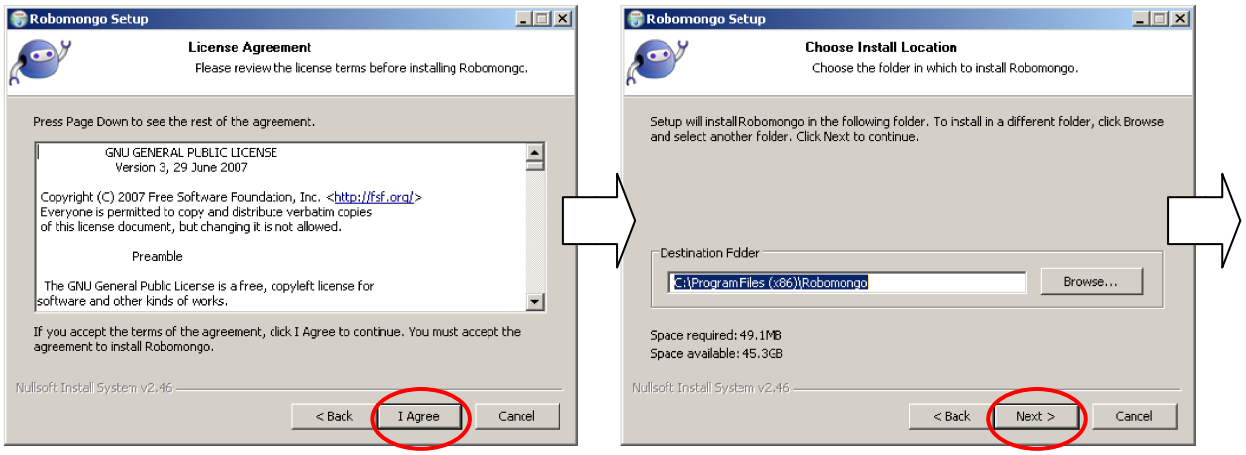
ติดตั้ง Robomongo

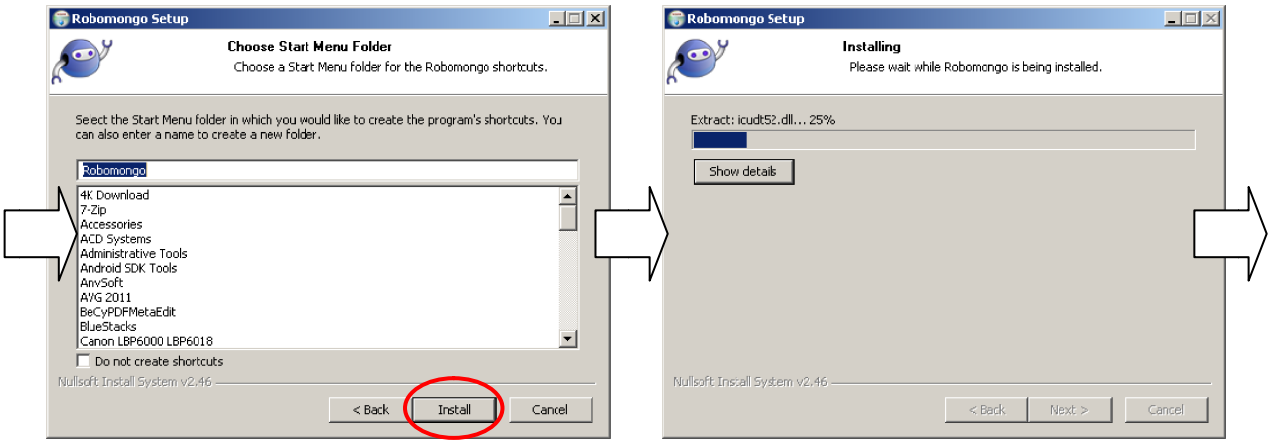
เนื่องจากการใช้ MongoDB ผ่านทางคอมมานไลน์ตรง ๆ บางคนอาจไม่ถนัด จึงอาจติดตั้งเครื่องมือช่วย ซึ่งในหนังสือนี้จะใช้ Robomongo โดยมีวิธีติดตั้งดังขั้นตอนต่อไปนี้

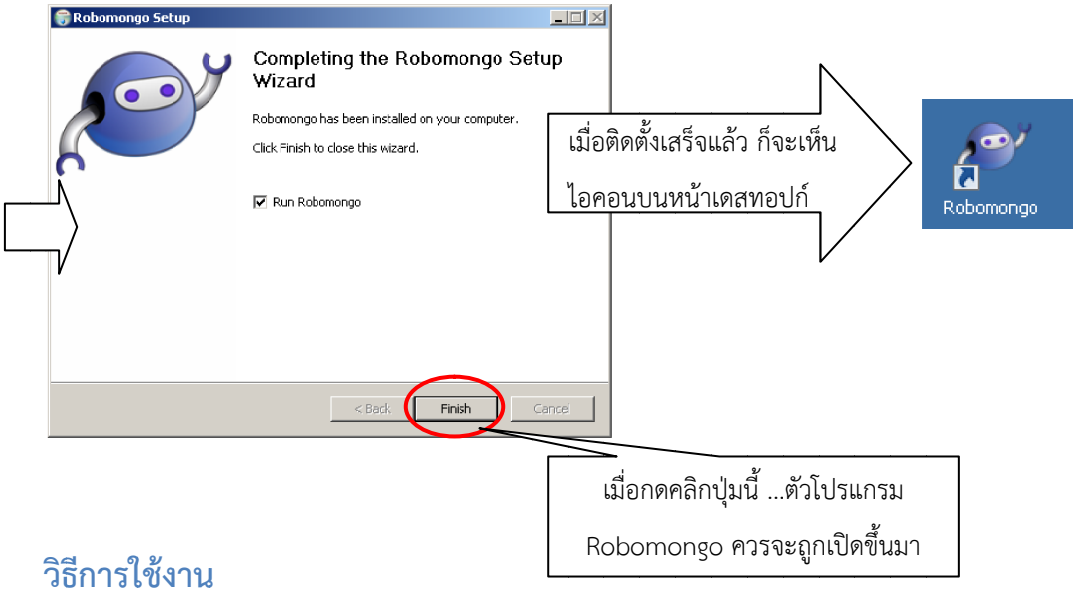
1) ให้ไปที่ลิงค์ http://app.robomongo.org/download.html แล้วดาวน์โหลดไฟล์ติดตั้ง ซึ่งผมจะเลือกเวอร์ชั่นที่ติดตั้งลงบนวินโดวส์



2) ดับเบิลคลิกที่ไฟล์ติดตั้ง เหมือนลงโปรแกรมปกติธรรมง่าย ๆ ไม่ยากครับ

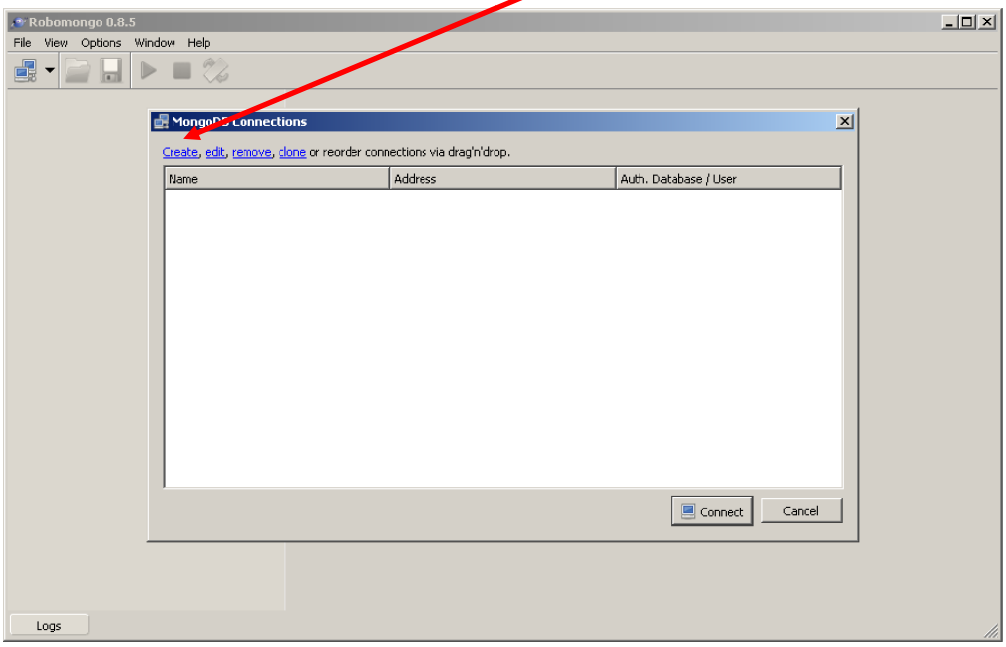




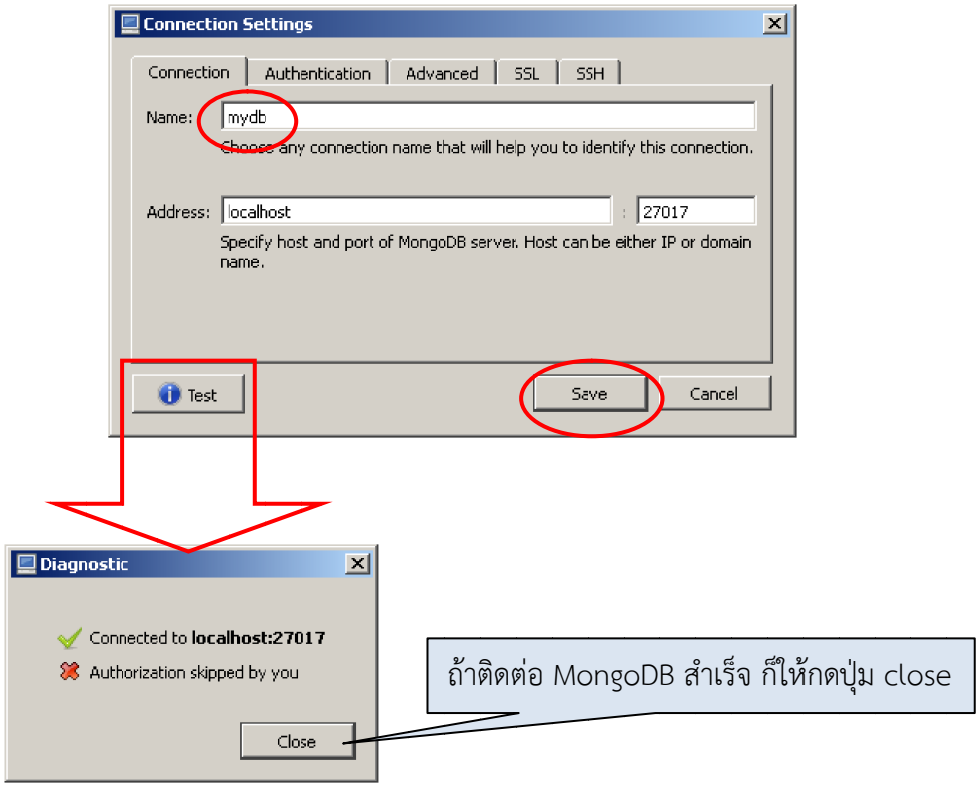


วิธีการใช้งาน

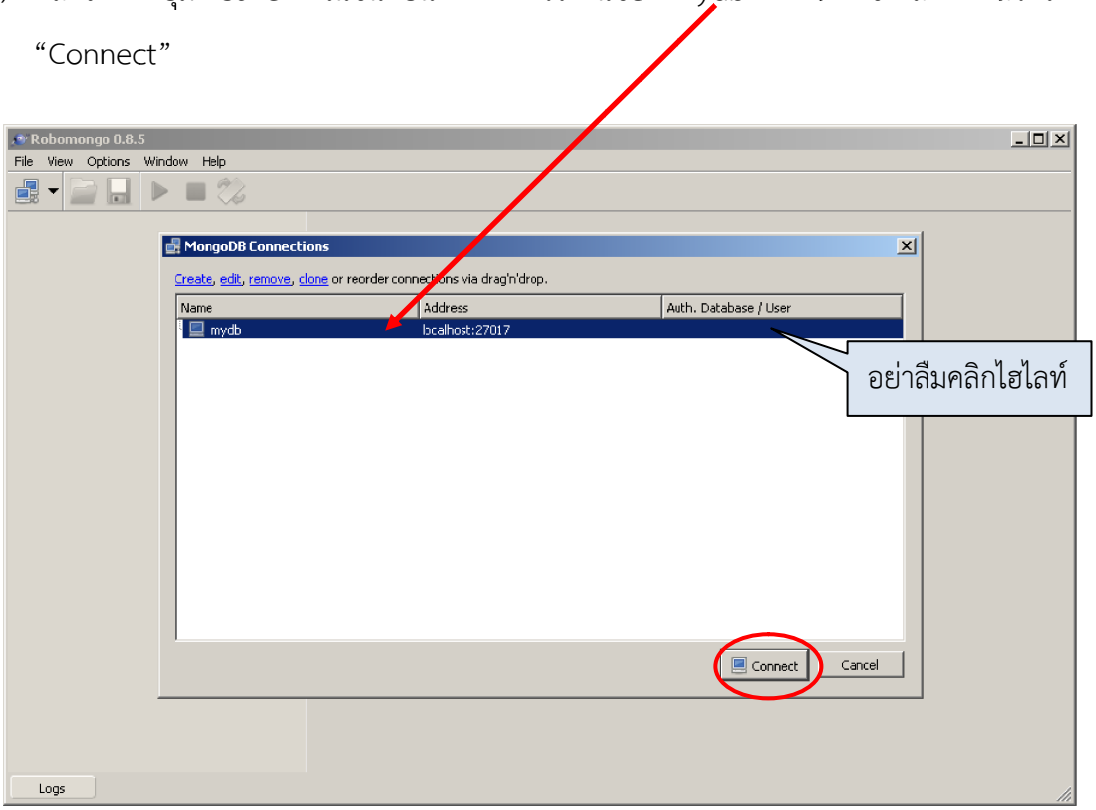
1) ให้เปิดโปรแกรม Robomongo ขึ้นมา แล้วก็คลิกเลือก “Create” (ปุ่มเล็กนิดเดียว)



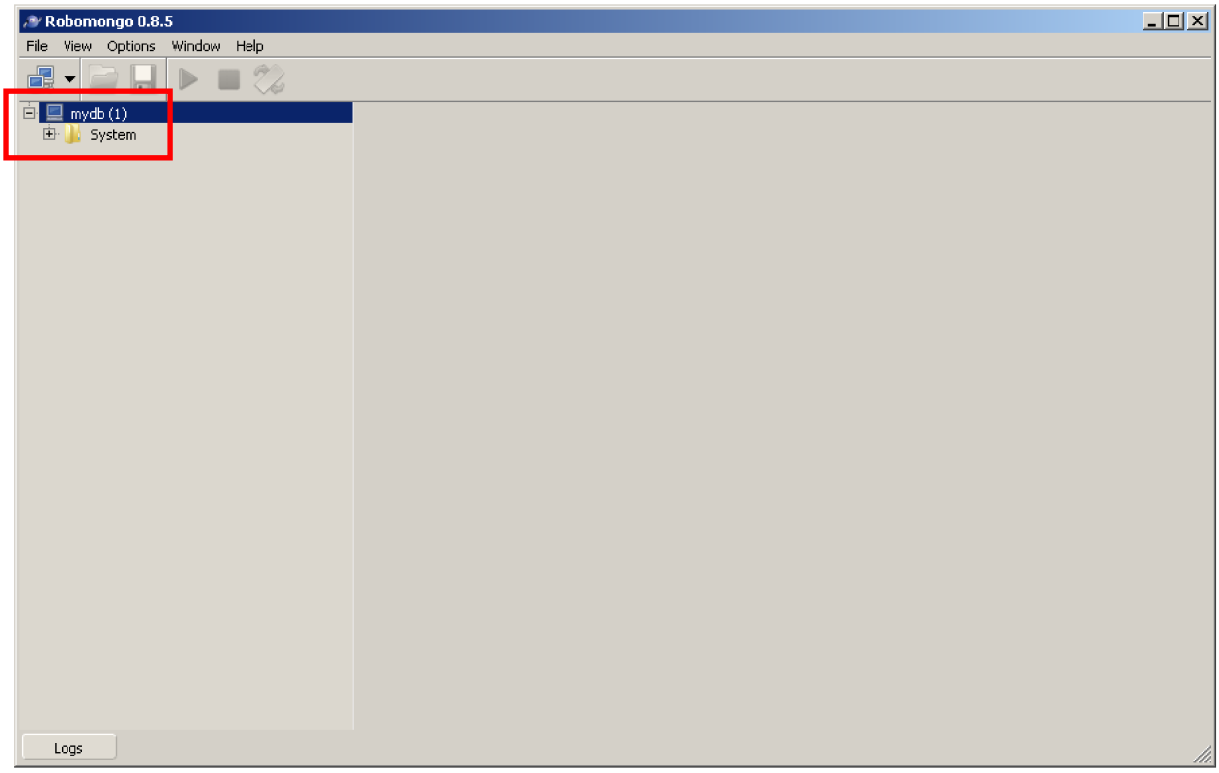
2) ตั้งชื่อ “Connection” ซึ่งในตัวอย่างนี้ตั้งชื่อเป็น “mydb” หลังจากนั้นก็ให้กด “Save” หรืออาจกด “Test” ก่อน เพื่อทดสอบดูว่าติดต่อกับ MongoDB ได้หรือไม่ (อย่าลืมเปิด MongoDB ด้วยละ)



3) หลังจากกดปุ่ม “Save” ในขั้นตอนที่ 2 ก็ควรเห็นชื่อ “mydb” ดังภาพข้างล่าง แล้วก็ให้กดปุ่ม “Connect”



4) ต่อมาผมก็จะเห็นข้อมูลการเชื่อมต่อ ดังรูปข้างล่าง



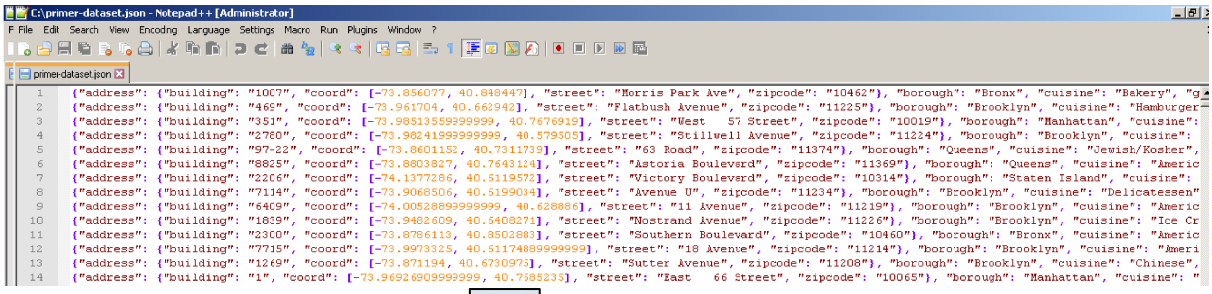
การนำเข้าข้อมูล

ในบทนี้ผมจะทำเวิร์คช็อปอย่างง่าย ...ด้วยการนำเข้าข้อมูล (Import) สู่ MongoDB ดังต่อไปนี้

1) ผมจะไปที่ลิงค์ข้างล่าง ซึ่งเป็น**ไฟล์ข้อมูลตัวอย่าง** ที่จะนำเข้า MongoDB

https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/dataset.json

แล้วบันทึกมันเป็นชื่อ “primer-dataset.json” ซึ่งถ้าเปิดไฟล์ขึ้นมาดู ก็จะเห็นแต่ละบรรทัด มันประกอบไปด้วยข้อมูลที่เขียนเป็น JSON ตามรูปข้างล่าง ...ซึ่งผมจะเก็บมันไว้ที่ “C:/primer-dataset.json”



สำหรับข้อมูล JSON ก็จะมีโครงสร้าง

เป็นคู่ Key กับ value

ซึ่งในตัวอย่างนี้ จะมีหน้าตาแบบนี้ครับ

{"address":

{

"building": "1007",

"coord": [-73.856077, 40.848447],

"street": "Morris Park Ave", "zipcode": "10462"

},

"borough": "Bronx",

"cuisine": "Bakery",

"grades": [{"date": {"$date": 1393804800000}, "grade": "A", "score": 2}, {"date": {"$date": 1378857600000}],

"name": "Morris Park Bake Shop",

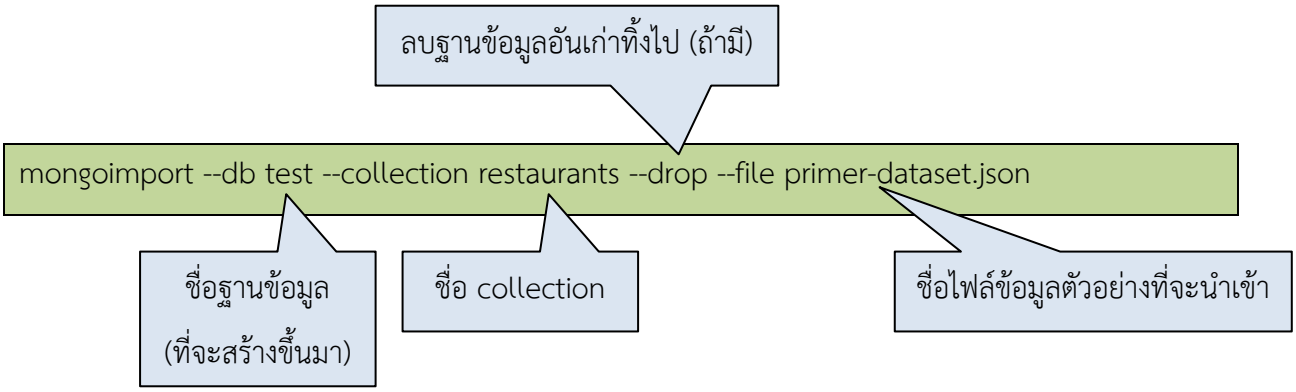
"restaurant_id": "30075445"

}

*** สำหรับข้อมูล JSON ถ้าใครอยู่สายเว็บไซต์ ก็น่าจะรู้จักกันดีอยู่แล้วแหละ โดยมันเป็นชนิดข้อมูลแบบหนึ่ง ซึ่งจะนิยมใช้แลกเปลี่ยนข้อมูลระหว่างไคลเอนต์กับเซิร์ฟเวอร์

แต่ถ้าใครไม่รู้จัก อาจลองไปศึกษาค้นคว้าในอินเตอร์ดู ...ไม่ยากครับ ง่าย ๆ

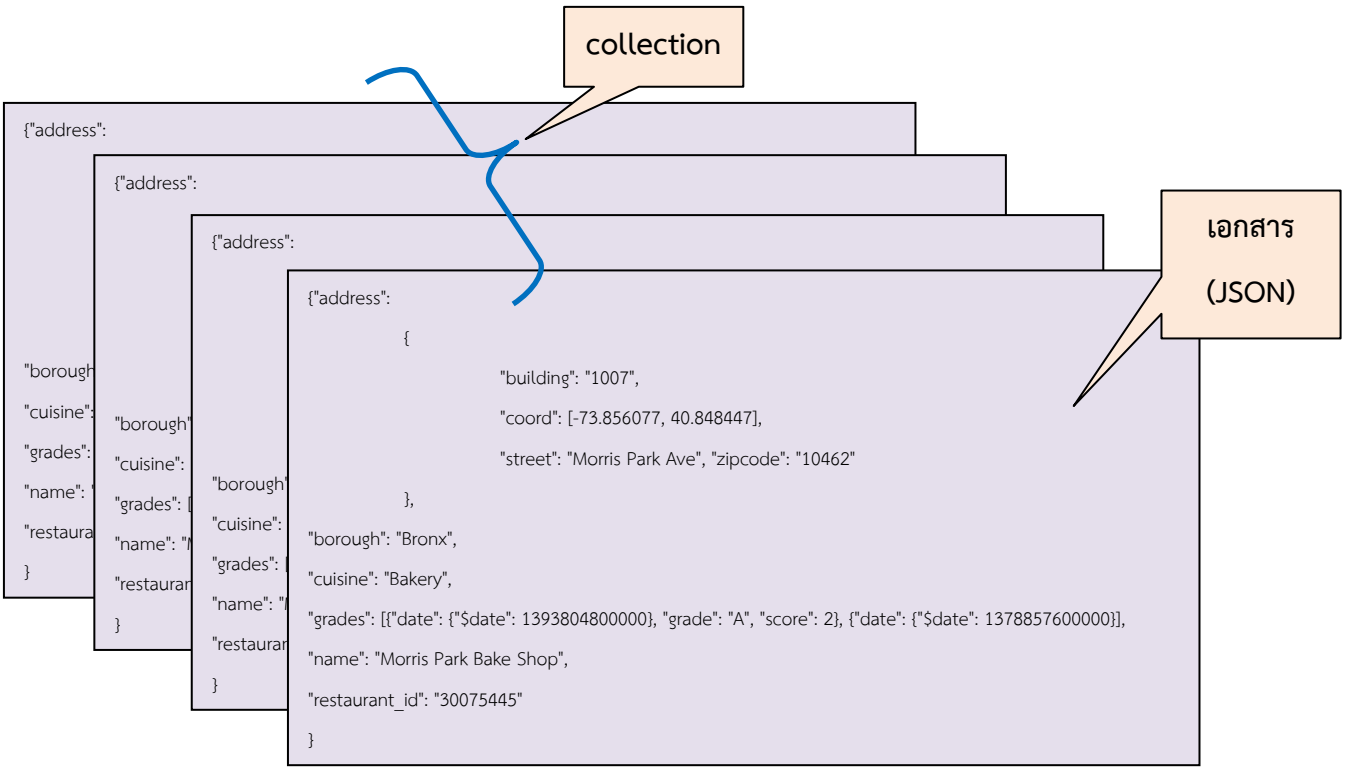
2) ไปไดเรคเทอรี่ C:\ ซึ่งได้เก็บไฟล์ในข้อ 1 แล้วพิมพ์คำสั่งต่อไปนี้ เพื่อนำเข้าข้อมูลตัวอย่าง



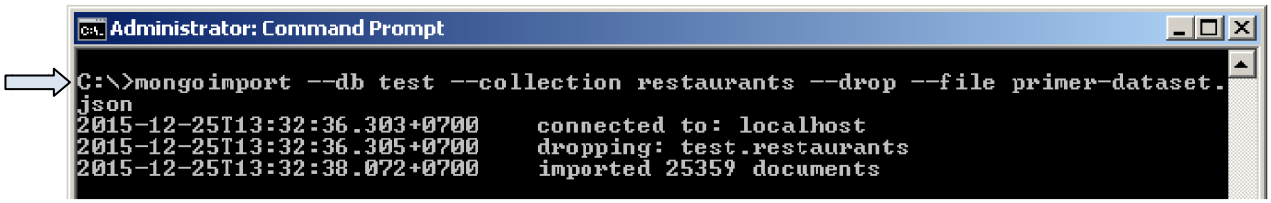
ขออธิบายแทรกนิดหนึ่ง ผมอยากให้คุณลืมเรื่องฐานข้อมูลแบบ SQL ไปก่อนเลย เพราะ MongoDB ไม่ได้นำ JSON มาเก็บเป็นแถวในตาราง แล้ว Normalize โน่นนี้นั้น (อุตสาเรียนมาตั้งยาก อยู่ดี ๆ ไม่ใช้คิดดู) ...แต่ทว่า ฐานข้อมูลก้อนหนึ่ง จะมีชุดข้อมูล JSON มากองอยู่รวมกันเป็นก้อนหนึ่ง (เก็บง่าย ๆ แบบนี้แหละ)

...ชุดข้อมูลของ JSON จะเรียกว่า **"collection"** (ที่แปลว่า "การเก็บ" ซึ่งในที่จะเก็บเอกสารเป็น JSON)

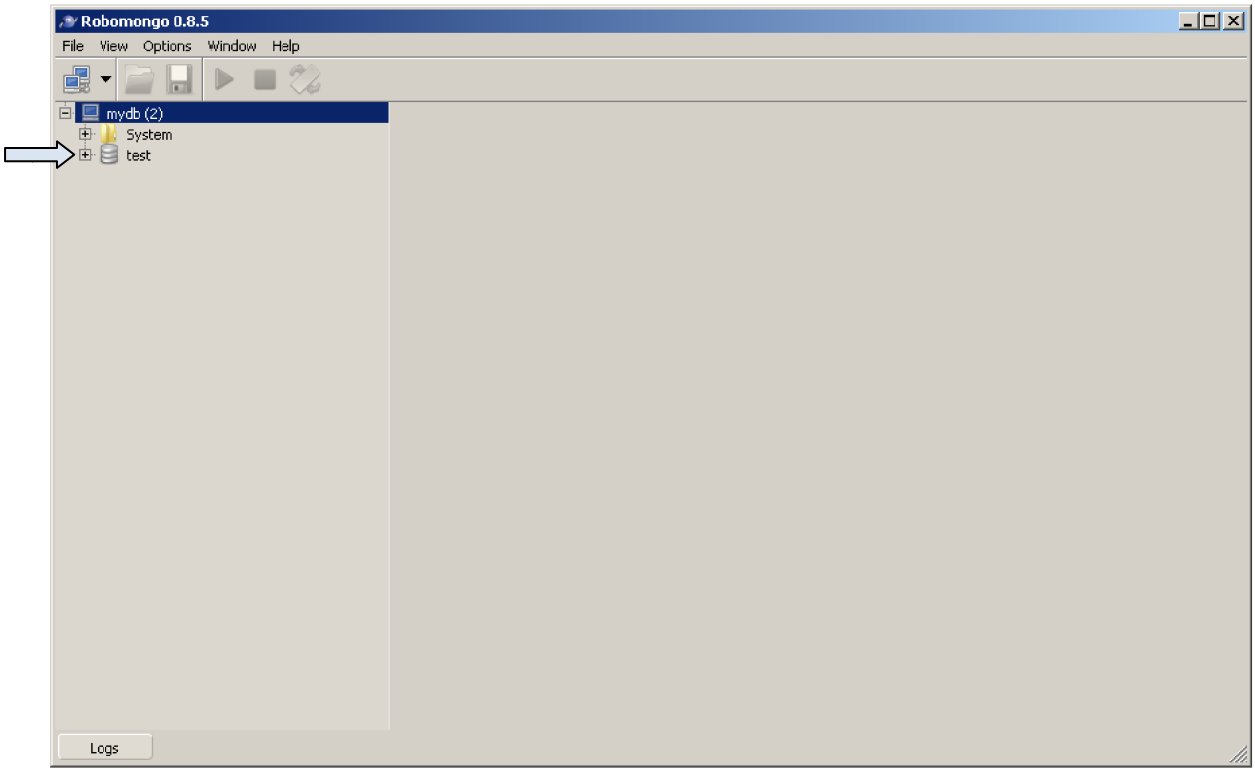
....ส่วนข้อมูล JSON จะเรียกว่า **"เอกสาร"** (document)



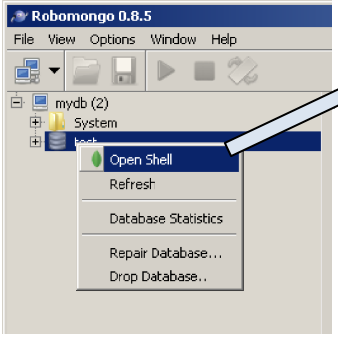
จากคำสั่งที่เขียนไว้ในตอนต้น เมื่อผมพิมพ์ลงไป ก็จะได้ผลลัพธ์ดังนี้

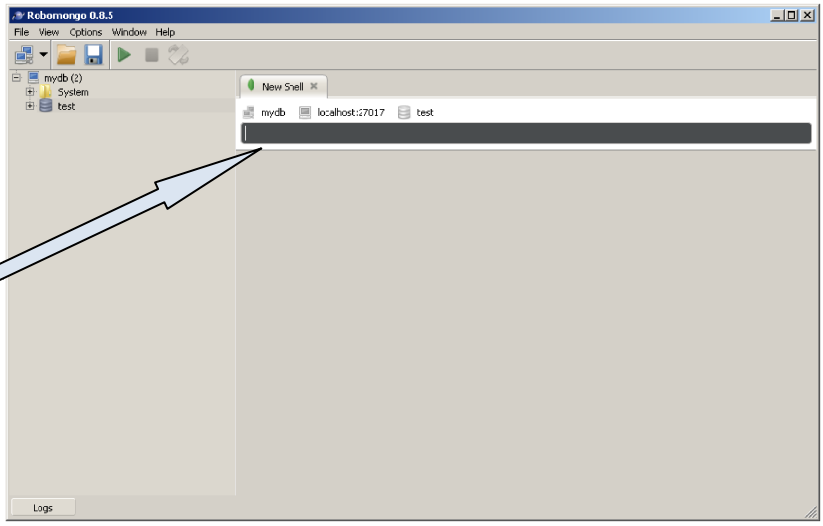


เมื่อผมลองเปิด Robomongo ขึ้นมา พร้อมทั้งเชื่อมต่อไปยัง MongoDB ตามที่ตั้งค่าไว้ในบทก่อนหน้านี้ (ชื่อการเชื่อมต่อคือ “mydb”) ...ก็จะเห็นว่ามีฐานข้อมูลชื่อ “test” ถูกสร้างขึ้นมา



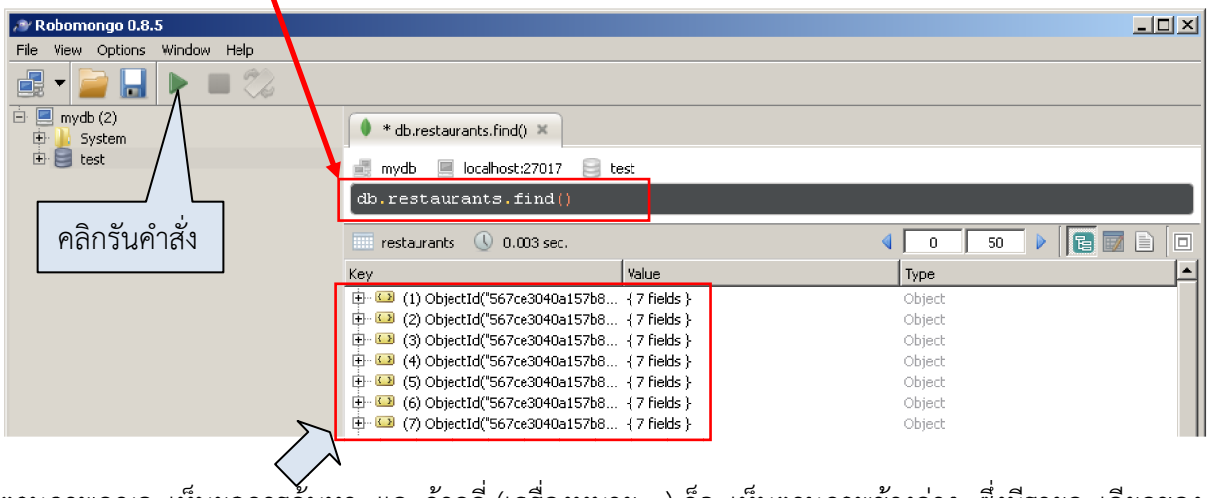
หลังจากนั้นเมื่อผมคลิกขวาชื่อ “test” แล้วเปิด “Open shell” ขึ้นมา



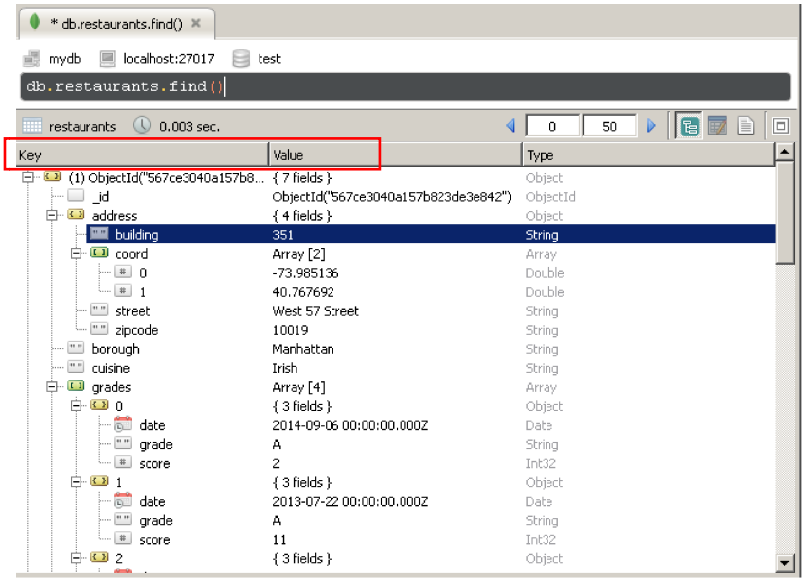


จากนั้นผมจะพิมพ์คำสั่งในช่องคอมมานไลน์ของ Robomongo เพื่อค้นหาเอกสารจาก collection ที่ชื่อ “restaurants” (ถ้านึกไม่ออก ก็คล้ายคำสั่ง select * from restaurants) ดังนี้

```
db.restaurants.find()
```



ตามภาพคุณจะเห็นผลการค้นหา และถ้าคลี่ (เครื่องหมาย +) ก็จะเห็นตามภาพข้างล่าง ซึ่งมีรายละเอียดของค่า Key กับ Value ของเอกสาร JSON ทั้งชุดเลย (เก็บง่ายมาก)



สำหรับรายละเอียดของโครงสร้าง JSON ที่เก็บใน MongoDB ยังมีเรื่องราวอีกพอควร เพราะมันสามารถมี JSON หรืออาร์เรย์ ซ้อนอยู่ข้างในได้หลาย ๆ ระดับชั้น แต่เล่มนี้จะอธิบายให้เห็นแค่นี้ก่อน ...ไม่ลงลึกมาก

ถ้าผมอยากจะค้นหาเอกสารด้วยการใช้คอมมานไลน์ ก็แลดูยุ่งยากสักหน่อย เพราะต้องติดต่อเข้าไปยังฐานข้อมูล “test” ก่อน ดังนี้

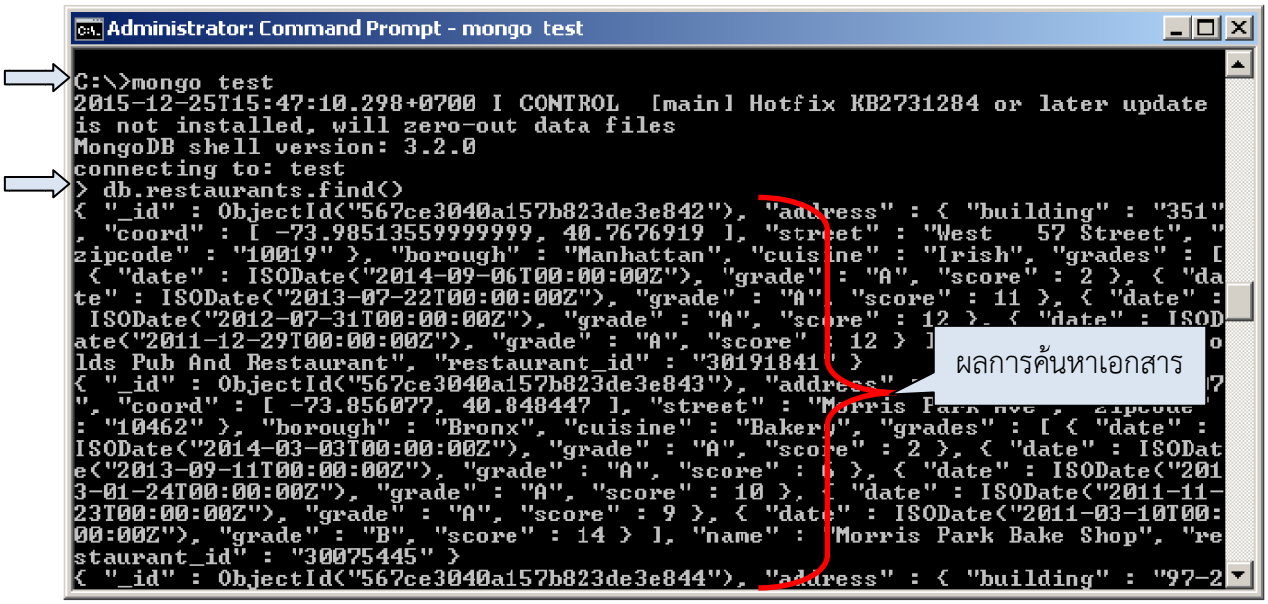
```
C:\>mongo test
```

ชื่อฐานข้อมูล

หลังจากนั้นจึงพิมพ์คำสั่ง

```
db.restaurants.find()
```

ลองดูตัวอย่างคำสั่งบนคอมมานไลน์เต็ม ๆ ดังหน้าจอข้างล่าง ...ซึ่งจะอ่านลำบากมาก 555



หวังว่าคุณคงเห็นไอเดียเวลาใช้งาน MongoDB บนคอมมานไลน์ ...โดยไม่ผ่านเครื่องมือช่วย (เช่น Robomongo) ซึ่งผมจะขอสรุปวิธีใช้งานผ่านคอมมานไลน์ได้ดังนี้

1. ต้องติดต่อเข้าไปยังฐานข้อมูลก่อน (ในตัวอย่างนี้คือ “test”)
2. เวลาค้นหา (รวมทั้ง ลบ อัพเพตข้อมูล และอื่น ๆ) ก็ให้มาทำที่ collection ไม่ใช่ตารางแบบ SQL

คำถาม เวลาเขียนโปรแกรม ถ้าไม่ใช้ภาษา SQL คุยกับ MongoDB ...แล้วใช้ภาษาหลักอะไรในการคุย?

คำตอบ ใช้ภาษาอังกฤษ

...แฮ ๆ ๆ ล้อเล่นครับ มันใช้หลากหลายภาษาเขียนโปรแกรมมาก ๆ

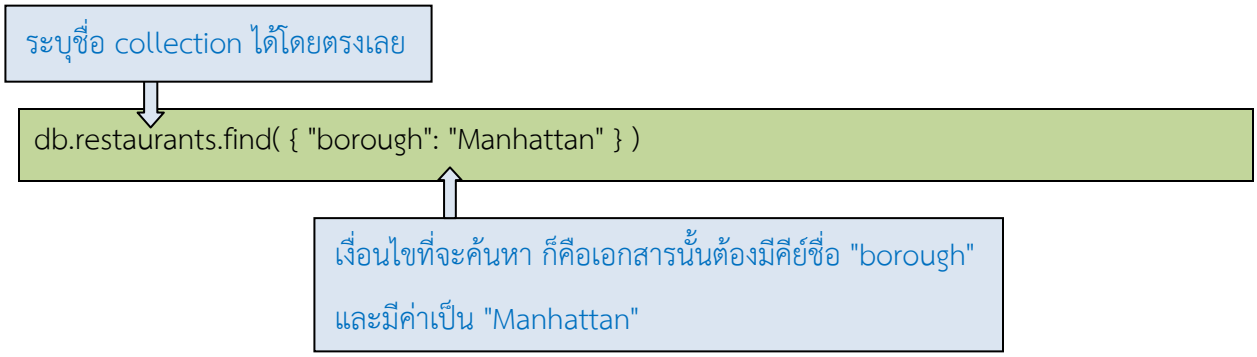
ภาษาที่ใช้คุยกับ MongoDB

เนื่องจาก MongoDB ไม่ใช้ภาษา SQL แต่จะใช้ API ซึ่งขึ้นอยู่กับแต่ละภาษา ดังนั้นภาษาเขียนโปรแกรมที่ MongoDB รองรับได้ ก็คือ...

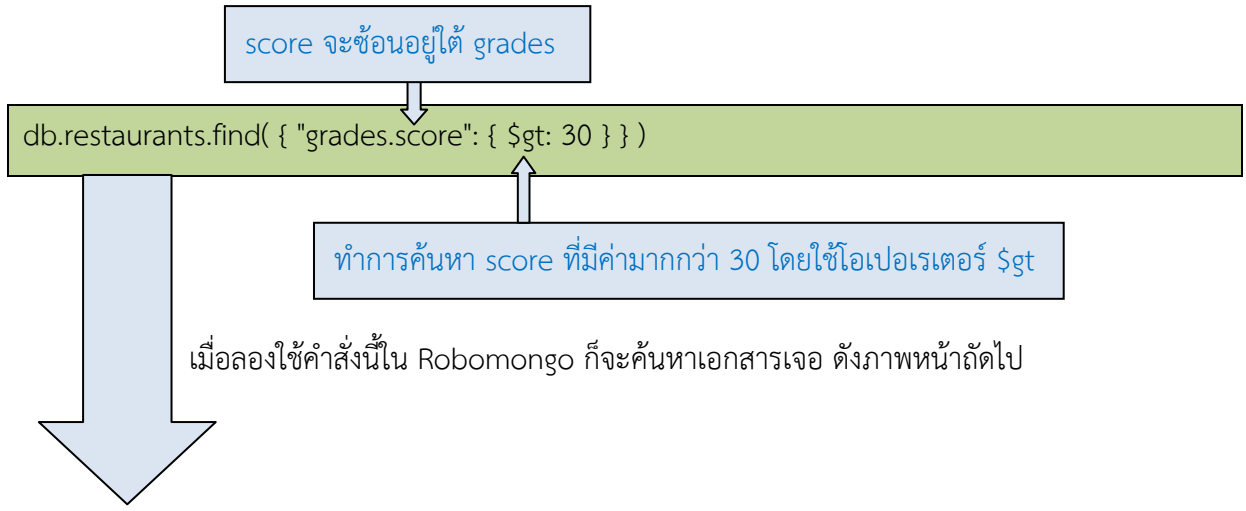
1. ใช้จาวาสคริปต์บน Node.js (เล่มนี้จะกล่าวถึง)
2. Python (โค้ดจะสั้นกว่าภาษาอื่นเยอะเลย แต่ทว่าเล่มนี้ไม่ได้กล่าวถึง)
3. C++ (เล่มนี้ไม่ได้กล่าวถึง)
4. Java (เล่มนี้ไม่ได้กล่าวถึง)
5. C# (เล่มนี้ไม่ได้กล่าวถึง)

ไม่เพียงเท่านั้น เรายังสามารถใช้คำสั่งคอมมานไลน์ของ MongoDB ได้โดยตรง ซึ่งผมจะอธิบายโดยย่อ ...แต่ถ้าอยากอ่านรายละเอียดมากกว่านี้ ก็ให้ไปที่ https://docs.mongodb.org/getting-started/shell/ เพราะผมก็สรุปมาจากที่นี้นี่แหละ **...และต่อไปเราจะลองมาดูคำสั่งคอมมานไลน์ของ MongoDB โดยตรงกัน**

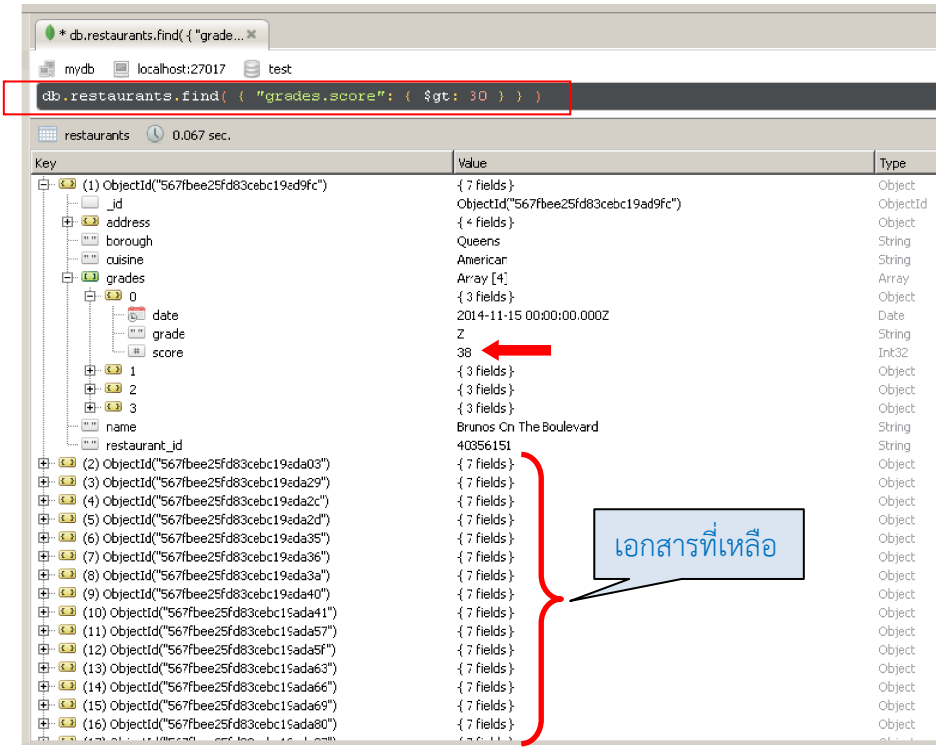
ถ้าต้องค้นหาเอกสาร ก็ให้ระบุเงื่อนไข (Criteria) ในการค้นหา ดังตัวอย่าง



หรือจะระบุเงื่อนไขในการค้นหาเพิ่มเติมด้วยก็ได้



เมื่อลองใช้คำสั่งนี้ใน Robomongo ก็จะค้นหาเอกสารเจอ ดังภาพหน้าถัดไป



เราสามารถอัพเดตเอกสาร ด้วยคำสั่งดังตัวอย่าง

```
db.restaurants.update(
    { "name" : "Juni" },
    {
      $set: { "cuisine": "Thailand (New)" },
      $currentDate: { "lastModified": true }
    }
)
```

ในตัวอย่างนี้จะอัพเดตเอกสารตัวแรกที่เจอ เมื่อเงื่อนไขเป็น { "name" : "Juni" } แถมมีเงื่อนไขตามโอเปอเรเตอร์เป็น $set กับ $currentDate ตามลำดับ (โอเปอเรเตอร์เหล่านี้จะเห็นอีกทีในบทถัดไป)

เราสามารถลบเอกสารออกได้ ด้วยการระบุเงื่อนไข ดังตัวอย่าง

```
db.restaurants.remove( { "borough": "Manhattan" } )
```

หรือจะลบ restaurants ทิ้งไปเลยก็ได้ (แนวคิดเหมือนคำสั่ง drop ที่มีใช้ในภาษา SQL)

```
db.restaurants.drop()
```

หรือจะเพิ่มเอกสารลงไป ก็ให้ใช้คำสั่งดังตัวอย่าง

```
db.restaurants.insert(
  {
    "address" : {
       "street" : "2 Avenue",
       "zipcode" : "10075",
       "building" : "1480",
       "coord" : [ -73.9557413, 40.7720266 ],
    },
    "borough" : "Manhattan",
    "cuisine" : "Italian",
    "grades" : [
       {
          "date" : ISODate("2014-10-01T00:00:00Z"),
          "grade" : "A",
          "score" : 11
       },
       {
          "date" : ISODate("2014-01-16T00:00:00Z"),
          "grade" : "B",
          "score" : 17
       }
    ],
    "name" : "Vella",
    "restaurant_id" : "41704620"
  }
)
```

ข้อมูล JSON ยาวหน่อย ซึ่งอาจไม่คุ้นชินกับคนที่เคยเขียนภาษา SQL มาก่อน

*** ถ้ายังไม่เห็นภาพการทำงาน ก็ขอให้ดูตัวอย่างในบทถัดไป ต่อจากนี้แล้วกันนะครับ

....บทนี้ขอเกริ่นนำคร่าว ๆ ไปก่อน

ใช้ Node.js

ในบทนี้จะกล่าวถึงการใช้จาวาสคริปต์เขียนเป็นสคริปต์ เพื่อติดต่อคุยกับ MongoDB เบื้องต้น แต่ทว่าไฟล์สคริปต์ดังกล่าวจะใช้ Node.js เป็นตัวรันอีกที (ไม่ใช่เว็บบราวเซอร์) ด้วยเหตุนี้ถ้าอยากอ่านเนื้อหาต่อไปรู้เรื่อง คุณต้องไปที่ลิงค์ http://www.patanasongsivilai.com/itebook_form.html

1) เพื่อดาวน์โหลด "วิธีติดตั้ง Node.js และ npm เบื้องต้น" มาอ่านก่อน
2) และอยากให้คุณอ่านหนังสือ "เสียดายไม่ได้อ่านจาวาสคริปต์ ฝั่งเซิร์ฟเวอร์ (Node.js ฉบับย่อ)" เล่ม 1 มาอ่านเสริมเพิ่มด้วย

เอาแหละ ผมจะคิดว่าคุณเข้าใจ Node.js ดีแล้ว ...แต่ถ้าไม่เข้าใจ ก็ลองอ่านเนื้อหาต่อไปนี้คร่าว ๆ เป็นไอเดียแล้วกันเนอะ ไม่ยาก ...แต่จะมึนโค้ดหน่อย โดยผมจะพาทำเวิร์คช็อปง่าย ๆ ดังนี้

เตรียมโปรเจคให้พร้อม

สร้างโฟลเดอร์ขึ้นมาดังนี้

```
C:\>mkdir testdb
C:\>cd testdb
C:\ testdb>
```

หลังจากนั้นจะใช้ npm ติดตั้งมอดูล "mongoose" ดังนี้

```
C:\ testdb>npm install mongoose --save
```

ส่วนโครงสร้างโปรเจคจะเป็นดังนี้

```
C:\ testdb
    |-- node_modules\...
    |-- test.js
```

ต้องบอกอย่างนี้นะครับ ...ไอเดียผมจะเขียนโค้ดจาวาสคริปต์ทั้งหมดไว้ที่ไฟล์ “test.js” และสั่งรันบน Node.js ด้วยคำสั่งดังนี้

```
C:\testdb>node test.js
```

ซึ่งไฟล์ test.js และผลการทำงานของมัน ผมก็จะเอาไว้ใช้อธิบายเนื้อหาต่อไปถัดจากนี้

เขียนส่วนหัวของไฟล์ test.js

ผมจะเขียนส่วนหัวของไฟล์ test.js แบบนี้เสมอ ดังโค้ดข้างล่าง

```
var MongoClient = require('mongodb').MongoClient;   // โหลดมอดูล mongodb
var assert = require('assert');                      // unit test
// var ObjectId = require('mongodb').ObjectID;
var url = 'mongodb://localhost:27017/test';          // เป็น url ที่จะติดต่อกับ MongoDB
```

แต่ให้สังเกตค่า URL ต้องตั้งค่าตาม MongoDB ที่เราได้ติดตั้งลงไป ด้วยรูปแบบดังนี้

```
'mongodb://ip_address_database:port/database_name';
```

- ip_address_database คือ ไอพีแอดเดรส (IP Address) ของเครื่องที่กำลังรัน MongoDB (กรณีนี้เข้าถึงเครื่องตัวเอง)
- port คือชื่อพอร์ตที่เปิดไว้ (ในหนังสือเปิดพอร์ตเป็น 27017)
- database_name คือชื่อฐานข้อมูล (ในหนังสือมีชื่อเป็น “test”)

หมายเหตุ การณ์ตั้งค่าคอนฟิคของ MongoDB รวมทั้งการตั้งยูสเซอร์/พาสเวิร์ดเข้าฐานข้อมูล (ตัวอย่างนี้ไม่ต้องตั้งค่า) ...เล่มนี้คงไม่กล่าวถึง เพราะนอกขอบเขตไป

ค้นหาข้อมูล

ผมจะทำการค้นหา (Query) เอกสารทั้งหมด จาก collection ที่ชื่อ "restaurants" ของบทก่อนหน้า ด้วยการระบุเงื่อนไข ...ดังโค้ดที่อยู่ในไฟล์ “test.js” ดังตัวอย่าง

```
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
var url = 'mongodb://localhost:27017/test';

var findRestaurants = function(db, callback) {
        // คำสั่งนี้จะค้นหาเอกสารใน restaurants ด้วยการระบุเงื่อนไข ...มีคีย์ชื่อ "borough" และมีค่าเป็น "Manhattan"
        var cursor =db.collection('restaurants').find({ "borough": "Manhattan" } );

        // ตัวแปร cursor จะชี้ไปยังเอกสารที่ค้นพบ จะคล้ายกับ cursor เวลาใช้งานในฐานข้อมูลแบบ SQL
        cursor.each(function(err, doc) {   // แต่ละรอบที่ each() เข้าถึงเอกสารใน restaurants ฟังก์ชั่นคอลแบ็คจะถูกเรียกให้ทำงาน
                assert.equal(err, null);
                if (doc != null) {
                        console.dir(doc);     // ฟังก์ชั่นนี้จะแสดงโครงสร้างอ็อบเจ็กต์ doc (ไม่ใช่ฟังก์ชั่นมาตรฐานในจาวาสคริปต์)
                } else {
                        callback();     // เมื่อเข้าถึงเอกสารใน restaurants ครบทุกตัวแล้ว ก็จะเรียกฟังก์ชั่นคอลแบ็คให้ทำงาน
                }
        }); // สิ้นสุด cursor.each()
};

MongoClient.connect(url, function(err, db) {
        assert.equal(null, err);

        findRestaurants(db, function() { // เมื่อค้นหาข้อมูลใน restaurants เสร็จแล้ว ฟังก์ชั่นคอลแบ็คจะถูกเรียกให้ทำงาน
            db.close();                  // เมื่อนั้นจึงเลิกติดต่อฐานข้อมูล
          }); // สิ้นสุด findRestaurants()
});
```

โค้ดมันจะยาวไปหน่อย จริง ๆ ไม่อยากให้คุณสนใจรายละเอียดมากเท่าไร ...แค่เห็นเป็นไอเดียพอแหละ

แต่ผมอยากแนะนำว่า ถ้าใครชอบเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ต้องปรับมุมมองให้เห็นเป็นการเขียนโปรแกรมแบบ Function programming หรือการเขียนโปรแกรมเชิงฟังก์ชั่นแท้ ๆ

...แน่นอนละบางคนอาจไม่ชอบเท่าไร เพราะเห็นฟังก์ชั่นซ้อนกันไปซ้อนกันมา ชวนให้ดูตาลาย แถมฟังก์ชั่นคอลแบ็คแต่ละตัว มันยังทำงานแบบอะซิงโครนัสอีกด้วย (ตาลายหนักกว่าเก่าอีก)

....ถ้าใครยังไม่เข้าใจการเขียนโค้ดบน Node.js แนะให้ไปอ่านหนังสือที่ผมแนะนำข้างต้นก่อนครับ

แต่ถึงอย่างไรก็ตาม ผมก็อยากอธิบายเมธอด db.collection().find() เพื่อใช้มันค้นหาเอกสารใน restaurants โดยเราสามารถระบุเงื่อนไขในรูปแบบดังตัวอย่าง

```
{ <field1>: <value1>, <field2>: <value2>, ... }
```

- โดยที่ field1, field2 คือ ชื่อคีย์ในข้อมูล JSON
- value1, value2 คือค่าใน JSON

ซึ่งในตัวอย่างดังกล่าว จะระบุเงื่อนไขในเมธอดเป็น

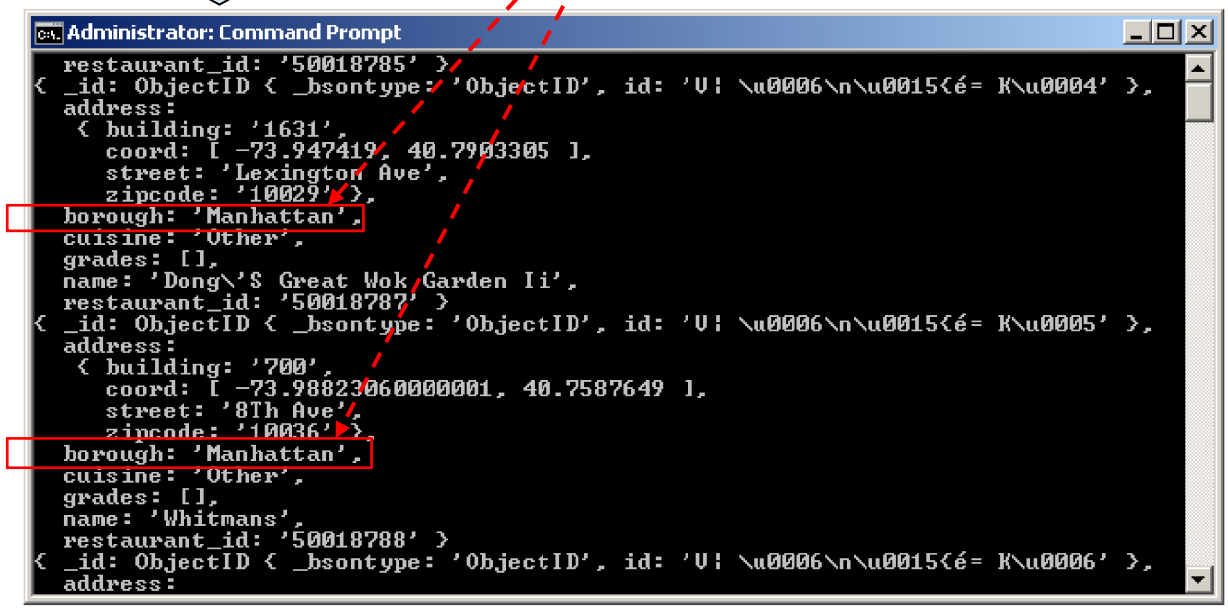
```
{ "borough": "Manhattan" }
```

แต่ถ้าเรียกเป็น db.collection().find() เฉย ๆ โดยไม่ระบุเงื่อนไขอะไรเลย (ไม่ระบุค่าอากิวเมนต์) ก็จะค้นหาเอกสารทั้งหมดใน collection ครับ

ส่วนผลลัพธ์ของโค้ดในตัวอย่างดังกล่าว หลังจากพิมพ์คำสั่ง

```
C:\testdb>node test.js
```

จะเห็นข้อมูล JSON ที่ดูแล้วละลานตาเต็มไปหมด ซึ่งข้างล่างเป็นแค่ส่วนหนึ่งของข้อมูลเท่านั้น ที่มีคีย์ชื่อ "borough" และมีค่าเป็น "Manhattan"



เพิ่มเอกสาร

ตัวอย่างต่อไปนี้จะทำการเพิ่ม (Insert) เอกสารเข้าไปใน restaurants ด้วยการแก้ไขโค้ด test.js ดังนี้

```
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
var url = 'mongodb://localhost:27017/test';

var insertDocument = function(db, callback) {
   db.collection('restaurants').insertOne( {            // เพิ่มเอกสารลงไปใน restaurants
      "address" : {
         "street" : "2 Avenue",
         "zipcode" : "10075",
         "building" : "1480",
         "coord" : [ -73.9557413, 40.7720266 ]
      },
      "borough" : "Manhattan",
      "cuisine" : "Italian",
      "grades" : [
         {
            "date" : new Date("2014-10-01T00:00:00Z"),
            "grade" : "A",
            "score" : 11
         },
         {
            "date" : new Date("2014-01-16T00:00:00Z"),
            "grade" : "B",
            "score" : 17
         }
      ],
      "name" : "Vella",
      "restaurant_id" : "41704621"
   }, function(err, result) {                   // ฟังก์ชั่นคอลแบ็ค
            assert.equal(err, null);

            console.log("Inserted a document into the restaurants collection.");
            callback();
   }); // สิ้นสุด insertOne()
};

// โค้ดชุดนี้คล้ายกับตัวอย่างที่แล้ว
MongoClient.connect(url, function(err, db) {
            assert.equal(null, err);
```

เป็นเอกสารที่เพิ่มเข้าไปใน restaurants

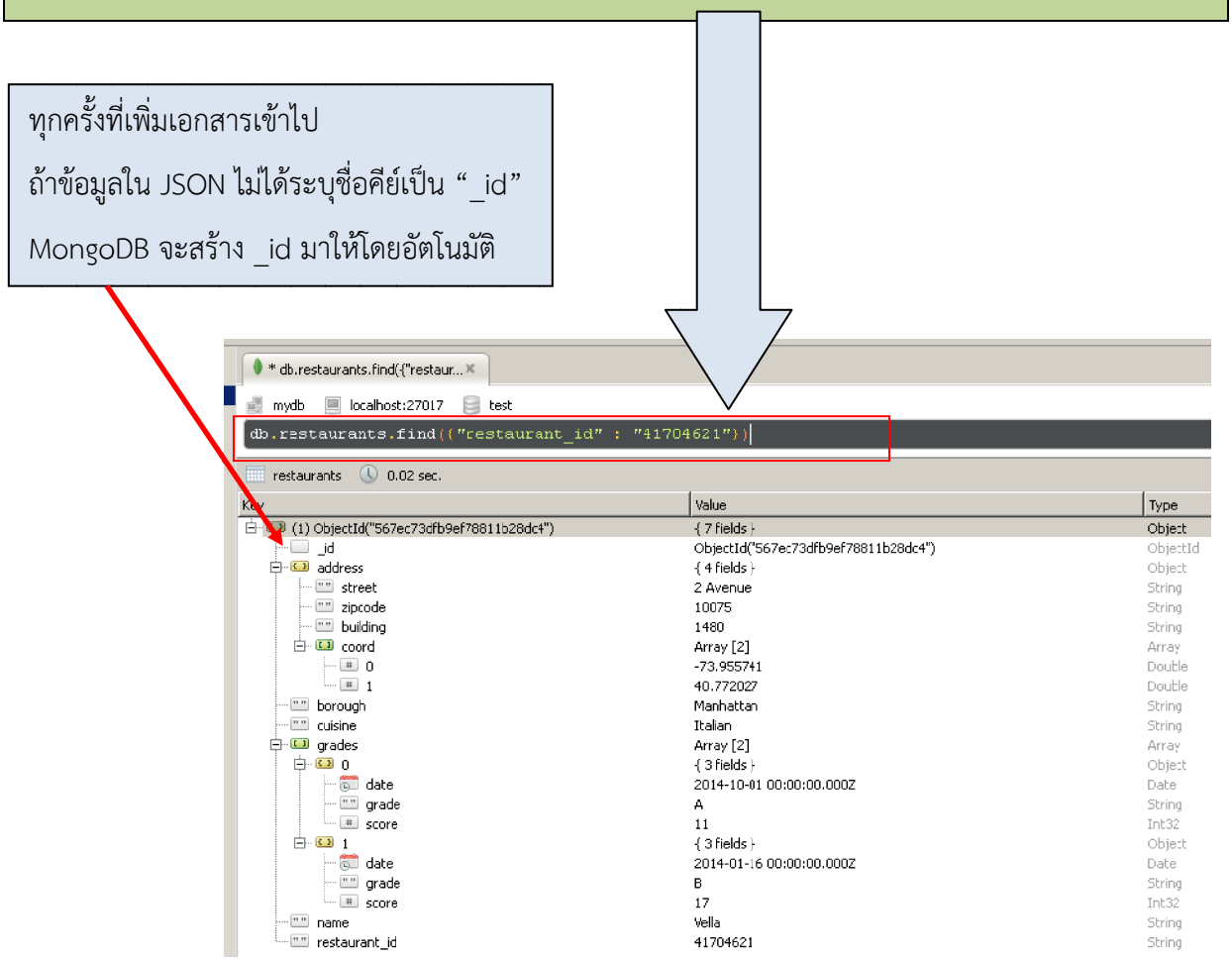
```
        insertDocument(db, function() {
                db.close();
        });
});
```

ในตัวอย่างนี้ เมธอด db.collection().insertOne() จะเพิ่มเอกสารลงไปใน restaurants ซึ่งมีผลการทำงานดังนี้

```
C:\testdb>node test.js
Inserted a document into the restaurants collection.
```

หลังจากนั้น ผมจะลองพิมพ์คำสั่งบน Robomongo เพื่อค้นหาเอกสารดังกล่าวที่เพิ่งเพิ่มเข้าไป ดังนี้

```
db.restaurants.find({"restaurant_id" : "41704621"})
```



*** สำหรับ _id จะใช้เป็น **primary key** ใน MongoDB และมีค่าเป็น ObjectId(...)

อัพเดตข้อมูล

สำหรับวิธีอัพเดตเอกสารใน restaurants ผมก็จะนำโค้ด test.js มาแก้ไขดังนี้

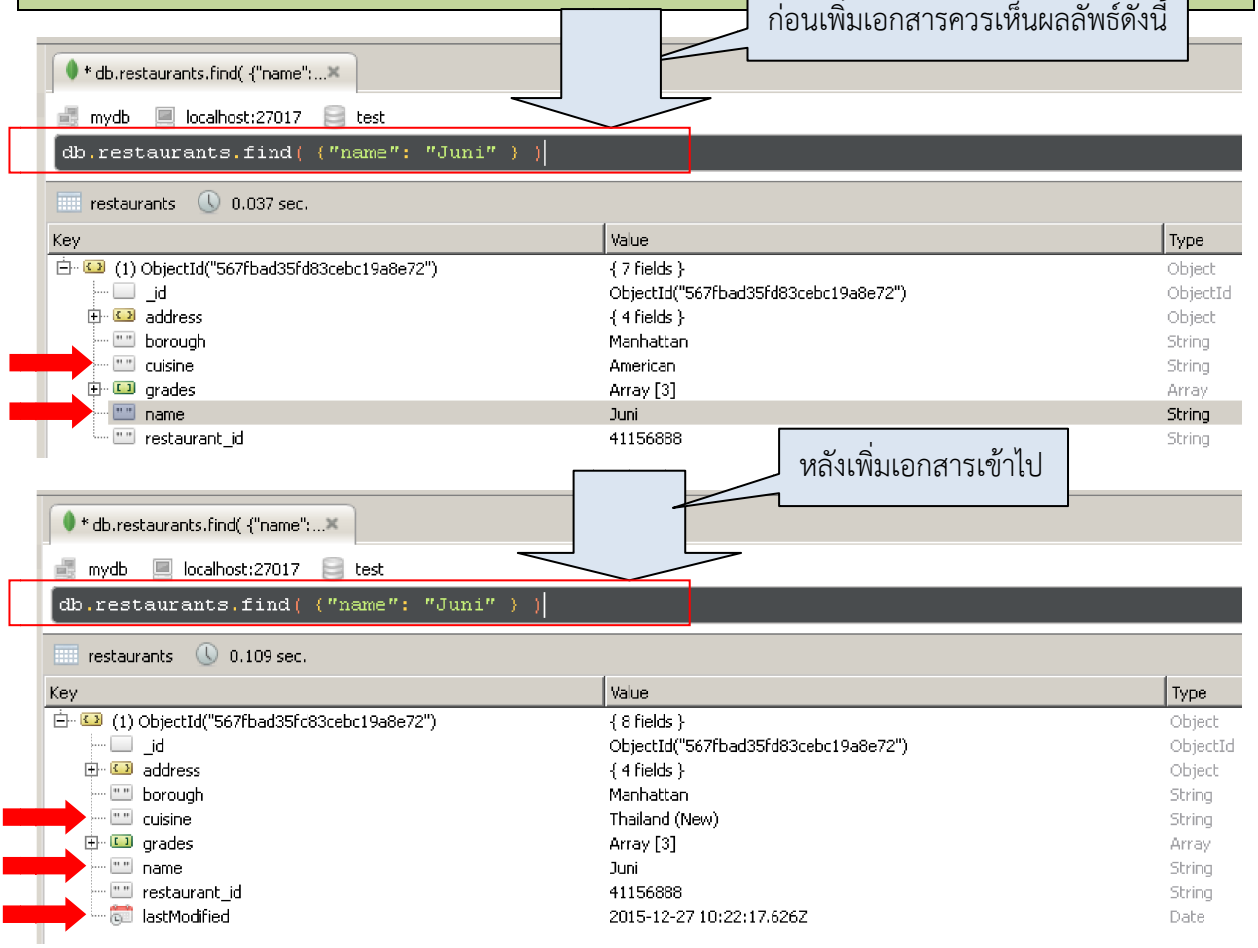
```
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
var url = 'mongodb://localhost:27017/test';

var updateRestaurants = function(db, callback) {
        db.collection('restaurants').updateOne(          // อัพเดตเอกสารตามเงื่อนไข
                { "name" : "Juni" },                     // เงื่อนไขได้แก่ คีย์ชื่อ "name" และมีค่าเป็น "Juni"
                {
                        $set: { "cuisine": "Thailand (New)" },
                        $currentDate: { "lastModified": true }
                 },
                function(err, results) {
                        console.log(results);
                        callback();
                 }); // สิ้นสุด updateOne()
};

// โค้ดชุดนี้คล้ายกับตัวอย่างที่แล้ว
MongoClient.connect(url, function(err, db) {
        assert.equal(null, err);

         updateRestaurants(db, function() {
                db.close();
          });
});
```

ในตัวอย่างนี้จะอัพเดตเอกสารตัวแรกที่เจอ เมื่อเงื่อนไขเป็น { "name" : "Juni" } ...ซึ่งจะมีรายละเอียดดังนี้

- $set: { "cuisine": "Thailand (New)" } → โอเปอเรเตอร์ $set จะบอกให้อัพเดตคีย์ที่ชื่อ "cuisine" ให้มีค่าเป็น "Thailand (New)"
- $currentDate: { "lastModified": true } → โอเปอเรเตอร์ $currentDate จะทำการเพิ่มฟิวด์ตัวสุดท้ายเป็นคีย์ที่ชื่อ "lastModified" และมีค่าเป็นวันที่ปัจจุบัน

เมื่อลองค้นเอกสารดังกล่าวใน Robomongo ด้วยคำสั่งในหน้าถัดไป

```
db.restaurants.find( {"name": "Juni" } )
```



สำหรับเมธอด db.collection().updateOne() จะอัพเดตแค่เอกสารตัวแรกที่เจอก่อน แต่ถ้าต้องการอัพเดตเอกสารทั้งหมดที่ค้นเจอ ...ก็ให้ใช้ updateMany() ดังตัวอย่าง

```
var updateRestaurants = function(db, callback) {
        db.collection('restaurants').updateMany(        // อัพเดตเอกสารทั้งหมด ตามเงื่อนไขที่ระบุ
            { "address.zipcode": "10016", cuisine: "Other" },
            {
                $set: { cuisine: "Category To Be Determined" },
                $currentDate: { "lastModified": true }
            },
            function(err, results) {
                console.log(results);
                callback();
            }); // สิ้นสุด updateMany()
};
```

ลบข้อมูล

ต่อมาจะเป็นวิธีลบข้อมูลใน restaurants โดยผมจะแก้ไขโค้ดใน “test.js” ดังนี้

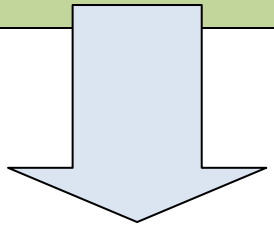
```
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
var url = 'mongodb://localhost:27017/test';

var removeRestaurants = function(db, callback) {
        db.collection('restaurants').deleteMany(  // ลบเอกสารทั้งหมดเลย ตามเงื่อนไขที่ระบุไว้
                { "borough": "Manhattan" },     // เงื่อนไขได้แก่ คีย์ชื่อ "borough" และมีค่าเป็น "Manhattan"
                function(err, results) {
                        console.log(results);
                        callback();
                }); // สิ้นสุด deleteMany()
};

// โค้ดชุดนี้คล้ายกับตัวอย่างที่แล้ว
MongoClient.connect(url, function(err, db) {
        assert.equal(null, err);

         removeRestaurants(db, function() {
                db.close();
          });
});
```

โค้ดชุดนี้ได้ใช้เมธอด db.collection().deleteMany() ลบข้อมูลทั้งหมด ตามเงื่อนไขที่ระบุไว้ก็คือ

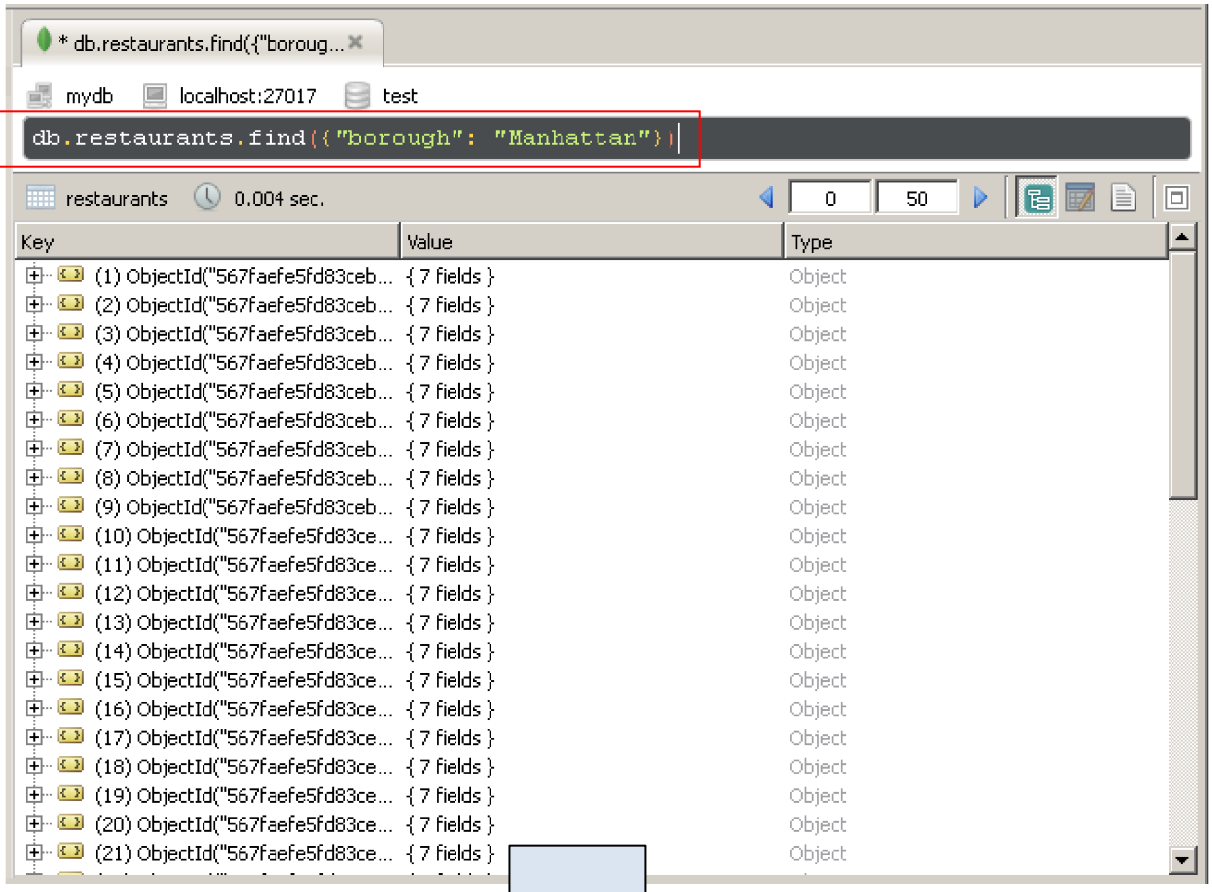
```
{ "borough": "Manhattan" }
```

เมื่อรันไฟล์ test.js ก็จะลบข้อมูลทิ้ง และผมจะลองใช้คำสั่งต่อไปนี้ ค้นหาเอกสารใน Robomongo ดูครับ

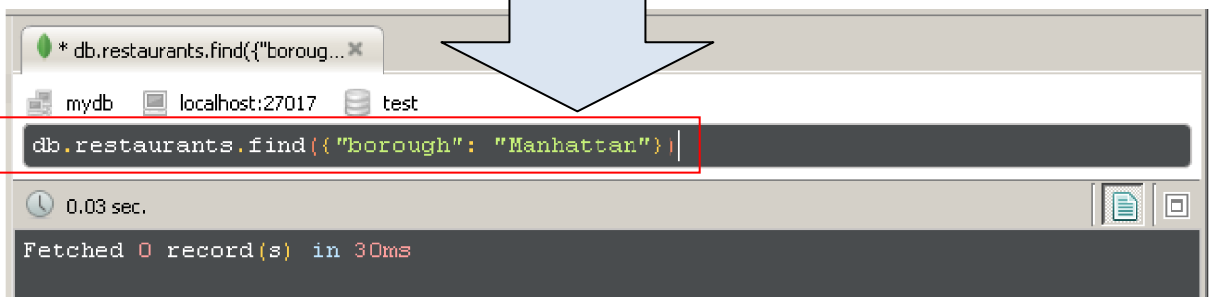
```
db.restaurants.find( {"borough": "Manhattan" } )
```



ดูหน้าถัดไป



หลังจากลบเอกสารไปแล้ว ก็จะค้นหาไม่เจอ



แต่ถ้าต้องการลบแค่เอกสารตัวเดียว ก็ให้ใช้เมธอด deleteOne() ดังตัวอย่าง

```
db.collection('restaurants').deleteOne( { "borough": "Queens" }, function(err, results) {
        /*
        ....โค้ด
        */
});
```

มันจะลบเอกสารแค่ตัวเดียวที่เจอครั้งแรก และตรงกับเงื่อนไข { "borough": "Queens" }

ถ้าคุณจะลบเอกสารทั้งหมด ก็ให้ระบุเงื่อนไขเป็น {} ดังตัวอย่าง

```
db.collection('restaurants').deleteMany( {}, function(err, results) {
        /*
        ....โค้ด
        */
});
```

ถ้าต้องการ drop ตัว restaurants ก็ให้ใช้คำสั่งดังนี้

```
db.collection('restaurants').drop( function(err, response) {
        /*
        ....โค้ด
        */
});
```

สำหรับการค้นหาข้อมูล เพิ่มข้อมูล อัพเดตข้อมูล ลบข้อมูล ที่อธิบายในบทนี้ มันเป็นแค่เบื้องต้นเท่านั้น ถ้าสนใจก็แนะนำให้ไปที่เว็บนี้โดยตรงเลย https://docs.mongodb.org/getting-started/node/

เพราะตัวอย่างทั้งหมดผมก็สรุปมาจากลิงค์นี้แหละ (ซอร์สโค้ดเดียวกัน) 

อ้างอิง

เข้าถึงล่าสุดเมื่อ 9 มกราคม 2559

[1] http://www.thaimongo.com/บทความ-mongodb/37-ทำความรู้จัก-nosql-คืออะไร.html

[2] http://meewebfree.com/site/general-web-technic/378-what-is-mongodb-database

[3] https://www.techtalkthai.com/introduce-sql-nosql-and-newsql-as-choices-of-database-technology/

[4] https://docs.mongodb.org/

[5] https://docs.mongodb.org/getting-started/node/