

การใช้งาน

MongoDB

เบื้องต้น

โดย แอดมินໂສ ໂອນ້ອຍອອກ

ประวัติการแก้ไข

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

ถ้าท่านดาวน์โหลดทิ้งไว้นาน และเพิ่งมาเปิดอ่าน ก็ขอรบกวนให้โหลดใหม่อีก
ครั้งที่

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

เพื่อคอมอัพเดตแก้ไข pdf ตัวใหม่เข้าไป หรือใครไปดาวน์โหลดมาจากที่อื่น
ก็อาจพลาดเวอร์ชั่นใหม่ล่าสุดได้ครับ

และรบกวนช่วย **กรอกแบบสอบถาม** ตามลิงค์ข้างบนด้วยนะครับ

แอดมินโซ โอน้อยอก

(จตุรพัชร์ พัฒนาวงศิริไไล)

9 ธันวาคม 2558

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

EBook เด่นนี้ส่วนใหญ่ถูกทำตามกฎหมาย ห้ามมิให้ผู้ใด นำไปเผยแพร่ต่อสาธารณะ เพื่อประโยชน์ในการศึกษา หรืออื่นๆ โดย
ไม่ได้รับความยินยอมเป็นลายลักษณ์อักษรจากผู้เขียน

เกริ่นนำ

เริ่มแรกเดิมที่ผมได้เขียนหนังสือ “เสียดายไม่ได้อ่านภาษาสคริปต์ ฝั่งเชิร์ฟเวอร์ (Node.js ฉบับย่อ)” เล่มที่ 2 ก็
จะจะเขียนเรื่องการใช้ Node.js ติดต่อฐานข้อมูล MongoDB

แต่เมื่อเขียนไปเขียนมา ก็กลัวคนอ่านไม่เข้าใจว่า ...MongoDB คืออะไร ผมเลยแยกเขียนออกมา จนกลายมา
เป็นเล่มนี้นั้นเอง ซึ่งจุดประสงค์หลัก ก็เพื่อแนะนำให้เห็นภาพว่า ถ้าใช้ Node.js ติดต่อกับ MongoDB ต้องทำ
อย่างไรมากกว่า (เล่นนี้จึงเหมือนเป็นชีรีส์ภาคต่อมากกว่า)

สุดท้ายนี้หากเนื้อหามีอะไรผิดพลาดไป เช่น ให้ข้อมูลผิด สะกดอะไรมิดไป มุ้งแป็กบ้าง ข้าบ้าง หรืออ่านแล้ว
มึนงงไป 7 วัน เป็นต้น ผมก็ขอภัยมา ณ โอกาสนี้ด้วย และถ้าคุณเข้าใจ ไม่เข้าใจยังไง ก็สามารถชี้แนะผมได้
ตลอดเวลา

...อีกทั้งผมก็ตั้งใจจะมั่นอัพเดตเนื้อหา 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) หรืออัปเดตตาราง เป็นต้น

คำถาม ถ้าไม่ใช่ตาราง และ MongoDB มันเก็บข้อมูลเป็นอะไร?

คำตอบ เก็บข้อมูลในรูปแบบ JSON

ตัวอย่าง JSON

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

***หมายเหตุ คำว่า **NoSQL** มันไม่ใช่มาตรฐานอะไรมากเลย เพียงแค่บอกว่าไม่ใช้ภาษา SQL คุยกับฐานข้อมูล ก็เท่านั้นเอง ด้วยเหตุนี้ฐานข้อมูล NoSQL ประเภทอื่น ๆ จึงอาจเก็บข้อมูลแบบอื่นได้ เช่นกัน ([ไม่ใช่ JSON](#))

สำหรับฐานข้อมูลแบบ NoSQL อาจแบ่งได้ดังนี้ [1]

ประเภท	ตัวอย่างชื่อรากฐานข้อมูล
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
- ค้นหาจากข้อมูลที่มีปริมาณมากหมายมหาศาล ได้อย่างรวดเร็ว (รองรับการทำ Full Index)
- เก็บข้อมูลได้ทั้งกว้าง และลึก (ไม่ใช่แบบตารางที่มีมิติเดียว ซึ่งเก็บข้อมูลได้แค่ 1 แถว)
- เรียกข้อมูลมาแสดงผลได้ง่าย (ไม่ต้องใช้ภาษา SQL ทำการ Join ตารางโน่นนั่นนั้น)
- แก้ไขข้อมูลได้รวดเร็ว
- สำรองข้อมูลได้ง่าย ไม่ต้องตั้งค่าอะไรเลยอะ
- เขียนชุดคำสั่งเป็นスคริปต์ แล้วสั่งรันทีเดียวได้
- เก็บข้อมูลด้วยระบบ GridFS (เก็บข้อมูลเป็นก้อน ๆ บนฮาร์ดดิสก์ ซึ่งสามารถรองรับการเพิ่มขึ้น หรือลดลงของปริมาณข้อมูล)
- มีบริการสอบบานาและดูแลเป็นพิเศษ (ເສີເຈີນ)

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



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

สรรสิ่งทุกอย่างในใต้หล้า ย่อมมีให้รีบส่องด้าน มีทั้งข้อดีและข้อเสีย ไม่วันแม้แต่ NoSQL ก็ย่อมมีข้อเสีย เช่นเดียวกัน ดังต่อไปนี้

- ถ้าจะเน้นความถูกต้อง จะผิดพลาดไม่ได้เลย ...NoSQL จะไม่เหมาะสม เพราะมีโอกาสที่ข้อมูลจะเกิดการ loss (สูญหาย) ได้มาก (เน้นเร็ว ไม่เน้นความถูกต้อง)
- เนื่องจาก NoSQL ไม่มีมาตรฐานกลาง ดังนั้นมีเปลี่ยนไปใช้ฐานข้อมูลค่ายใหม่ ก็ต้องเสียเวลาศึกษาใหม่
- ผู้เชี่ยวชาญด้าน NoSQL สำหรับองค์กรในไทย ยังมีไม่มาก ซึ่งสวนทางกับเทรนด์ BigData ที่กำลังมาแรงในปัจจุบัน (นับตั้งแต่ผู้เชี่ยวชาญแต่งหนังสือ)



ตามลัทธิเต๋า จะมีหิน-หยาง

หมายถึงสรรพสิ่งต้องคู่กัน

มีพระอาทิตย์ ต้องมีพระจันทร์

มีชาย ต้องมีหญิง

มีกลางวัน ต้องมีกลางคืน

ตั้งนั้น ...มี SQL ก็ต้องมี NoSQL

เสริมนิพนธ์อย

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 อาจไม่ละเอียดเท่าไรนัก

“MongoDB เป็นฐานข้อมูลโอลิเพ่นชอร์สที่เก็บเอกสาร (**document database**) ที่มาพร้อมกับประสิทธิภาพที่สูง รองรับการใช้งานตลอดเวลาได้สูง ปรับตัวขยายขนาดได้อย่างอัตโนมัติ ซึ่งตัว MongoDB ได้กำจัดแนวคิดเรื่อง Object Relational Mapping (ORM) จึงทำให้มันมีความสะดวกในการพัฒนาซอฟต์แวร์”

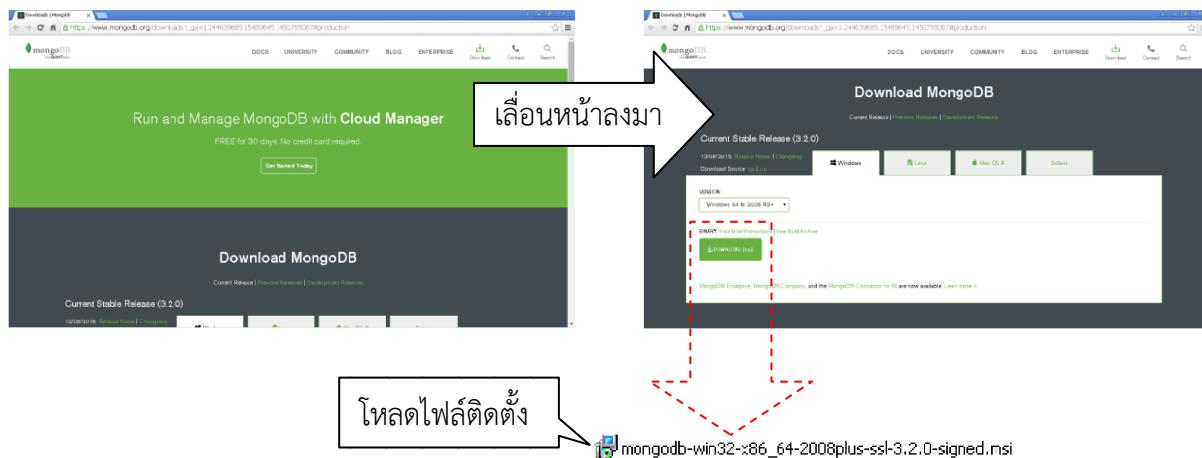
....ยังไม่ต้องซีเรียสกับคำนิยามมากครับ แค่ยกมาให้อ่านเฉย ๆ อีกทั้งผมก็เกลามาจาก Google translate ซึ่งมันแปลได้แค่นี้แหละ อิ ๆ ๆ

วิธีติดตั้ง MongoDB

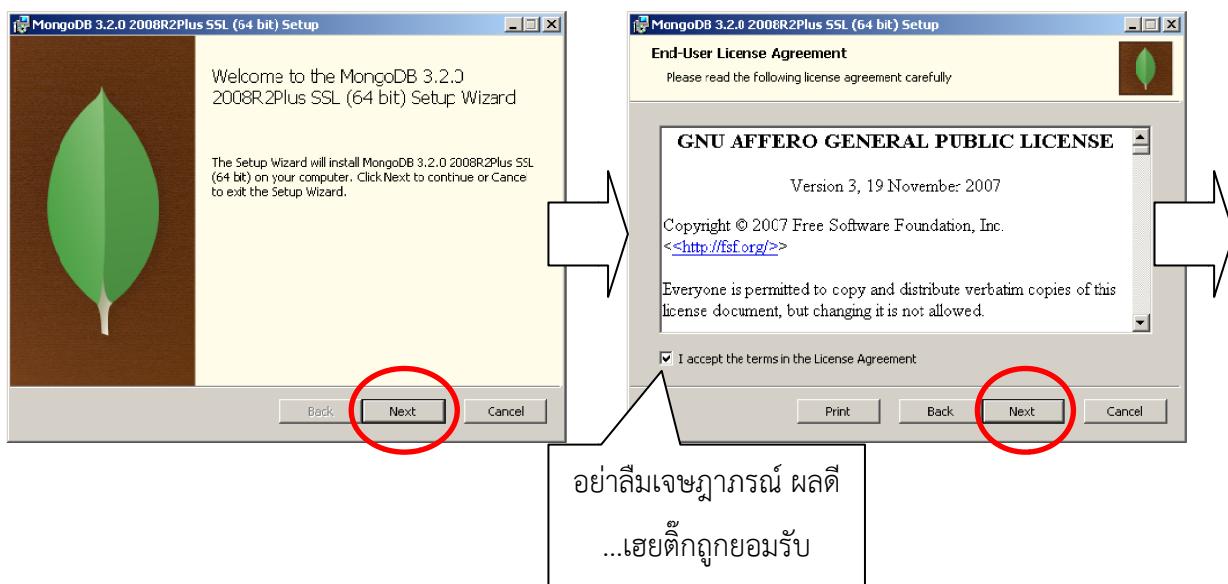
1) ให้คุณไปที่ลิงค์ข้างล่าง เพื่อดาวน์โหลดไฟล์ติดตั้ง

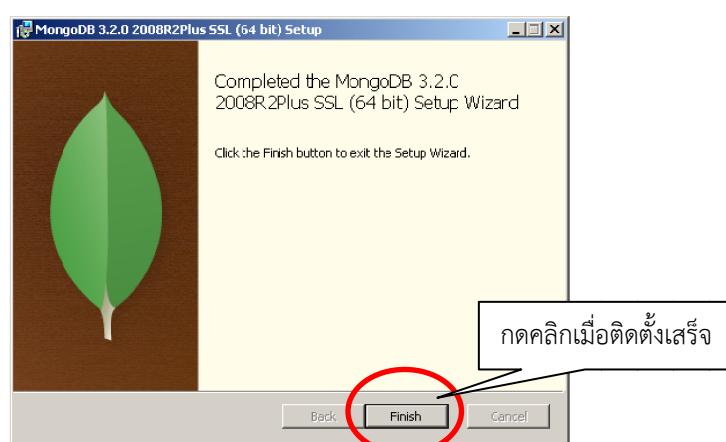
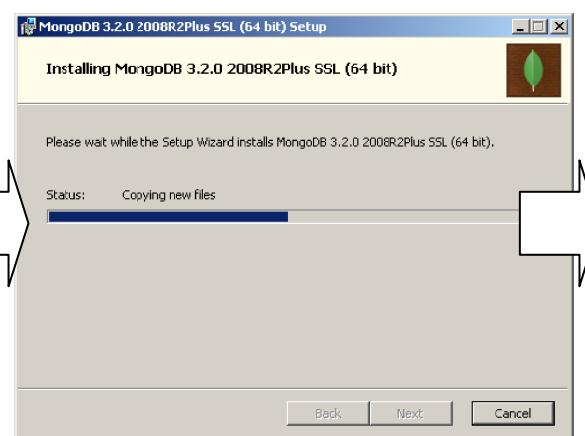
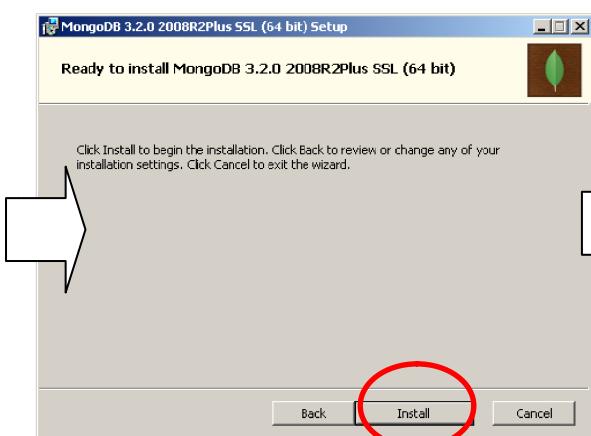
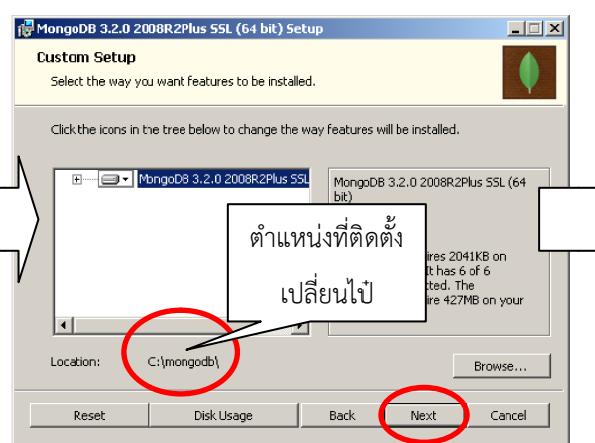
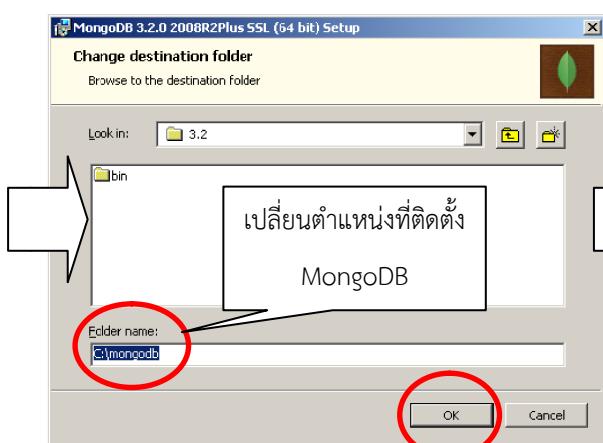
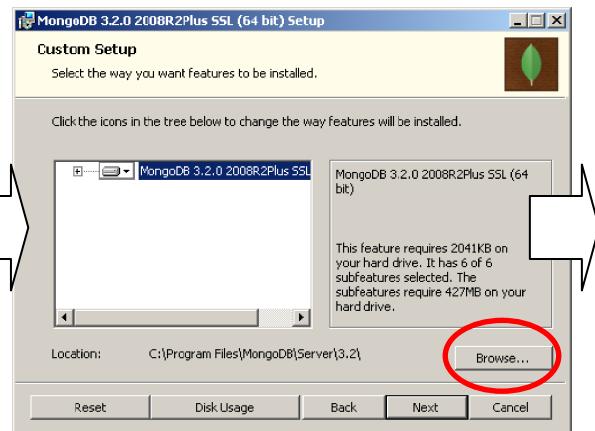
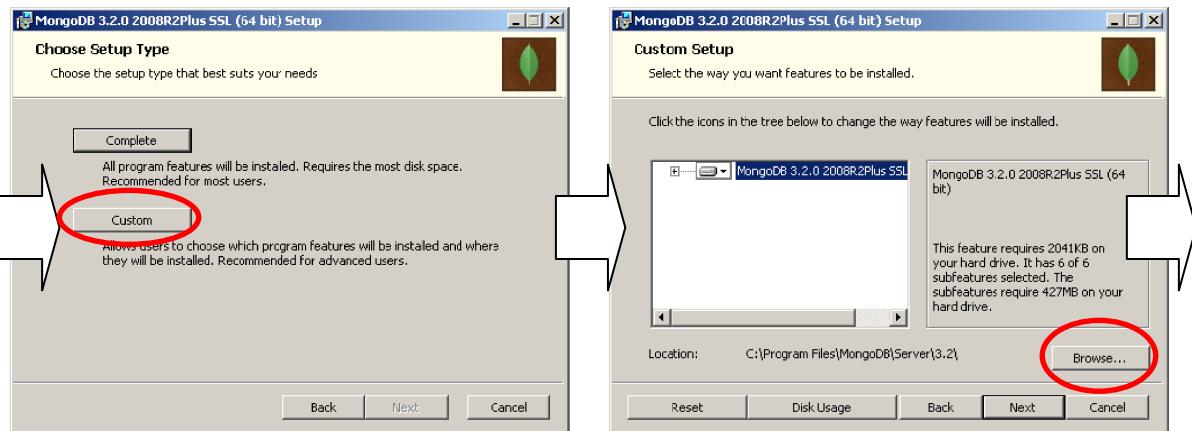
https://www.mongodb.org/downloads?_ga=1.244639685.15489645.1450755087#production

2) คุณจะเห็นหน้าเว็บ และให้เลื่อนเพจลงมาเรื่อย ๆ จนเจอปุ่มให้กดดาวน์โหลดไฟล์ติดตั้ง ซึ่งในที่นี่ผมจะเลือกลงบน **winдов斯ครับ** (เครื่องผู้เขียนเป็น 64 บิต)



3) เมื่อดาวน์โหลดเสร็จแล้ว ก็ให้ดับเบลคลิกที่ไฟล์ติดตั้ง เมื่อ昆ลงโปรแกรมปกติ ซึ่งรูปข้างล่างจะเป็นแค่ไอเดียเฉย ๆ ... เพราะถ้ามีเวอร์ชันใหม่อกมา ก็ให้ติดตั้งทำงานองเดียวกัน (หวังว่าจะครับ)



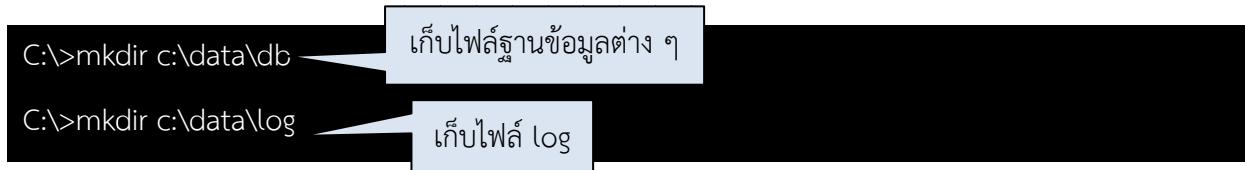


หมายเหตุ ★

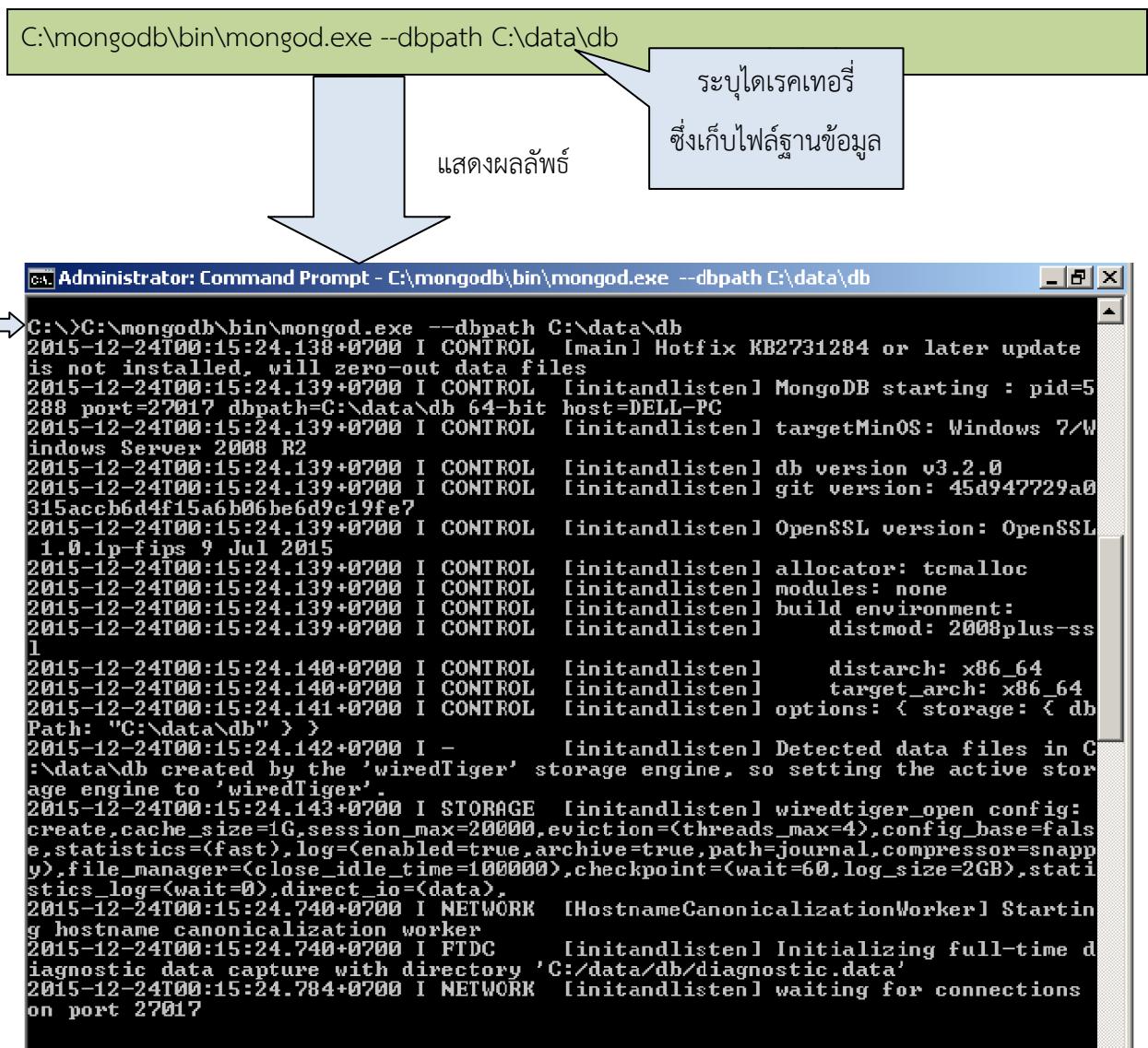
ถ้าติดตั้งบน Windows Server 2008 R2 หรือ Windows 7 อาจต้องติดตั้ง hotfix ก่อน ตามลิงค์ ข้างล่างก่อน (กรณีมีปัญหาใช้งาน MongoDB ไม่ได้)

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

4) เตรียมสร้างโฟลเดอร์ใหม่ สำหรับเก็บไฟล์ฐานข้อมูลต่าง ๆ และไฟล์ log ด้วยการพิมพ์คำสั่งข้างล่าง



5) ให้ลองรัน MongoDB ตามคำสั่งข้างล่าง



ในภาพนี้ MongoDB กำลังทำงานอยู่ (MongoDB ของผู้เขียนจะถูกติดตั้งอยู่ที่ [C:\mongodb](#)) ซึ่งหน้าจอนี้
ห้ามปิดเด็ดขาด ปล่อยค้างไว้อย่างนี้แหละ

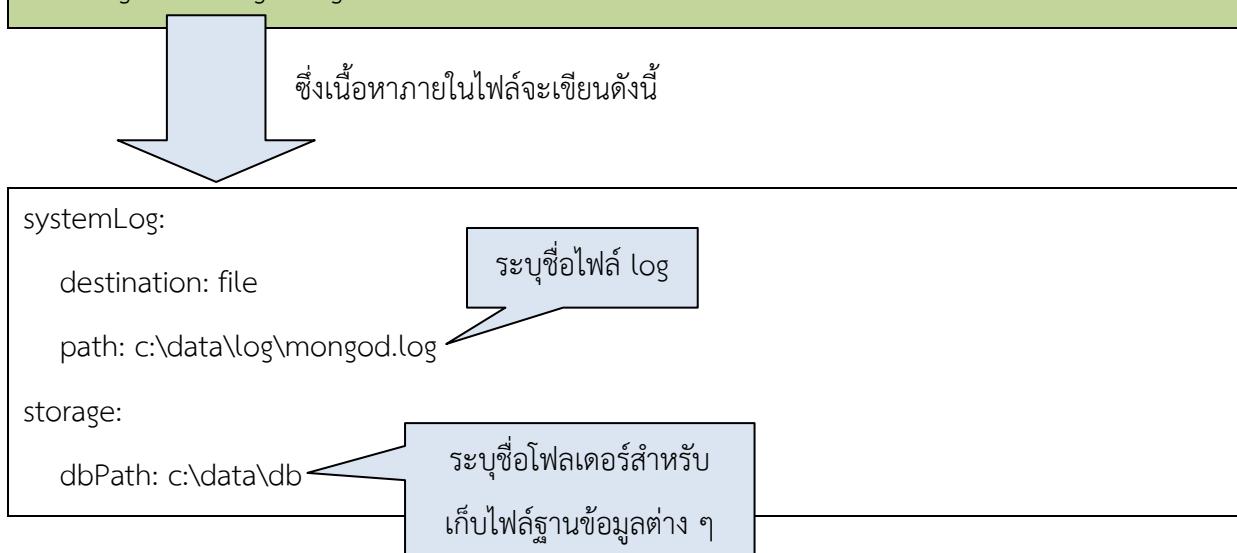
...และถ้าคุณแอบไปเปิดโฟลเดอร์ [C:\data\db](#) ก็จะเห็นว่ามีไฟล์ต่าง ๆ ของ MongoDB ได้ถูกสร้างเอาไว้ให้

วิธีตั้งค่า MongoDB ให้เป็นเซอร์วิส

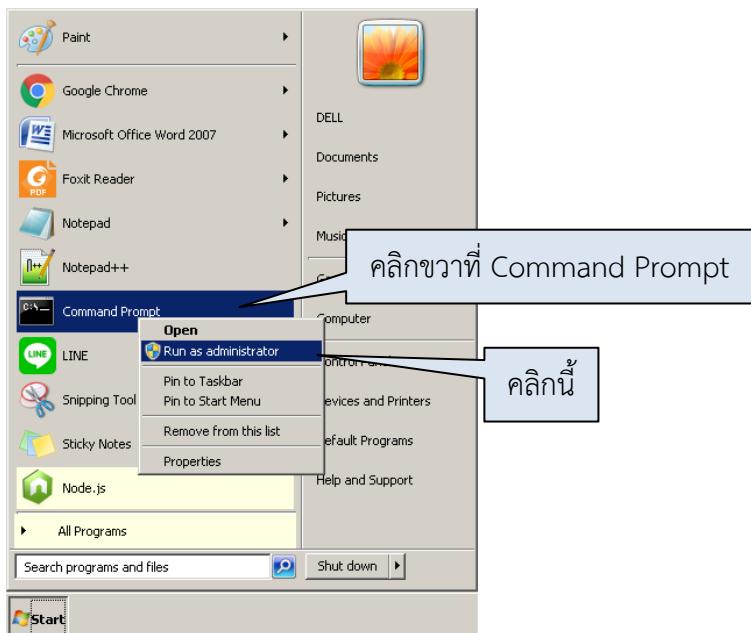
ในบทก่อนหน้านี้ จะเห็นว่าเราตั้ง MongoDB ผ่านทางคอมมานไลน์ มันจะยุ่งยากเกินไป แต่เราสามารถเปลี่ยนให้มาตั้งอยู่เบื้องหลังได้ ด้วยการทำเป็นเซอร์วิส (Service) ในวินโดว์ (เมื่อเปิดคอมขึ้นมา มันก็จะรันให้ทันที) ซึ่งจะมีวิธีการตั้งค่าตามขั้นตอนต่อไปนี้

- 1) สร้างไฟล์ “mongod.cfg” ที่ไฟล์เดอร์ดังต่อไปนี้

C:\mongodb\mongod.cfg

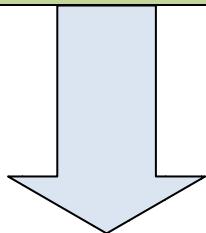


สำหรับขั้นตอนต่อจากนี้ไป ควรเปิดคอมมานไลน์บนวินโดว์ โดยได้รับสิทธิเป็น “Administrative” ดังภาพ



2) ติดตั้ง MongoDB ให้กลายมาเป็นเซอร์วิส ด้วยคำสั่งบนคอมมานไลน์ ดังนี้

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



ไฟล์ที่สร้างในข้อ 1

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

3) ถ้าจะสั่งให้เซอร์วิส MongoDB ทำงาน ก็ให้ใช้คำสั่งดังนี้

```
net start MongoDB
```

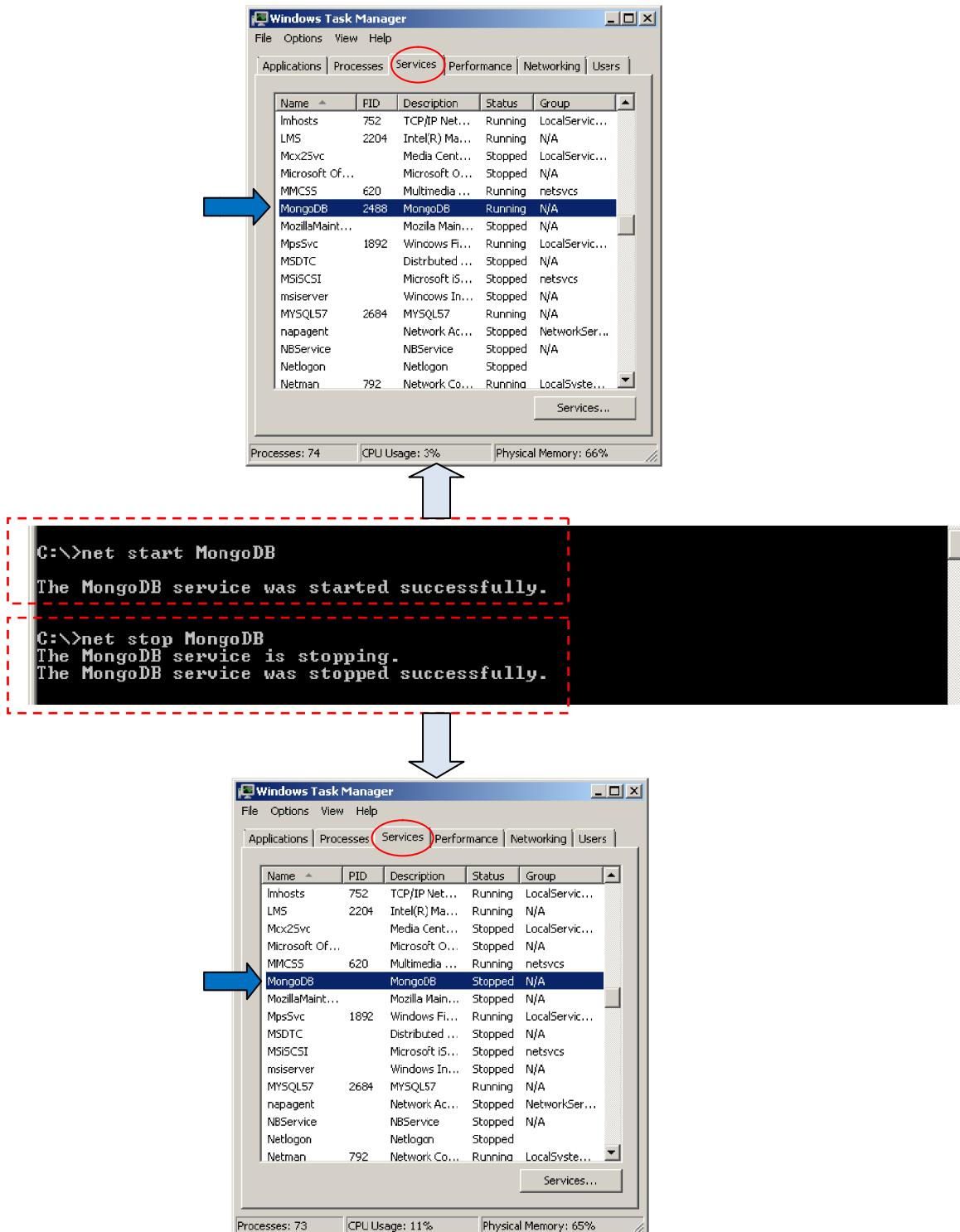
4) ถ้าจะสั่งให้เซอร์วิส MongoDB หยุดทำงาน ก็ให้ใช้คำสั่งดังนี้

```
net stop MongoDB
```

5) ถ้าจะลบเซอร์วิส MongoDB ก็ให้ใช้คำสั่งดังนี้

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

สำหรับตัวอย่างการทำงานของคำสั่งในข้อ 3, 4 และ 5 ก็ให้ดูรูปในหน้าถัดไปได้เลยครับ



เมื่อจะลบเซอร์วิส ก็ให้พิมพ์คำสั่งดังนี้

```
C:\>"C:\mongodb\bin\mongod.exe" --remove
2015-12-24T00:12:30.736+0700 I CONTROL [main] Hotfix KB2731284 or later update
is not installed, will zero-out data files
2015-12-24T00:12:30.736+0700 I CONTROL [main] Trying to remove Windows service
'MongoDB'
2015-12-24T00:12:30.738+0700 I CONTROL [main] Service 'MongoDB' removed
```

สำหรับ OS อื่น ๆ นอกจากวินโดวส์ ★

ถ้าจะติดตั้ง และรัน MongoDB บน Linux ก็ให้ไปอ่านเอกสารได้ที่

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

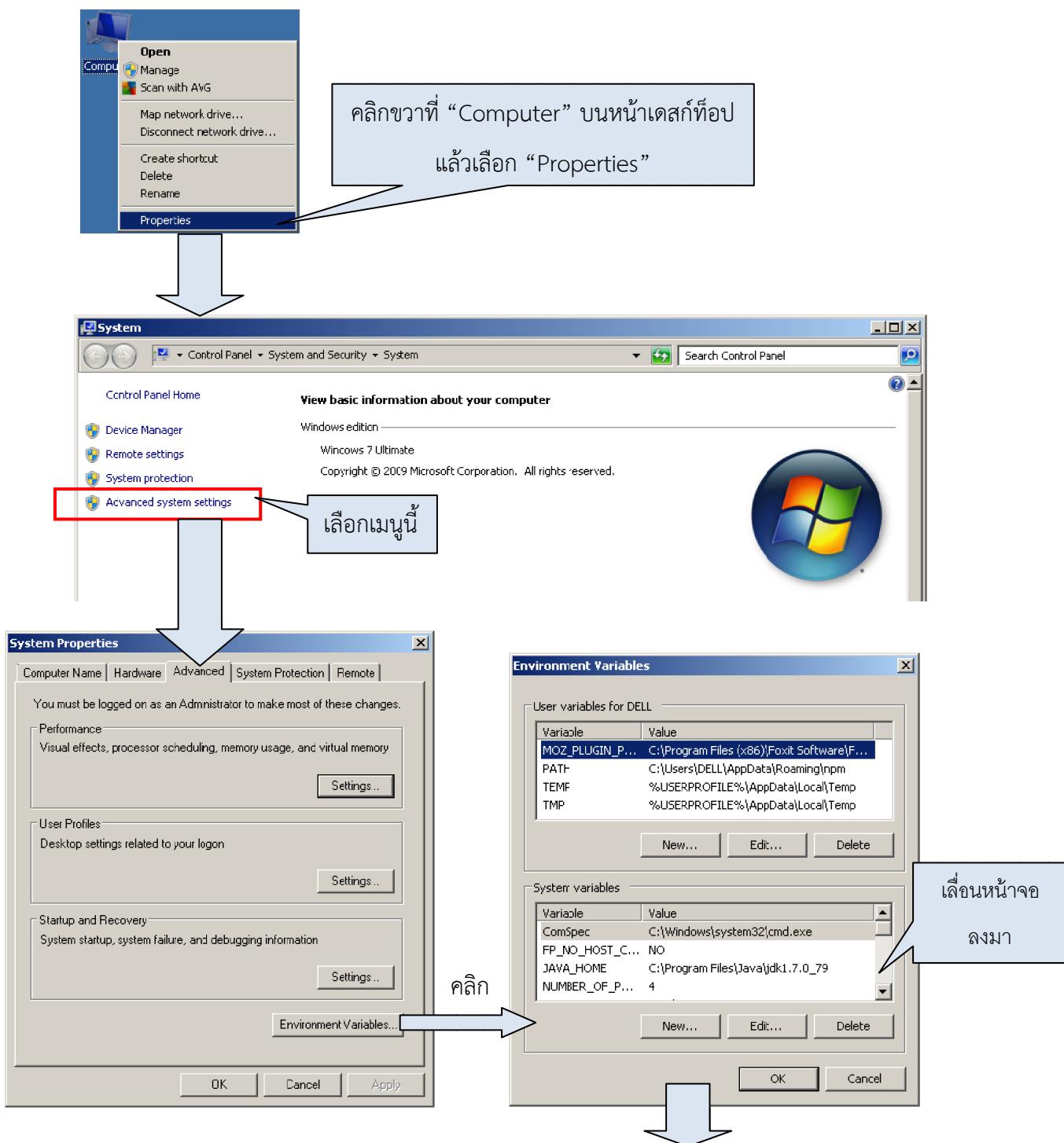
หรือถ้าติดตั้ง และรัน MongoDB บน OS X ก็ให้ไปอ่านเอกสารได้ที่

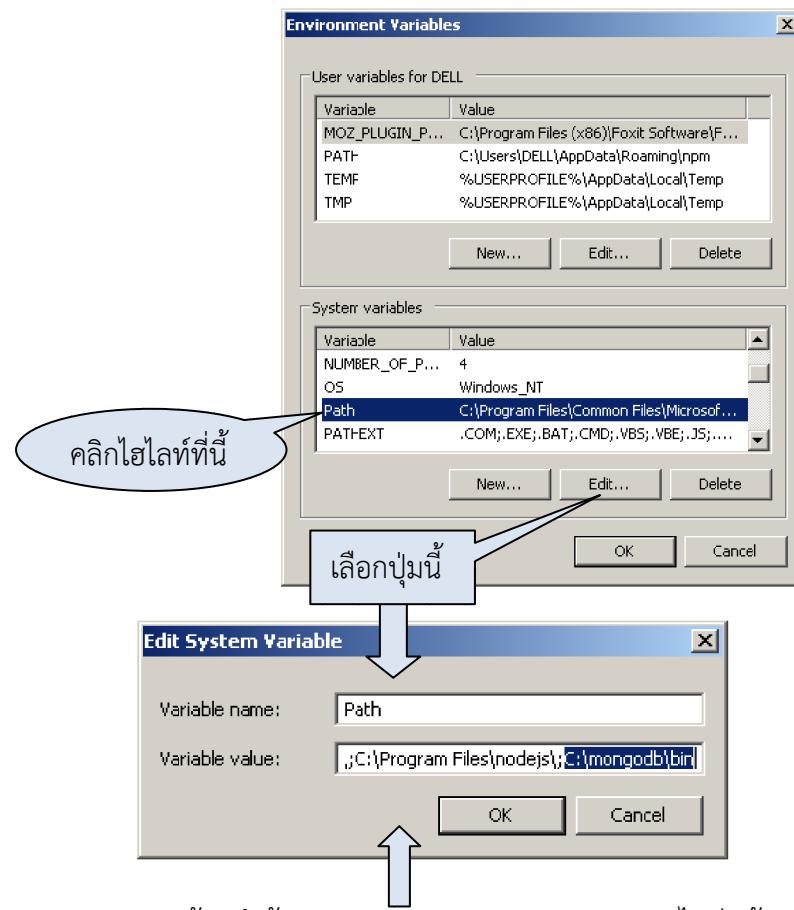
<https://docs.mongodb.org/v3.0/tutorial/install-mongodb-on-os-x/>

ເຊື່ອພາບນວນໂວດສີ

ເນື່ອງຈາກຄໍາສັ່ງຕ່າງ ຈະ MongoDB ບນຄອມມານໄລ້ນໍ້າຂອງຜູ້ເຂົ້າໃນ ຈະເກີບອູ້ທີ່ C:\mongodb\bin ຜົ່ງທີ່ໃຫ້
ຢູ່ຢາກ ເພົ່າຕ້ອງເຮັດວຽກຄໍາສັ່ງຜ່ານພາຣ (pat) ເຕັມ ເຊັ່ນ "C:\mongodb\bin\mongod.exe" ເປັນດັນ

...ແຕ່ເຮົາໄມ່ຕ້ອງຮະບຸພາຣເຕັມກີ່ໄດ້ ດ້ວຍການເຂົ້າໄປແກ້ Environment variable (ຕັ້ງແປງຂອງຮະບບ) ດັ່ງການ





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

...เราสามารถทดสอบได้ว่า ตั้งพาราเซอร์เรียบร้อยแล้วหรือไม่ ด้วยการพิมพ์ “mongod -version” บนคอมมานาไลน์ ที่ไดเรกทอรีไหนก็ได้ ซึ่งควรจะเห็นเลขเวอร์ชันตามภาพ (เลขเวอร์ชันจะเปลี่ยนไปตามไฟล์ติดตั้ง ที่เราดาวน์โหลดมา)

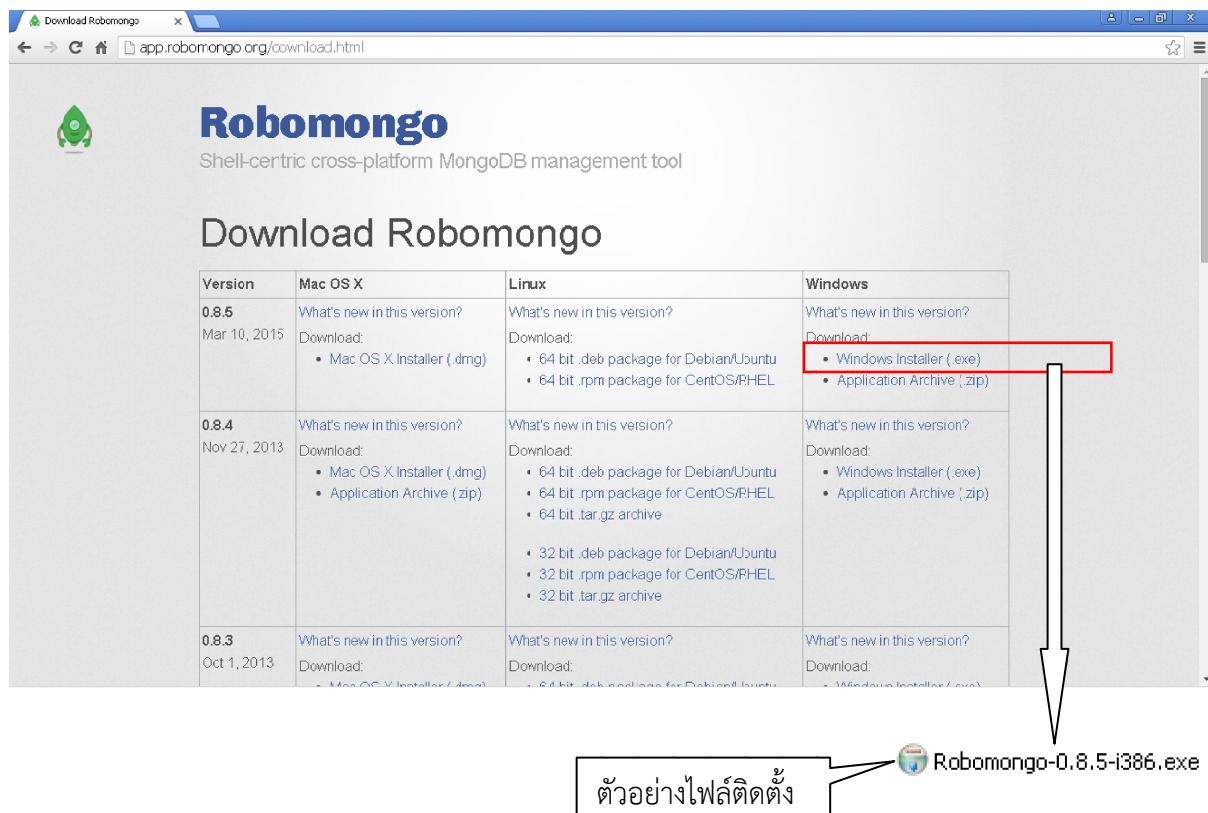
Administrator: Command Prompt

```
C:\Users\DELL>mongod -version
db version v3.2.0
git version: 45d947729a0315acccb6d4f15a6b06be6d9c19fe7
OpenSSL version: OpenSSL 1.0.1p-fips 9 Jul 2015
allocator: tcmalloc
modules: none
build environment:
    distmod: 2008plus-ssl
    distarch: x86_64
    target_arch: x86_64
```

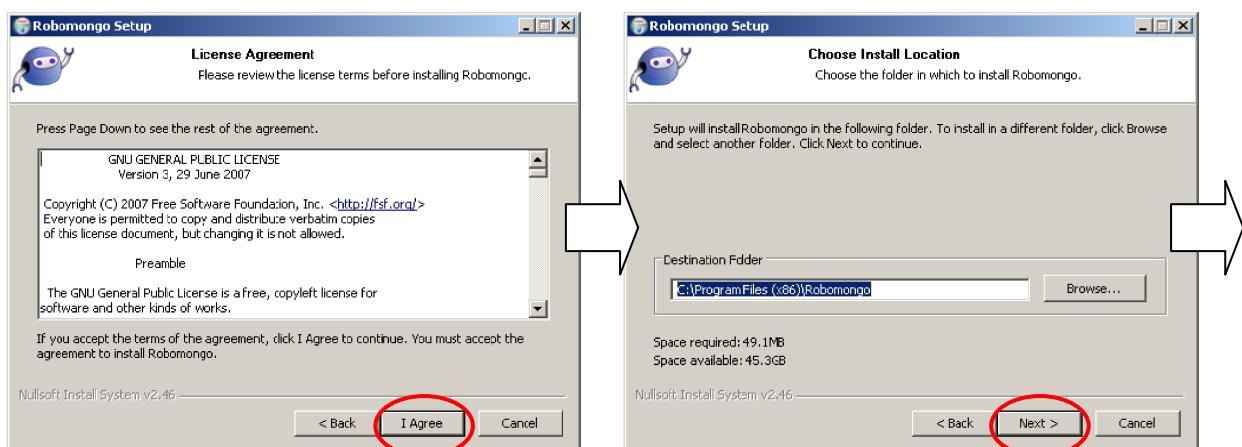
ติดตั้ง Robomongo

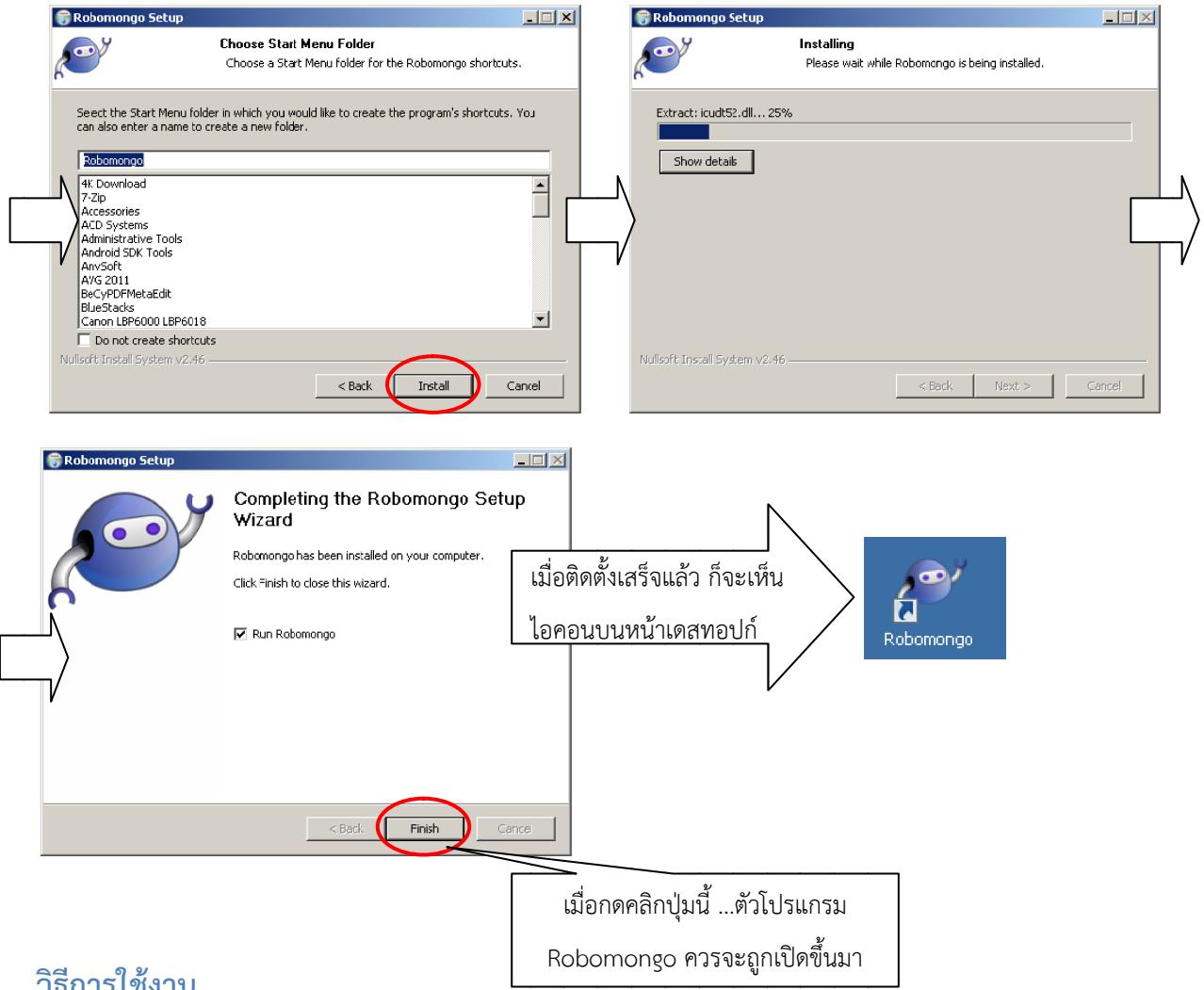
เนื่องจากการใช้ MongoDB ผ่านทางคอมมานไลน์ตรง ๆ บางคราอาจไม่ถนัด จึงอาจติดตั้งเครื่องมือช่วย ซึ่งในหนังสือนี้จะใช้ Robomongo โดยวิธีติดตั้งดังขั้นตอนต่อไปนี้

- ให้ไปที่ลิงค์ <http://app.robomongo.org/download.html> และดาวน์โหลดไฟล์ติดตั้ง ซึ่งผมจะเลือกเวอร์ชันที่ติดตั้งลงบนวินโดวส์



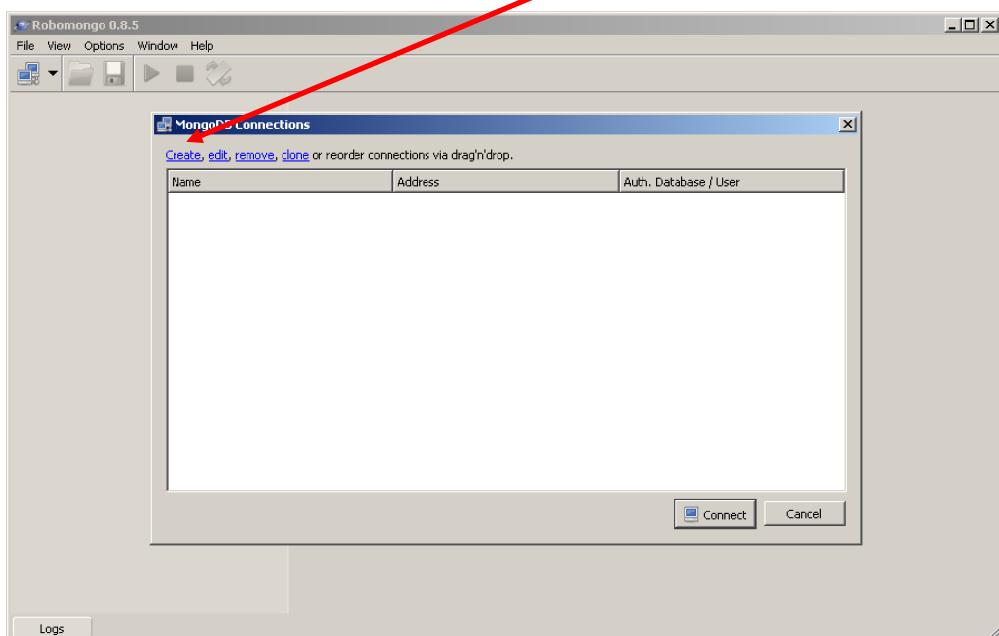
- ตั้งเบลคลิกไฟล์ติดตั้ง เมื่อ安装โปรแกรมปกติธรรมง่าย ๆ ไม่ยากครับ



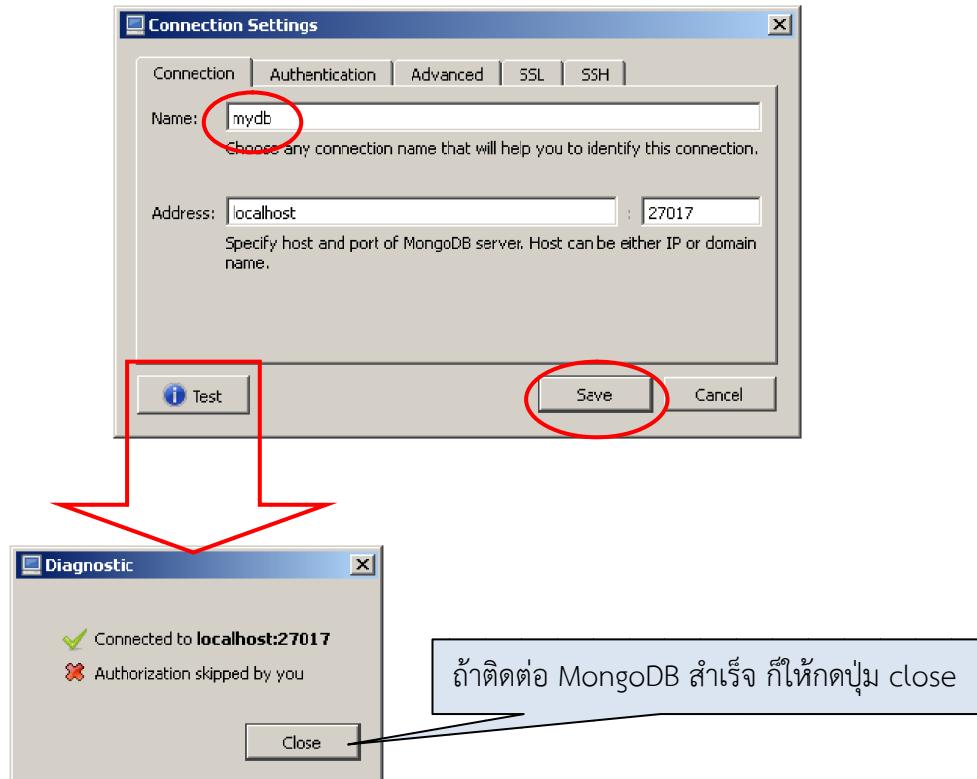


วิธีการใช้งาน

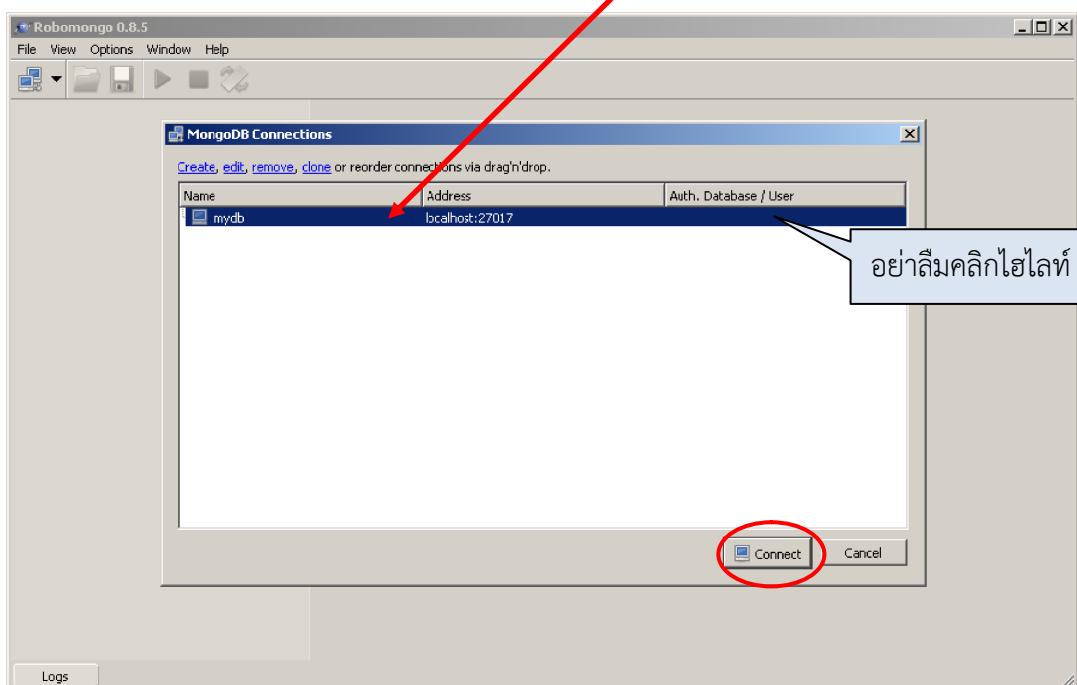
- ให้เปิดโปรแกรม Robomongo ขึ้นมา แล้วก็คลิกเลือก “Create” (ปุ่มเล็กนิดเดียว)



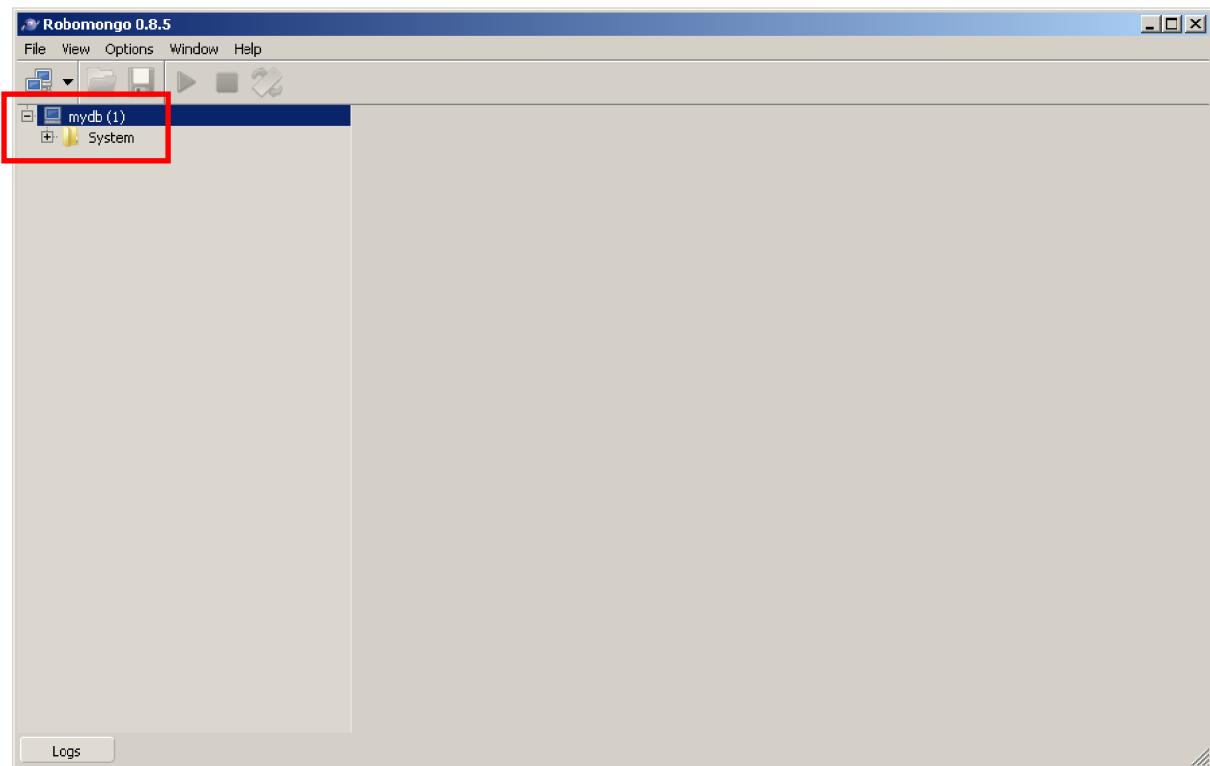
- 2) ตั้งชื่อ “Connection” ซึ่งในตัวอย่างนี้ตั้งชื่อเป็น “mydb” หลังจากนั้นก็ให้กด “Save” หรืออาจกด “Test” ก่อน เพื่อทดสอบว่าติดต่อ กับ MongoDB ได้หรือไม่ (อย่าลืมเปิด MongoDB ด้วยละ)



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



4) ต่อมาผมก็จะเห็นข้อมูลการเชื่อมต่อ ดังรูปข้างล่าง



การนำเข้าข้อมูล

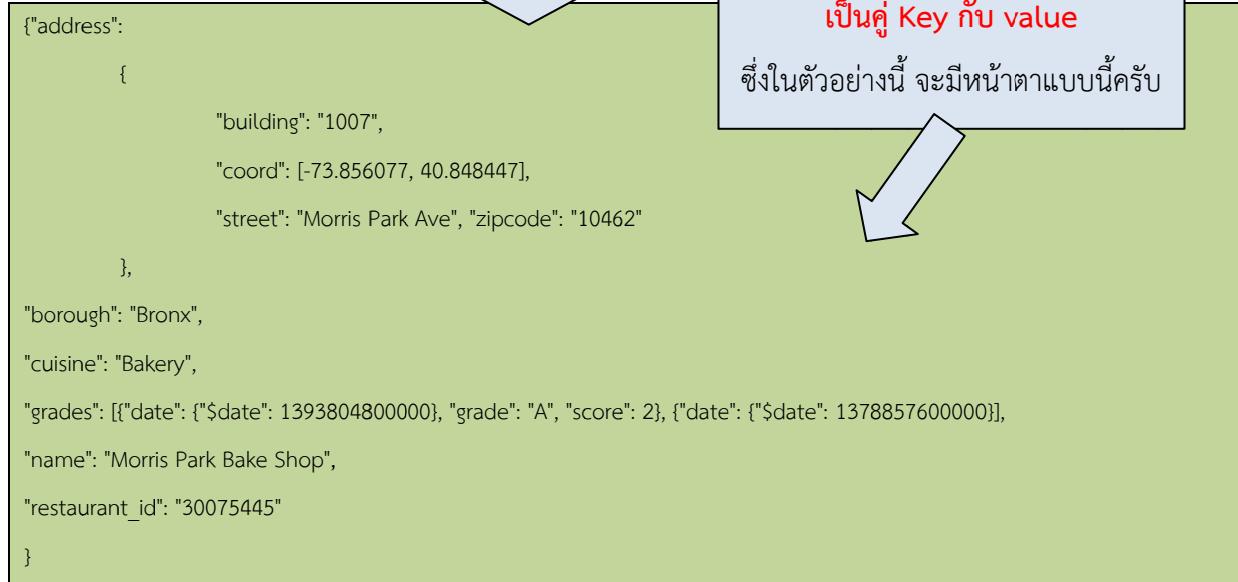
ในบทนี้ผมจะทำเวิร์คช้อปอย่างง่าย ...ด้วยการนำเข้าข้อมูล (Import) ลง MongoDB ดังต่อไปนี้

- 1) ผมจะไปที่ลิงค์ข้างล่าง ซึ่งเป็นไฟล์ข้อมูลตัวอย่าง 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”

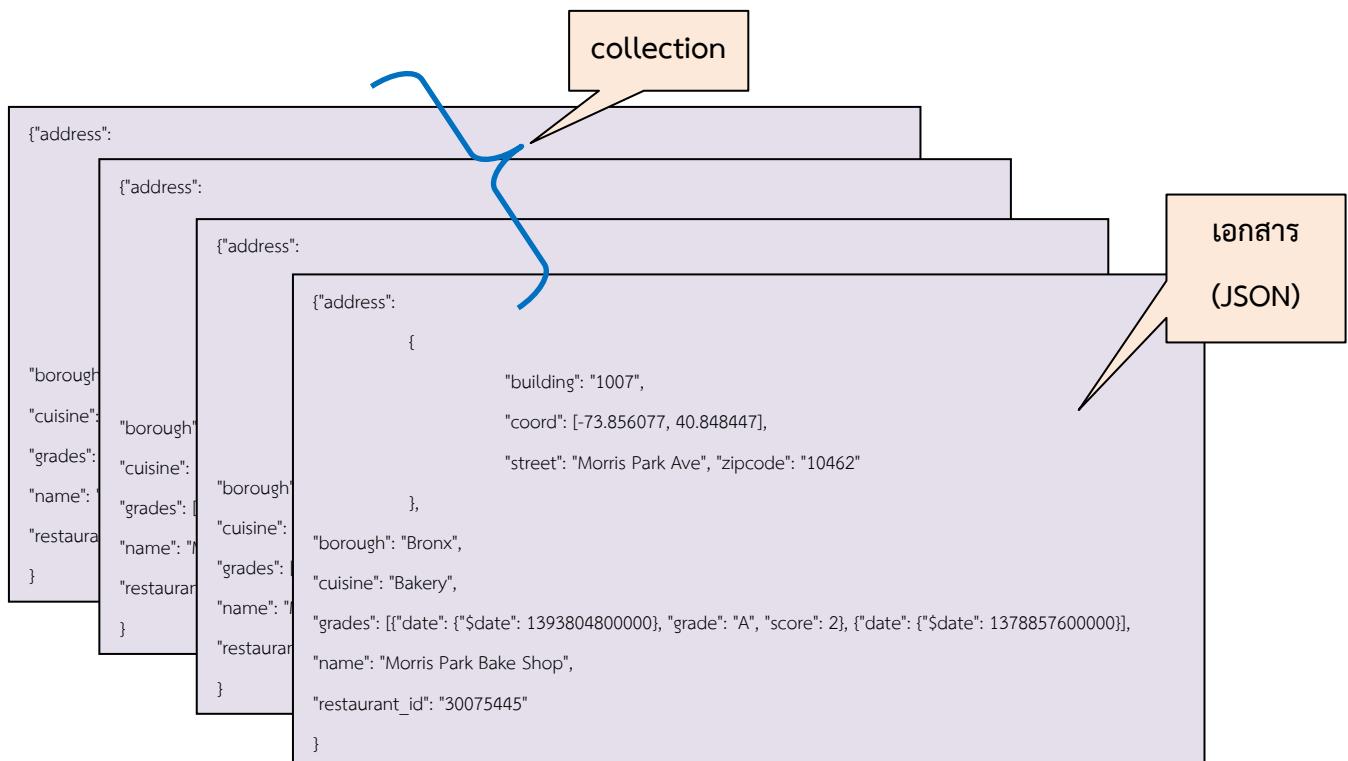
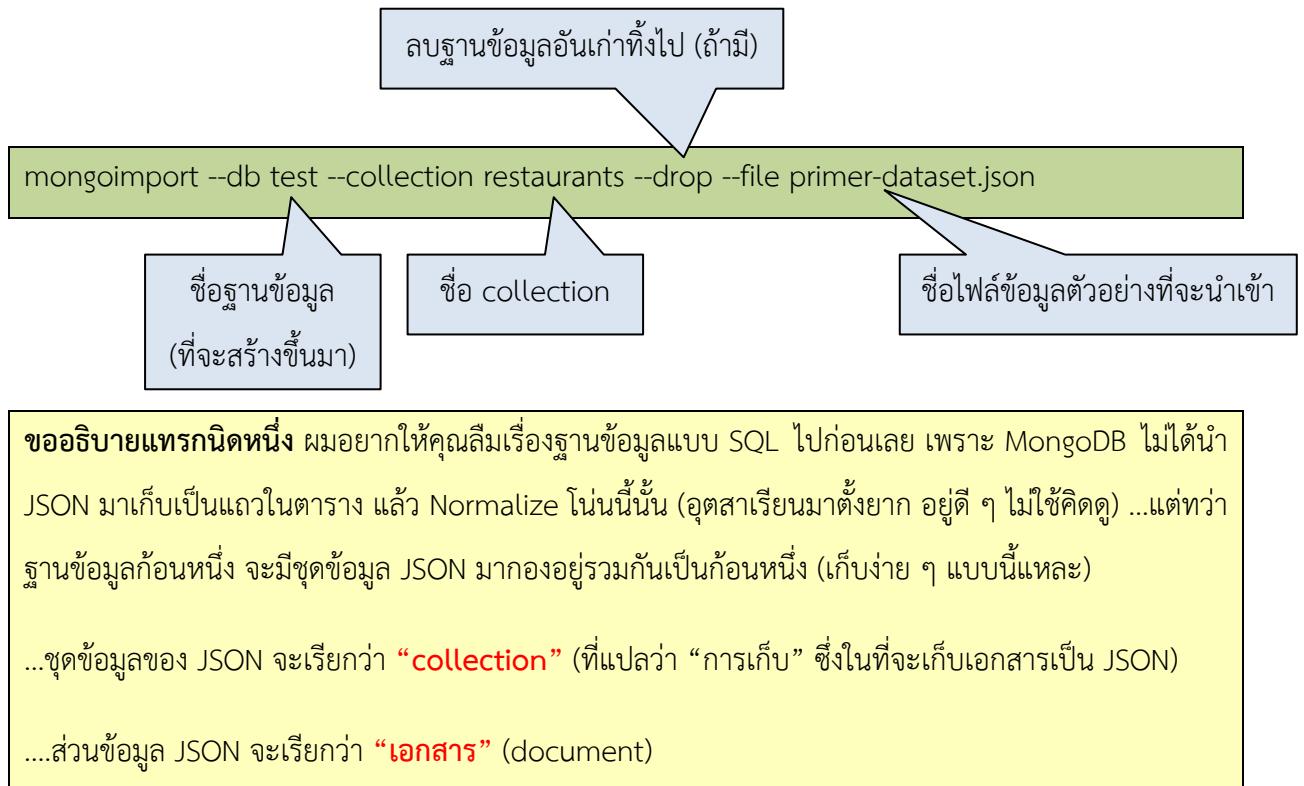
```
1 {"address": {"building": "10C7", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "borough": "Bronx", "cuisine": "Bakery", "g
2 {"address": {"building": "465", "coord": [-73.961704, 40.662942], "street": "Flatbush Avenue", "zipcode": "11225"}, "borough": "Brooklyn", "cuisine": "Hamburger"
3 {"address": {"building": "351", "coord": [-73.98513559999999, 40.7676919], "street": "West 57 Street", "zipcode": "10019"}, "borough": "Manhattan", "cuisine": "
4 {"address": {"building": "2760", "coord": [-73.98241999999999, 40.579305], "street": "Stillwell Avenue", "zipcode": "11224"}, "borough": "Brooklyn", "cuisine": "
5 {"address": {"building": "97-22", "coord": [-73.8601152, 40.7311739], "street": "63 Road", "zipcode": "11374"}, "borough": "Queens", "cuisine": "Jewish/Kosher",
6 {"address": {"building": "8825", "coord": [-73.8803827, 40.7643124], "street": "Astoria Boulevard", "zipcode": "11369"}, "borough": "Queens", "cuisine": "American"
7 {"address": {"building": "2226", "coord": [-74.1377282, 40.5119572], "street": "Victory Boulevard", "zipcode": "10314"}, "borough": "Staten Island", "cuisine": "
8 {"address": {"building": "7114", "coord": [-73.9068506, 40.5199041], "street": "Avenue U", "zipcode": "11234"}, "borough": "Brooklyn", "cuisine": "Delicatessen"
9 {"address": {"building": "6469", "coord": [-74.00528899999999, 40.628886], "street": "Nostrand Avenue", "zipcode": "11226"}, "borough": "Brooklyn", "cuisine": "Ice Cr
10 {"address": {"building": "1839", "coord": [-73.9482609, 40.5408271], "street": "Southern Boulevard", "zipcode": "10460"}, "borough": "Bronx", "cuisine": "American"
11 {"address": {"building": "2310", "coord": [-73.8786113, 40.8502882], "street": "11 Avenue", "zipcode": "11219"}, "borough": "Brooklyn", "cuisine": "American"
12 {"address": {"building": "7715", "coord": [-73.9973325, 40.51174889999999], "street": "18 Avenue", "zipcode": "11214"}, "borough": "Brooklyn", "cuisine": "American"
13 {"address": {"building": "1269", "coord": [-73.871194, 40.6730973], "street": "Sutter Avenue", "zipcode": "11208"}, "borough": "Brooklyn", "cuisine": "Chinese",
14 {"address": {"building": "1", "coord": [-73.96926909999999, 40.7385235], "street": "East 66 Street", "zipcode": "10065"}, "borough": "Manhattan", "cuisine": "Chin
```



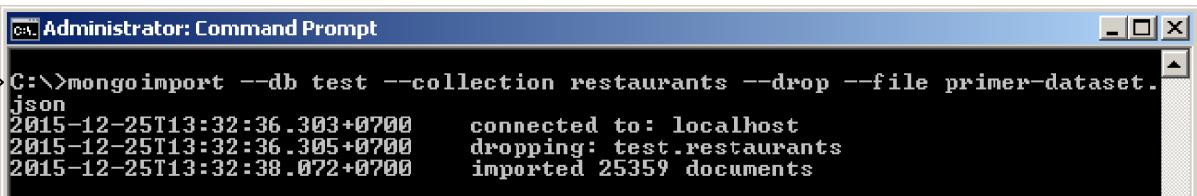
*** สำหรับข้อมูล JSON ถ้าใครอยู่สายเว็บไซต์ ก็น่าจะรู้จักกันดีอยู่แล้วแหล่ โดยมันเป็นชนิดข้อมูลแบบหนึ่ง ซึ่งจะนิยมใช้แล้วเปลี่ยนข้อมูลระหว่างคลื่อนตัวกับเชิร์ฟเวอร์

แต่ถ้าไม่รู้จัก อาจลองไปศึกษาค้นคว้าในอินเตอร์ดู ...ไม่ยากครับ ง่าย ๆ

2) ไปติดเครื่องท่อรี C:\ ซึ่งได้เก็บไฟล์ในข้อ 1 แล้วพิมพ์คำสั่งต่อไปนี้ เพื่อนำเข้าข้อมูลตัวอย่าง

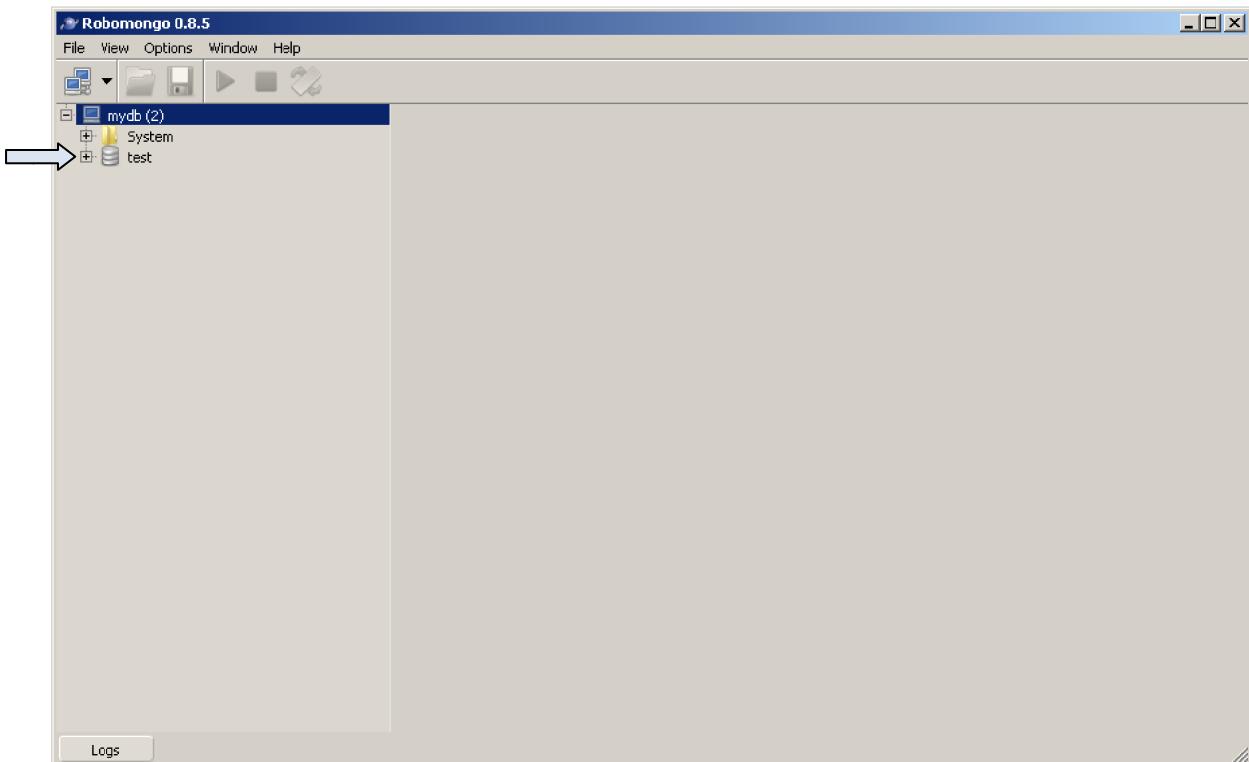


จากคำสั่งที่เขียนไว้ในตอนต้น เมื่อผนพิมพ์ลงไป ก็จะได้ผลลัพธ์ดังนี้



```
C:\>mongoimport --db test --collection restaurants --drop --file primer-dataset.json
2015-12-25T13:32:36.303+0700      connected to: localhost
2015-12-25T13:32:36.305+0700      dropping: test.restaurants
2015-12-25T13:32:38.072+0700      imported 25359 documents
```

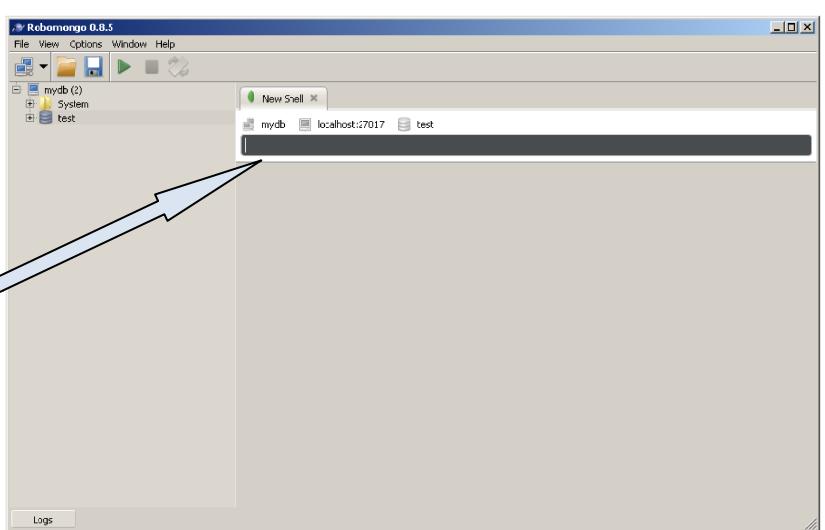
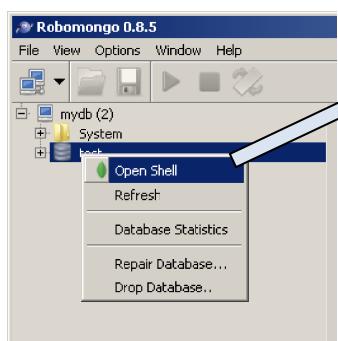
เมื่อผลลองเปิด Robomongo ขึ้นมา พร้อมทั้งเชื่อมต่อไปยัง MongoDB ตามที่ตั้งค่าไว้ในบทก่อนหน้านี้ (ชื่อการเชื่อมต่อคือ “mydb”) ... ก็จะเห็นว่ามีฐานข้อมูลชื่อ “test” ถูกสร้างขึ้นมา



หลังจากนั้นเมื่อคลิกขวาชื่อ

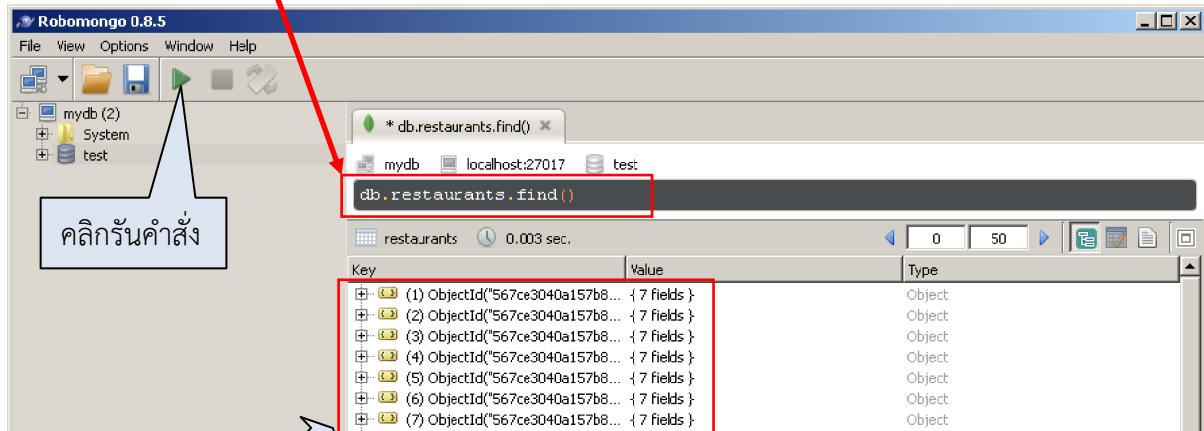
“test” และเปิด “Open

shell” ขึ้นมา



จากนั้นผู้ใช้พิมพ์คำสั่งในช่องคอมมานไลน์ของ Robomongo เพื่อค้นหาเอกสารจาก collection “restaurants” (ถ้าไม่ออก ก็คล้ายคำสั่ง select * from restaurants) ดังนี้

db.restaurants.find()



ตามภาพคุณจะเห็นผลการค้นหา และถ้าคลิก (เครื่องหมาย +) ก็จะเห็นตามภาพข้างล่าง ซึ่งมีรายละเอียดของค่า Key กับ Value ของเอกสาร JSON ทั้งหมดเลย (เก็บง่ายมาก)

This screenshot shows a detailed view of one of the documents returned by the query. The document structure is expanded to show nested fields like address, building, coord, street, zipcode, borough, cuisine, and grades. The "grades" field is expanded to show three elements, each with date, grade, and score. The "Value" column for the "grades" field is highlighted with a red box.

Key	Value	Type
(1)	{ 7 fields }	Object
_id	ObjectId("567ce3040a157b8...")	ObjectId
address	{ 4 fields }	Object
building	351	String
coord	Array [2]	Array
0	-73.985136	Double
1	40.767692	Double
street	West 57 Street	String
zipcode	10019	String
borough	Manhattan	String
cuisine	Irish	String
grades	Array [4]	Array
0	{ 3 fields }	Object
date	2014-09-06 00:00:00.000Z	Data
grade	A	String
score	2	Int32
1	{ 3 fields }	Object
date	2013-07-22 00:00:00.000Z	Data
grade	A	String
score	11	Int32
2	{ 3 fields }	Object

สำหรับรายละเอียดของโครงสร้าง JSON ที่เก็บใน MongoDB ยังมีเรื่องราวอีกพ่อครว เพราแม้สามารถมี JSON หรืออาร์เรย์ ข้อนอยู่ข้างในได้หลาย ๆ ระดับชั้น แต่เล่นนี้จะอธิบายให้เห็นแค่นี้ก่อน ...ไม่ลงลึกมาก

ถ้าตอนอยากระบุค้นหาเอกสารด้วยการใช้คอมมาน์ไลน์ ก็แลดูยุ่งยากสักหน่อย เพราะต้องติดต่อเข้าไปยังฐานข้อมูล “test” ก่อน ดังนี้

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

หลังจากนั้นจึงพิมพ์คำสั่ง

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

ลองดูตัวอย่างคำสั่งบนคอมมาน์ไลน์เต็ม ๆ ดังนี้จากอ้างอิงล่าสุด ...ซึ่งจะอ่านลำบากมาก 555

```
C:\>mongo test
2015-12-25T15:47:10.298+0700 I CONTROL [main] Hotfix KB2731284 or later update
is not installed, will zero-out data files
MongoDB shell version: 3.2.0
connecting to: test
> db.restaurants.find()
[{"_id": ObjectId("567ce3040a157b823de3e842"), "address": {"building": "351", "coord": [-73.9851355999999, 40.7676919], "street": "West 57 Street", "zipcode": "10019"}, "borough": "Manhattan", "cuisine": "Irish", "grades": [{"date": ISODate("2014-09-06T00:00:00Z"), "grade": "A", "score": 2}, {"date": ISODate("2013-07-22T00:00:00Z"), "grade": "A", "score": 11}, {"date": ISODate("2012-07-31T00:00:00Z"), "grade": "A", "score": 12}, {"date": ISODate("2011-12-29T00:00:00Z"), "grade": "A", "score": 12}], "name": "The Dubliner", "restaurant_id": "30191841"}, {"_id": ObjectId("567ce3040a157b823de3e843"), "address": {"building": "97-2", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "borough": "Bronx", "cuisine": "Bakerly", "grades": [{"date": ISODate("2014-03-03T00:00:00Z"), "grade": "A", "score": 2}, {"date": ISODate("2013-09-11T00:00:00Z"), "grade": "A", "score": 6}, {"date": ISODate("2011-11-23T00:00:00Z"), "grade": "A", "score": 10}, {"date": ISODate("2011-03-10T00:00:00Z"), "grade": "A", "score": 9}, {"date": ISODate("2011-03-10T00:00:00Z"), "grade": "B", "score": 14}], "name": "Morris Park Bake Shop", "restaurant_id": "30075445"}]
```

หวังว่าคุณคงเห็นໄວเดียวกันใช้งาน MongoDB

Robomongo) ซึ่งผมจะขอสรุปวิธีใช้งานผ่านคอมมาน์ไลน์ได้ดังนี้

บนคอมมาน์ไลน์ ...โดยไม่ผ่านเครื่องมือช่วย (เช่น

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

คำถาม เวลาเขียนโปรแกรม ถ้าไม่ใช้ภาษา SQL คุยกับ MongoDB ...แล้วใช้ภาษาหลักอะไรในการคุย?

คำตอบ ใช้ภาษาอังกฤษ

...และ ๆ ๆ ล้อเล่นครับ มันใช้หลักหลากรากภาษาเขียนโปรแกรมมาก ๆ

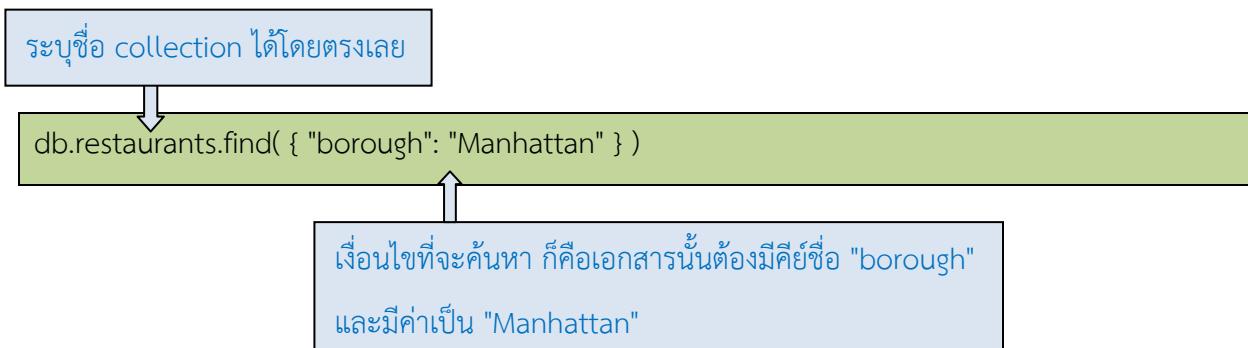
ภาษาที่ใช้คุยกับ MongoDB

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

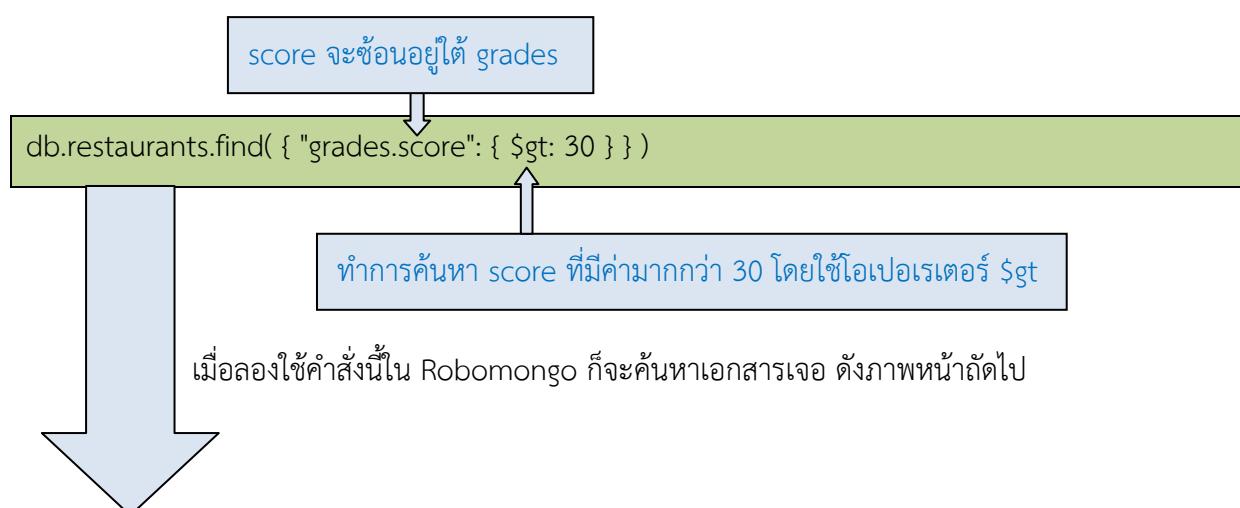
1. ใช้ภาษาสคริปต์บน Node.js (เล่นนี้จะกล่าวถึง)
2. Python (โค้ดจะสั้นกว่าภาษาอื่นเยอะเลย แต่ทว่าเล่นนี้ไม่ได้กล่าวถึง)
3. C++ (เล่นนี้ไม่ได้กล่าวถึง)
4. Java (เล่นนี้ไม่ได้กล่าวถึง)
5. C# (เล่นนี้ไม่ได้กล่าวถึง)

ไม่เพียงเท่านั้น เราสามารถใช้คำสั่งคอมมานไลน์ของ MongoDB ได้โดยตรง ซึ่งผมจะอธิบายโดยย่อ ...แต่ถ้าอยากรอ่านรายละเอียดมากกว่านี้ ก็ให้ไปที่ <https://docs.mongodb.org/getting-started/shell/> เพราะผมก็สรุปมาจากการที่นี่แหละ ...และต่อไปเราจะลองมาดูคำสั่งคอมมานไลน์ของ MongoDB โดยตรงกัน

ถ้าต้องค้นหาเอกสาร ก็ให้ระบุเงื่อนไข (Criteria) ในการค้นหา ดังตัวอย่าง



หรือจะระบุเงื่อนไขในการค้นหาเพิ่มเติมด้วยก็ได้



db.restaurants.find({ "grades.score": { \$gt: 30 } })

Key	Value	Type
(1) ObjectId("567fbee25fd83cebc19ad9fc")	{ 7 fields }	Object
_id	ObjectId("567fbee25fd83cebc19ad9fc")	ObjectId
address	{ 4 fields }	Object
borough	Queens	String
cuisine	American	String
grades	Array [4]	Array
0	{ 3 fields }	Object
date	2014-11-15 00:00:00.000Z	Date
grade	Z	String
score	38	Int32
1	{ 3 fields }	Object
2	{ 3 fields }	Object
3	{ 3 fields }	Object
name	Brunos On The Boulevard	String
restaurant_id	40356151	String
(2) ObjectId("567fbee25fd83cebc19ada03")	{ 7 fields }	Object
(3) ObjectId("567fbee25fd83cebc19ada29")	{ 7 fields }	Object
(4) ObjectId("567fbee25fd83cebc19ada2c")	{ 7 fields }	Object
(5) ObjectId("567fbee25fd83cebc19ada2d")	{ 7 fields }	Object
(6) ObjectId("567fbee25fd83cebc19ada35")	{ 7 fields }	Object
(7) ObjectId("567fbee25fd83cebc19ada36")	{ 7 fields }	Object
(8) ObjectId("567fbee25fd83cebc19ada3a")	{ 7 fields }	Object
(9) ObjectId("567fbee25fd83cebc19ada40")	{ 7 fields }	Object
(10) ObjectId("567fbee25fd83cebc19ada41")	{ 7 fields }	Object
(11) ObjectId("567fbee25fd83cebc19ada57")	{ 7 fields }	Object
(12) ObjectId("567fbee25fd83cebc19ada5f")	{ 7 fields }	Object
(13) ObjectId("567fbee25fd83cebc19ada63")	{ 7 fields }	Object
(14) ObjectId("567fbee25fd83cebc19ada66")	{ 7 fields }	Object
(15) ObjectId("567fbee25fd83cebc19ada69")	{ 7 fields }	Object
(16) ObjectId("567fbee25fd83cebc19ada80")	{ 7 fields }	Object

เอกสารที่เหลือ

เราสามารถอัพเดตเอกสาร ด้วยคำสั่งดังต่อไปนี้

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

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

เราสามารถลบเอกสารออกได้ ด้วยการระบุเงื่อนไข ดังตัวอย่าง

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

หรือจะลบ restaurants ทั้งไปเลยก็ได้ (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 ยawaHNอย ซึ่งอาจไม่คุ้นชินกับคนที่เคยเขียนภาษา SQL มา ก่อน

*** ถ้ายังไม่เห็นภาพการทำงาน ก็ขอให้ดูตัวอย่างในบทถัดไป ต่อจากนี้แล้วกันนะครับ

....บทนี้ขอเกริ่นนำคร่าว ๆ ไปก่อน

ໃຊ້ Node.js

ໃນບທນີຈະກລາວດີງການໃຊ້ຈາວສຄຣີປ່ຕໍເຂົ້ານເປັນສຄຣີປ່ຕໍ ເພື່ອຕິດຕ່າງມີກັບ MongoDB ເບື້ອງຕັນ ແຕ່ທວ່າໄຟລ໌ ສຄຣີປ່ຕໍດັກລ່າວຈະໃຊ້ Node.js ເປັນຕົວຮັນອີກທີ (ໄມ້ໃຊ້ເວັບບຣາວເຊອຮ່) ດ້ວຍເຫຼຸ້ນນີ້ຄ້າອາຍາກອ່ານເນື້ອທາຕ່ອໄປຮູ້ເຮືອງ ຄຸນຕ້ອງໄປທີ່ລົງຄໍ http://www.patanasongsivilai.com/itebook_form.html

- 1) ເພື່ອດາວໂຫຼດ “ວິທີຕິດຕັ້ງ Node.js ແລະ npm ເບື້ອງຕັນ” ມາອ່ານກ່ອນ
- 2) ແລະອາຍາກໃຫ້ຄຸນອ່ານໜັງສູ່ “ເສີຍດາຍໄນ້ໄດ້ອ່ານຈາວສຄຣີປ່ຕໍ ຝຶ່ງເຊີ່ງຟເວົ້ວ່າ (Node.js ຈຸບບ່ອຍ່ອ)” ເລີ່ມ
1 ນາ່ອ່ານເສີມເພີ່ມດ້ວຍ

ເອາແහລະ ພມຈະຄືດວ່າຄຸນເຂົ້າໃຈ Node.js ດີແລ້ວ ...ແຕ່ຄ້າໄໝເຂົ້າໃຈ ກໍລອງອ່ານເນື້ອທາຕ່ອໄປນີ້ຮ່ວາງໆ ໃນໄອເດີຍ ແລ້ວກັນເນອະ ໄນຍາກ ...ແຕ່ຈະມີນໂຄດໜ່ອຍ ໂດຍພມຈະພາທຳເວີຣົກຂຶ້ອປົງຈ່າຍ ແລ້ວ ດັ່ງນີ້

ເຕີຣີມໂປຣເຈກໃຫ້ພຣ້ອມ

ສ້າງໄຟລ໌ເດວຽ່ນຂຶ້ນມາດັ່ງນີ້

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

ໜັກນີ້ຈະໃຊ້ npm ຕິດຕັ້ງມອດູລຸ “mongoose” ດັ່ງນີ້

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

ສ່ວນໂຄຮ່ວງສ້າງໂປຣເຈກຈະເປັນດັ່ງນີ້

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

ต้องบอกอย่างนี่นะครับ ... ໄວเดียวนจะเขียนโค้ด jawscript ทั้งหมดไว้ที่ไฟล์ “test.js” และสั่งรันบน Node.js ด้วยคำสั่งดังนี้

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

ซึ่งไฟล์ test.js และผลการทำงานของมัน ผูกก็จะเอาไว้เชื่อมิบายเนื้อหาต่อไปด้านนี้

เขียนส่วนหัวของไฟล์ test.js

ผมจะเขียนส่วนหัวของไฟล์ test.js แบบนี้罷ๆ ดังโค๊ดข้างล่าง

```
var MongoClient = require('mongodb').MongoClient; // โหลด模偶 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 ที่เราได้ติดตั้งลงมา ด้วยรูปแบบดังนี้

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

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

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

ค้นหาข้อมูล

ผมจะทำการค้นหา (Query) เอกสารทั้งหมด จาก collection ที่ชื่อ "restaurants" ของบก่อนหน้า 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 จะใช้ไปยังเอกสารที่ค้นพบ จะคล้ายกับ cursor เวลาใช้งานในฐานข้อมูลแบบ SQL
    cursor.each(function(err, doc) { // แต่ละรอบที่ each() เข้าถึงเอกสารใน restaurants ฟังก์ชันคอลแบ็คจะถูกเรียกให้ทำงาน
        assert.equal(err, null);
        if (doc != null) {
            console.dir(doc); // พิมพ์ชื่อที่จะแสดงโครงสร้างออบเจกต์ doc (ไม่ใช่พิมพ์ชื่อมาตรฐานในภาษาสคริปต์)
        } else {
            callback(); // เมื่อเข้าถึงเอกสารใน restaurants ครบทุกตัวแล้ว ก็จะเรียกฟังก์ชันคอลแบ็คให้ทำงาน
        }
    });
}; // สิ้นสุด cursor.each()

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

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

โค้ดมันจะยาวไปหน่อย จริง ๆ ไม่อยากให้คุณสนใจรายละเอียดมากเท่าไร ...แค่เห็นเป็นໄอเดียพอแหละ

แต่ผมอยากระนำว่า ถ้าใครชอบเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ต้องปรับมุมมองให้เห็นเป็นการเขียนโปรแกรมแบบ Function programming หรือการเขียนโปรแกรมเชิงฟังก์ชันแท้ ๆ

...แน่นอนบางคนอาจไม่ชอบเท่าไร เพราะเห็นฟังก์ชันซ้อนกันไปซ้อนกันมา ชวนให้ดูตาลาย แฉมฟังก์ชันคอลแบ็คแท้ละตัว มันยังทำงานแบบซิ่งครั้นสือกัดดวย (ตามยหนักกว่าเก่าอีก)

...ถ้าใครยังไม่เข้าใจการเขียนโค้ดบน Node.js แนะนำให้ไปอ่านหนังสือที่ผมแนะนำข้างต้นก่อนครับ

แต่ถึงอย่างไรก็ตาม ผู้ใดก็อยากรอชิบะイメรอด db.collection().find() เพื่อใช้มันค้นหาเอกสารใน restaurants โดยเราสามารถระบุเงื่อนไขในรูปแบบดังตัวอย่าง

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

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

ซึ่งในตัวอย่างดังกล่าว จะระบุเงื่อนไขในเมรอดเป็น

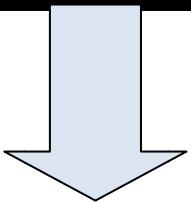
```
{ "borough": "Manhattan" }
```

แต่ถ้าเรียกเป็น db.collection().find() เฉย ๆ โดยไมระบุเงื่อนไขอะไรเลย (ไม่ระบุค่าอากวิเมนต์) ก็จะค้นหาเอกสารทั้งหมดใน collection ครับ

ส่วนผลลัพธ์ของโค้ดในตัวอย่างดังกล่าว หลังจากพิมพ์คำสั่ง

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

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



```
Administrator: Command Prompt
restaurant_id: '50018785' ,
< _id: ObjectId < _bsontype: 'ObjectId', id: '01 \u0006\n\u0015\x{e= }\u0004' >,
address:
  { building: '1631',
    coord: [ -73.947419, 40.7903305 ],
    street: 'Lexington Ave',
    zipcode: '10029' },
borough: 'Manhattan',
cuisine: 'Other',
grades: [],
name: 'Dong\'s Great Wok Garden II',
restaurant_id: '50018787' ,
< _id: ObjectId < _bsontype: 'ObjectId', id: '01 \u0006\n\u0015\x{e= }\u0005' >,
address:
  { building: '200',
    coord: [ -73.98823060000001, 40.7587649 ],
    street: '8Th Ave',
    zipcode: '10036' },
borough: 'Manhattan',
cuisine: 'Other',
grades: [],
name: 'Whitmans',
restaurant_id: '50018788' ,
< _id: ObjectId < _bsontype: 'ObjectId', id: '01 \u0006\n\u0015\x{e= }\u0006' >,
address:
```

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( { // เพิ่มเอกสารลงใน 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" // ผู้ซื้อชื่นชอบเบ็ค
  }, function(err, result) { // ผู้ซื้อชื่นชอบเบ็ค
    assert.equal(err, null);

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

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

เป็นเอกสารที่เพิ่มเข้าไปใน restaurants

// โค้ดชุดนี้คล้ายกับตัวอย่างที่แล้ว

```

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

```

ในตัวอย่างนี้ เมื่อ用 db.collection().insertOne() จะเพิ่มเอกสารลงไปใน restaurants ซึ่งมีผลการทำงานดังนี้

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

หลังจากนั้น ผู้จะลองพิมพ์คำสั่งบน Robomongo เพื่อค้นหาเอกสารดังกล่าวที่เพิ่งเพิ่มเข้าไป ดังนี้

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

ทุกครั้งที่เพิ่มเอกสารเข้าไป
ถ้าข้อมูลใน JSON ไม่ได้ระบุชื่อคีย์เป็น “_id”
MongoDB จะสร้าง _id มาให้โดยอัตโนมัติ

Key	Value	Type
<code>_id</code>	<code>{ 7 fields :</code> <code>ObjectId('567ec73dfb9ef78811b28dc4")}</code>	<code>Object</code>
<code>address</code>	<code>{ 4 fields :</code> <code>street</code> : '2 Avenue' <code>zipcode</code> : '10075' <code>building</code> : '1480'	<code>Object</code>
<code>coord</code>	<code>Array [2]</code> <code>0</code> : { <code>lat</code> : -73.955741, <code>lon</code> : 40.772027} <code>1</code> : { <code>lat</code> : -73.955741, <code>lon</code> : 40.772027}	<code>Array</code>
<code>borough</code>	'Manhattan'	<code>String</code>
<code>cuisine</code>	'Italian'	<code>String</code>
<code>grades</code>	<code>Array [2]</code> <code>0</code> : { <code>date</code> : '2014-10-01 00:00:00.000Z', <code>grade</code> : 'A', <code>score</code> : 11} <code>1</code> : { <code>date</code> : '2014-01-16 00:00:00.000Z', <code>grade</code> : 'B', <code>score</code> : 17}	<code>Object</code>
<code>name</code>	'Vella'	<code>String</code>
<code>restaurant_id</code>	'41704621'	<code>String</code>

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

อัพเดตข้อมูล

สำหรับวิธีอัพเดตเอกสารใน restaurants မ苟ก็จะนำโค้ด 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

- `$set: { "cuisine": "Thailand (New)" }` → โอเพอเรเตอร์ `$set` จะบอกให้อัพเดตคีย์ที่ชื่อ "cuisine" ให้มีค่าเป็น "Thailand (New)"
- `$currentDate: { "lastModified": true }` → โอเพอเรเตอร์ `$currentDate` จะทำการเพิ่มพิวเดตตัวสุดท้ายเป็นคีย์ที่ชื่อ "lastModified" และมีค่าเป็นวันที่ปัจจุบัน

เมื่อลองค้นเอกสารดังกล่าวใน Robomongo ด้วยคำสั่งในหน้าถัดไป

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

Key	Type	Value
ObjectID("567fbad35fd83cebc19a8e72")	Object	{ 7 fields }
_id	ObjectId	ObjectId("567fbad35fd83cebc19a8e72")
address	Object	{ 4 fields }
borough	String	Manhattan
cuisine	String	American
grades	Array	Array [3]
name	String	Juni
restaurant_id	String	41156888

Key	Type	Value
ObjectID("567fbad35fc83cebc19a8e72")	Object	{ 8 fields }
_id	ObjectId	ObjectId("567fbad35fc83cebc19a8e72")
address	Object	{ 4 fields }
borough	String	Manhattan
cuisine	String	Thailand (New)
grades	Array	Array [3]
name	String	Juni
restaurant_id	String	41156888
lastModified	Date	2015-12-27 10:22:17.626Z

สำหรับเมธอด db.collection().updateOne() จะอัปเดตแค่เอกสารตัวแรกที่เจอก่อน แต่ถ้าต้องการอัปเดตเอกสารทั้งหมดที่ค้นเจอ ...ก็ให้ใช้ updateMany() ดังตัวอย่าง

```
var updateRestaurants = function(db, callback) {
  db.collection('restaurants').updateMany( // อัปเดตเอกสารทั้งหมด 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();
    });
});
```

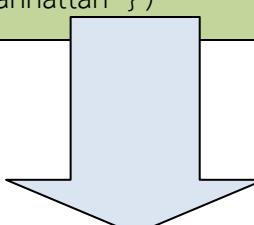
โค้ดชุดนี้ได้ใช้เมธอด db.collection().deleteMany() ลบข้อมูลทั้งหมด ตามเงื่อนไขที่ระบุไว้ก็คือ

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

เมื่อรันไฟล์ test.js ก็จะลบข้อมูลทั้ง และผมจะลองใช้คำสั่งต่อไปนี้ ค้นหาเอกสารใน Robomongo ดูครับ

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

ดูหน้าถัดไป



The screenshot shows the MongoDB shell interface. The top bar displays the command: `db.restaurants.find({ "borough": "Manhattan" })`. Below the command, the results pane shows a table with three columns: Key, Value, and Type. The 'Key' column contains 21 ObjectId values, each corresponding to a document in the collection. The 'Value' column shows the document structure, which is a JSON object with 7 fields. The 'Type' column indicates that all documents are of type Object. The results are paginated with buttons for 0, 50, and 50+.

หลังจากลบเอกสารไปแล้ว ก็จะค้นหาไม่เจอ

The screenshot shows the MongoDB shell interface again. The command `db.restaurants.find({ "borough": "Manhattan" })` is run, and the results pane displays the message: `Fetched 0 record(s) in 30ms`. This indicates that no documents were found, confirming the deletion.

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/>

เพราะตัวอย่างทั้งหมดผมก็สรุปมาจากการลิงค์นี้แหละ (ขอรับสโค้ดเดียวกัน) 

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/>