MongoDB y Java: Parte V, más consultas, usuarios, roles y autenticación

Para finalizar vamos a ver algunas otras acciones y consultas que podemos realizar sobre la base de datos, cómo activar y configurar la autenticación, y el uso de usuario y roles. Con la autenticación conseguiremos que solo los usuarios que definamos, haciendo uso de una contraseña y mediante la asignación de roles, puedan acceder a cierta base de datos y dentro de ella realizar determinadas acciones.

Creando bases de datos y colecciones desde Java

Ya hemos visto cómo utilizar la herramienta mongoimport para añadir información a una base de datos a partir de un archivo JSON. El driver MongoDB de Java también permite crear bases de datos y colecciones directamente.

Para crear una nueva base de datos, tenemos que usar getDatabase y sobre ella, getCollection. En el momento en que escribamos algo en la colección, se creará la base de datos.
[crayon-55b02cdc63079475322190/]
Si queremos que se cree explicitamente sin contenido, podemos usar createCollection:
[crayon-55b02cdc63086468523544/]
Para listar todas las bases de datos, podemos usar las siguientes líneas:
[crayon-55b02cdc6308d511641726/]
Y lo mismo para las colecciones:
[crayon-55b02cdc63093868288938/]
Estos dos últimos ejemplos recuperan todos los nombres de las bases de datos y las colecciones respectivamente, y los muestran uno por uno.

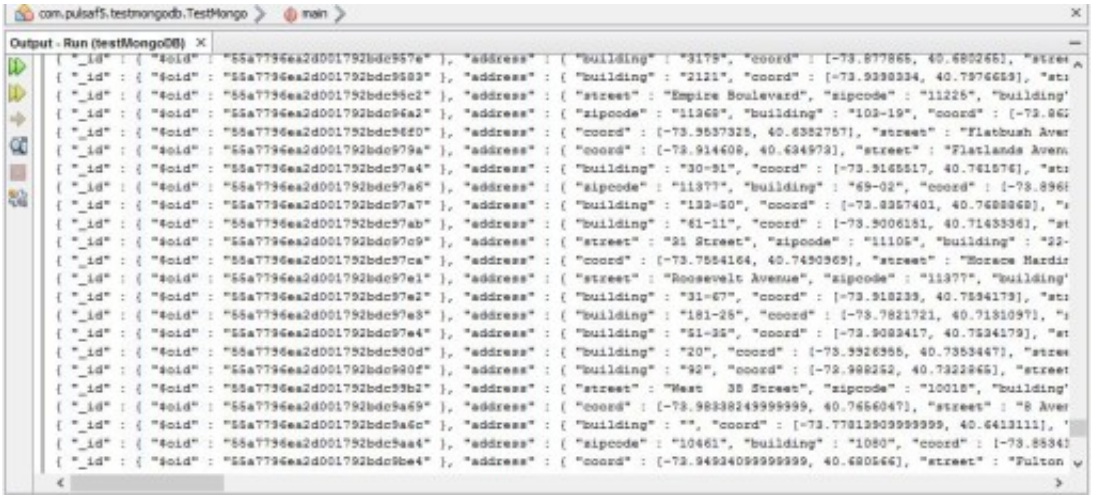
Para eliminar una colección o una base de datos existente, hacemos drop() sobre ella:
[crayon-55b02cdc63099167946674/]

Actualizar un documento

En el ejemplo de la parte IV de este tutorial ya hemos modificado un documento, aunque no lo hemos comentado.
[crayon-55b02cdc6309e215224951/]
Lo que hace esa línea es buscar el documento cuyo nombre es "Cafetería FIC" y establece el valor "Facultad de Informática" al campo "borough". La operación de actualización en este caso es "$set", que sustituye el valor anterior por el nuevo. Podéis consultar todas las operaciones de actualización soportadas en el link al final del post.

Conjuntos de documentos

En los ejemplos de consultas que hemos visto, suponíamos que devolvían un único documento, o forzábamos este comportamiento mediante el uso de first(). El caso más habitual al realizar consultas es que estas devuelvan un conjunto de documentos. Por ejemplo vamos a recuperar todos los restaurantes cuyo campo "cuisine" tiene el valor "Hamburgers".
[crayon-55b02cdc630a4059986632/]
Con printBlock lo que hacemos es pasarle al método forEach(), que recorre la lista de documentos devuelta por la consulta, la función que imprime el objeto en formato JSON. El resultado se puede ver en la siguiente imagen.



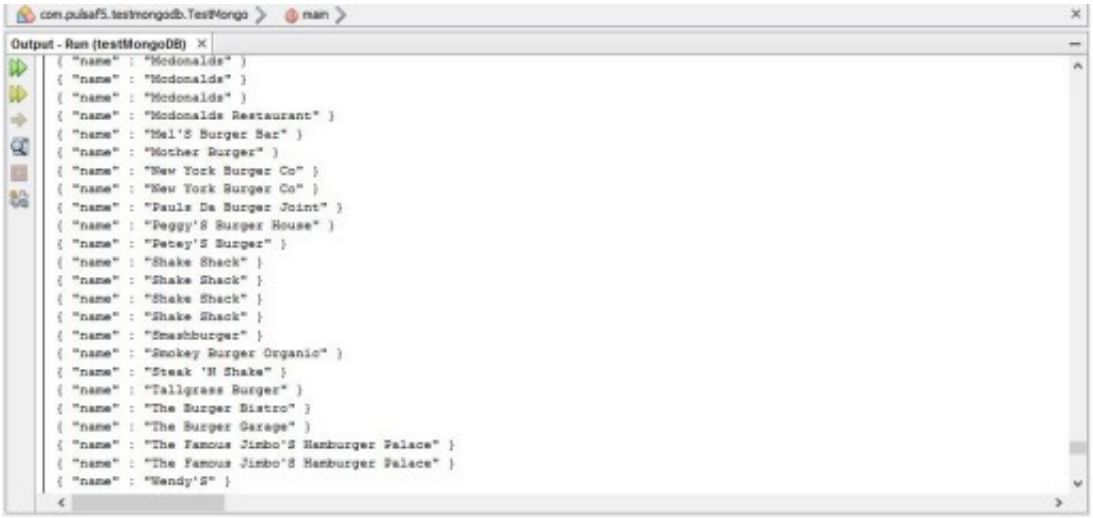
Proyecciones

En el ejemplo anterior recuperamos toda la información de los restaurantes. Si por ejemplo queremos mostrar únicamente el nombre, una opción sería, en la función que imprime cada uno de los documentos en formato JSON, filtrar el campo "name". Esto es poco práctico y muy costoso, ya que estamos cargando la base de datos recuperando un montón de información que no vamos a usar, para posteriormente filtrarla. Lo mejor es hacer uso de las proyecciones, que nos permitirán recuperar de cada documento o conjunto de ellos, únicamente los campos que nos sean útiles.
[crayon-55b02cdc630ab674371802/]
Como vemos usar proyecciones es tan simple como llamar a la función projection() sobre la consulta y pasarle las funciones de proyección que nos convengan. En este caso utilizamos excludeId() para eliminar los identificadores internos de MongoDB, e include("name") para que

muestre este campo que es el que nos interesa. Debido a que usamos dos proyecciones diferentes las agrupamos con fields() que crea una proyección combinando ambas. Más información sobre los tipos de proyecciones la encontraréis al final del post.

Ordenando los resultados

Finalmente, si queremos ordenar los resultados de una consulta, podemos hacer uso de la función sort():
[crayon-55b02cdc630b2845012015/]
A esta función le pasamos un criterio de ordenación, en este caso ascending("name") para que nos ordene los restaurantes por orden alfabético. De nuevo encontraréis más información sobre cómo ordenar las consultas en el link al final del post. El resultado de esta consulta es el siguiente:



Autenticación y roles

Hasta ahora nos hemos conectado directamente con las bases de datos sin control ninguno. Esto solo es válido para un entorno de pruebas, en la gran mayoría de casos deberemos restringir y controlar el acceso a las bases de datos desplegadas sobre MongoDB. Para permitir autenticación, lo primero es editar el archivo de configuración que habíamos creado anteriormente y añadir lo siguiente:
[crayon-55b02cdc630b8922404069/]
y a continuación reiniciamos mongod.exe. Si intentamos ejecutar nuestro proyecto desde NetBeans deberíamos ver el siguiente error:
[crayon-55b02cdc630be103619045/]

Ahora deberemos conectarnos usando un shell Mongo y haciendo uso de la "localhost exception" para crear el primer usuario. Para ello utilizamos el ejecutable mongo.exe presente en la carpeta de instalación de MongoDB, y desde un terminal escribimos lo siguiente:
[crayon-55b02cdc630c3497851935/]
A continuación creamos el administrador con las siguientes líneas:
[crayon-55b02cdc630c9627415728/]
El resultado, si todo ha ido bien, es el siguiente:

Nos desconectamos y nos volvemos a conectar con el usuario que hemos creado:
[crayon-55b02cdc630cf913902812/]
Y añadimos un nuevo usuario que tendrá acceso a las bases de datos:
[crayon-55b02cdc630d4538826384/]
Como vemos le estamos asignando los roles readWrite en las bases de datos "test" y "myNewBD".
Ahora en el proyecto Java de NetBeans, cambiaremos las primeras líneas en las que nos conectamos a la MongoDB por las siguientes:
[crayon-55b02cdc630da538720968/]
Si ejecutamos ahora el proyecto, veremos que nos deja realizar las consultas y accesos a la base de datos "test", pero salta una excepción cuando intentamos listar todas las bases de datos presentes, ya que dicho usuario no tiene el rol correspondiente asignado. Comentando las líneas en las que realizamos el listado, y ejecutamos de nuevo el proyecto nos saltará una nueva excepción al intentar hacer drop de la base de datos ya que, de nuevo, el usuario "testUser" carece de dicho rol. En este caso para solucionarlo, añadiremos el rol necesario para hacer el drop de "myNewDB" usando grantRolesToUser(). Nos conectamos al shell Mongo como la última vez y ejecutamos lo siguiente:
[crayon-55b02cdc630e0873163802/]
Con esto el usuario testUser podrá realizar el drop sin problemas.

Fuentes:

MongoDB field update operations list

MongoDB Java projection Class

MongoDB Java sorts Class

MongoDB authenticating

MongoDB enable authentication

MongoDB configuration security.authorization

MongoDB add user administrator

MongoDB db.createUser

MongoDB grantRolesToUser