**Programación Java de BD MongoDB**

**Objetivos:**

* Realizar conexiones con BD MongoDB
* Consultar datos de un documento
* Acceder a colecciones
* Añadir documentos a una colección
* Añadir atributos a un documento
* Modificar valores de atributos de un documento
* Eliminar documentos de una colección

**EJERCICIO**

Realiza un programa Java que conecta con la base de datos **geografía de MongoDB** usada en la hoja 3-03 y que presenta el siguiente menú y, en función de la opción elegida, realiza sobre la colección **ccaa** la operación solicitada en las condiciones que se indican a continuación:

1.- Añadir Comunidad Autónoma sin provincias

2.- Consultar Comunidad Autónoma

3.- Asignar provincias a Comunidad Autónoma

4.- Añadir provincia a Comunidad Autónoma

5.- Modificar nombre de Comunidad Autónoma

6.- Eliminar provincia en Comunidad Autónoma

7.- Asignar capital a Comunidad Autónoma

8.- Asignar fecha Estatuto Autonomía

9.- Eliminar Comunidad Autónoma

Cuando es necesario, se explica dentro de las opciones los métodos de la API de MongoDB que se deben usar y la sintaxis de dichos métodos.

**OPCIÓN 1:**

Se piden por teclado los siguientes datos de una Comunidad Autónoma:

* Código
* Nombre
* Abreviatura
* Nombre y habitantes de la ciudad capital de la Comunidad Autónoma.
* Habitantes
* Extensión

Con los datos recogidos, se debe añadir un nuevo documento a la colección ccaa con los nombres de atributos usados en la hoja 5-03. Los datos de la capital se guardan dentro de un subdocumento capital.

Se debe crear un objeto **Document** en el que se cargan todos los atributos y sus valores. Una vez que tenemos todos los atributos en el **Document,** lo insertamos mediante el método **insertOne.**

**A continuación tienes un código de ejemplo que añade un documento con los datos de Extremadura (incluyendo las provincias). Se supone que coleccion es un objeto de la clase Collection.**

*//Creamos el documento con un primer atributo*

Document doc2=new Document("codigo","09");

*//vamos añadiendo, uno a uno, los atributos restantes*

doc2.append("nombre","Extremadura");

doc2.append("abreviatura","EXT");

//Creamos un Document anidado para la capital

Document docCap=new Document ("nombre","Mérida")

.append("habitantes",59200);

doc2.append("capital",docCap);

*//Lista de Provincias que se carga en el array provincias*

List<String> lista=new ArrayList();

lista.add("Badajoz");

lista.add("Caceres");

doc2.append("provincias", lista);

doc2.append("habitantes",1067000);

doc2.append("extension", 41635);

*//insertamos el documento*

**coleccion.insertOne(doc2);**

También puedes usar un código como el siguiente. En este caso se va llamando de forma anidada al método append y se irán así añadiendo atributos al Document:

Document doc1=new Document("codigo","09")

.append("nombre","Extremadura")

.append("abreviatura","EXT")

.append("capital",

new Document ("nombre","Mérida")

.append("habitantes",59200))

.append("provincias", Arrays.asList("Badajoz","Cáceres"))

.append("habitantes",1067000)

.append("extension", 41635);

coleccion.insertOne(doc1);

**OPCIÓN 2:**

Se pide por teclado el código de una Comunidad Autónoma y, si existe, se escribe su nombre, número de habitantes, extensión y el nombre de su capital. A continuación, se escriben los datos de la Comunidad Autónona en formato JSON. Si no existe, se escribe un mensaje indicándolo.

Para hacer una consulta con un filtro, se usa el método **find de Collection** al que se le pasa un **filtro.**

**coleccion.find(filtro);**

Los resultados de la consulta de **find** se devuelven en un objeto **FindIterable.** A ese objeto se le pueden aplicar unos métodos. Los principales son:

|  |  |
| --- | --- |
| **Método** | **Realiza** |
| **first()** | Devuelve el primer Document del resultado |
| **iterator** | Devuelve un iterador (MongoCursor) de Document con todo el resultado que podemos recorrer con hasnext y next. |
| **into(ArrayList a)** | Carga los resultados en un ArrayList |
| **limit(int n)** | Del resultado generado, limita a n documents. |
| **projection(Bson bson)** | Del resultado generado selecciona los atributos indicados en bson. |
| **forEach(action)** | Para cada document del resultado realiza una acción |
| **sort(criterio)** | Ordena los documents del resultado |

Por otra parte, el método **find()** recibe un objeto **Filter.** Los objetos **Filter** se pueden crear con métodos de filtro. La sintaxis de los métodos de filtro es muy similar a los selectores de filtro que vimos en la presentación 4 de MongoDB.

Por ejemplo, podemos usar:

|  |  |
| --- | --- |
| **Método de filtro** | **Realiza** |
| **eq(atributo,valor)** | Compara si el valor del atributo es igual a valor |
| **ne(atributo,valor)** | Compara si el valor del atributo no es igual a valor |
| **gt(atributo,valor)** | Compara si el valor del atributo es mayor que valor |
| **and(cond1, cond2, …)** | Operación AND entre todas las condiciones de filtro indicadas. |
| **or(cond1, cond2, …)** | Operación OR entre todas las condiciones de filtro indicadas. |
| **not(cond)** | Operación NOT condición |

En nuestro caso, si tenemos un objeto Collection c, para obtener un Document con los datos de la Comunidad Autónoma de código “01”, haríamos:

Document doc=c.find(eq("codigo","01")).first();

Una vez, que tenemos un Document, para obtener el valor de un atributo habitantes (entero), podemos usar:

int habs=doc.getInteger("habitantes", -1);

// si no hay atributo carga -1

O también:

int habitantes=(Integer)doc.get("habitantes");

Para obtener el valor del atributo nombre:

String nom=(String)doc.get("nombre");

O bien:

String nom= doc.getString("nombre");

Para obtener el contenido del subdocumento capital:

Document docCap=(Document)doc.get("capital");

Y para obtener el contenido de doc en formato JSON:

String json= doc.toJson();

**OPCIÓN 3:**

Se pide por teclado el código de una comunidad autónoma y, si existe, se piden por teclado los nombres de las provincias de esa comunidad autónoma y se cargan esos nombres en un atributo **provincias** del documento correspondiente a la comunidad autónoma.

Para modificar documentos de una colección se pueden usar los métodos **updateOne y updateMany.**

Los dos métodos reciben un objeto Bson que indica la condición de filtrado (idéntico al que se puede usar en las consultas) y un objeto Bson que indica la operación de actualización a realizar. Para representar la operación de actualización, se usan métodos de modificación que son equivalentes a los operadores de actualización que vimos en la presentación 3 (set, unset, inc, addToSet, …).

Por ejemplo, para asignar “Extremadura” al atributo nombre, haríamos:

set("nombre","Extremadura")

Y para establecer que, en la colección c, el nombre de la comunidad autónoma de código “99” es “Extremadura”, haríamos:

c.updateOne(eq("codigo","99"), set("nombre","Extremadura"));

En nuestro caso, tenemos que cargar un atributo que tiene valores de tipo array. En el valor asignado se debe para un ArrayList con los nombres de las provincias.

**OPCIÓN 4:**

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado el nombre de una provincia de esa comunidad autónoma y se añade a las provincias ya existentes de esa comunidad autónoma.

Recuerda que el operador de modificación que usamos en las instrucciones mongodb para este tipo de modificación es **addToSet.**

Se debe comprobar si se ha realizado la modificación o no. Para ello, debes usar el objeto **UpdateResult** que pueden devolver los métodos **updeteOne y updateMany.** Un objeto de este tipo, dispone de los métodos **getMatchedCount() y getModifiedCount()** que devuelven cuantos documentos cumplían la condición de filtro de update y cuantos fueron realmente modificados.

**OPCIÓN 5:**

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado un nuevo nombre de esa comunidad autónoma y se modifica en el documento de la colección ese nombre.

**OPCIÓN 6:**

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado el nombre de una provincia y se elimina ese nombre en el array de provincias.

Se debe escribir un mensaje que confirme si se ha realizado o no la eliminación de la provincia.

**OPCIÓN 7:**

Se pide por teclado el código de una comunidad autónoma y, si existe y no tiene asignada capital, se piden los datos de su capital (nombre y habitantes) y se asignan al documento de la comunidad autónoma en el atributo capital.

**OPCIÓN 8:**

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado la fecha de la entrada en vigor de su estatuto de autonomía en formato dd/MM/yyyy y se almacena en un atributo fecha-estatuto.

**OPCIÓN 9:**

Se pide por teclado el código de una comunidad autónoma y, si existe, se escribe su contenido en JSON y se pregunta si se quiere eliminar. Si la respuesta es S, se elimina el documento de la colección.