GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN

CIENCIA DE DATOS

Agenda
Sesión 6/18

Python y MongoDB

- PyMongo
- Bases de datos en MongoDB
- Colecciones
- Búsquedas, limites y Filtros
- Ordenar
- Eliminar y Actualizar

Review de sesión anterior

Python y MongoDB

MongoDB almacena datos en documentos similares a JSON, lo que hace que la base de datos sea muy flexible y escalable.

Para poder experimentar con los ejemplos de código de este tutorial, necesitará acceso a una base de datos MongoDB.

Puede descargar una base de datos gratuita de MongoDB en https://www.mongodb.com

O comience de inmediato con un servicio en la nube de MongoDB en https://www.mongodb.com/cloud/atlas

PyMongo

Python necesita un controlador MongoDB para acceder a la base de datos MongoDB.

Le recomendamos que utilice PIP para instalar "PyMongo".

Desde terminal ejecutar:

pip install pymongo

pip install "pymongo[srv]"

PyMongo

Prueba PyMongo

Para probar si la instalación fue exitosa, o si ya tiene "pymongo" instalado, cree una página de Python con el siguiente contenido:

```
import pymongo
```

Si el código anterior se ejecutó sin errores, "pymongo" está instalado y listo para ser utilizado.

Bases de datos en MongoDB

Crear una base de datos

Para crear una base de datos en MongoDB, comience creando un objeto MongoClient, luego especifique una URL de conexión con la dirección IP correcta y el nombre de la base de datos que desea crear.

MongoDB creará la base de datos si no existe y se conectará a ella.

Crear una base de datos

```
import pymongo

myclient = pymongo.MongoClient("URL del Servidor")

mydb = myclient["mydatabase"]
```

URL del Servidor:

podría ser localhost podría ser la dirección que Atlas nos proporciona

Importante: En MongoDB, no se crea una base de datos hasta que obtiene contenido.

Crear una base de datos

```
import pymongo

myclient = pymongo.MongoClient("URL del Servidor")

mydb = myclient["mydatabase"]
```

URL del Servidor:

podría ser localhost podría ser la dirección que Atlas nos proporciona

Importante: En MongoDB, no se crea una base de datos hasta que obtiene contenido.

Crear una base de datos

Compruebe si existe la base de datos

Recuerde: en MongoDB, una base de datos no se crea hasta que obtiene contenido, por lo que si es la primera vez que crea una base de datos, debe completar los siguientes dos capítulos (crear una colección y crear un documento) antes de verificar si la base de datos existe.

```
print(myclient.list_database_names())
dblist = myclient.list_database_names()
if "mydatabase" in dblist:
  print("The database exists.")
#Devolverá una lista de las bases de datos de tu sistema
```

Colecciones

Crear una colección

Para crear una colección en MongoDB, use el objeto de base de datos y especifique el nombre de la colección que desea crear.

MongoDB creará la colección si no existe.

```
import pymongo

myclient = pymongo.MongoClient("URL del Servidor")
mydb = myclient["mydatabase"]   #Se accede a la base de datos
mycol = mydb["customers"]       #Se accede a la colección
```

Colecciones

Puede comprobar si existe una colección en una base de datos enumerando todas las colecciones:

```
print(mydb.list_collection_names())
```

```
collist = mydb.list_collection_names()
if "customers" in collist:
  print("The collection exists.")
```

Documentos

Un documento en MongoDB es lo mismo que un registro en bases de datos SQL.

Insertar en la colección
Para insertar un registro, o documento como se llama en MongoDB, en una colección, usamos el método insert_one().

El primer parámetro del método insert_one() es un diccionario que contiene los nombres y valores de cada campo en el documento que desea insertar.

Documentos

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

mydict = { "name": "John", "address": "Highway 37" }

x = mycol.insert_one(mydict)
```

Insertar documentos

Devuelve el campo _id

El insert_one()método devuelve un objeto InsertOneResult, que tiene una propiedad inserted_id, que contiene la identificación del documento insertado.

```
mydict = { "name": "Peter", "address": "Lowstreet 27" }
x = mycol.insert_one(mydict)
print(x.inserted_id)
```

Insertar documentos

Si no especifica un _idcampo, MongoDB agregará uno por usted y le asignará una identificación única para cada documento.

En el ejemplo anterior no _idse especificó ningún campo, por lo que MongoDB asignó un _id único para el registro (documento).

Insertar documentos

Insertar varios documentos

Para insertar varios documentos en una colección en MongoDB, usamos el método insert_many().

El primer parámetro del método insert_many() es una lista que contiene diccionarios con los datos que desea insertar:

Insertar documentos

```
mylist = [
  { "name": "Amy", "address": "Apple st 652"},
  { "name": "Hannah", "address": "Mountain 21"},
  { "name": "Michael", "address": "Valley 345"},
  { "name": "Sandy", "address": "Ocean blvd 2"},
 { "name": "William", "address": "Central st 954"},
  { "name": "Chuck", "address": "Main Road 989"},
  { "name": "Viola", "address": "Sideway 1633"}
]

x = mycol.insert_many(mylist)

print(x.inserted_ids)
```

Insertar documentos

```
mylist = [
 { “_id": 1, "name": "John", "address": "Highway 37"},
 { "_id": 2, "name": "Peter", "address": "Lowstreet 27"},
 { "_id": 3, "name": "Amy", "address": "Apple st 652"},
 { "_id": 4, "name": "Hannah", "address": "Mountain 21"},
 { "_id": 5, "name": "Michael", "address": "Valley 345"},]

x = mycol.insert_many(mylist)

#print list of the _id values of the inserted documents:
print(x.inserted_ids)
```

Búsquedas, limites y Filtros

En MongoDB usamos los métodos find y findOne para buscar datos en una colección.

Al igual que la instrucción SELECT se usa para buscar datos en una tabla en una base de datos MySQL.

Find One

Para seleccionar datos de una colección en MongoDB, podemos usar el método find_one().

El método find_one() devuelve la primera aparición en la selección.

Búsquedas, limites y Filtros

```
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

x = mycol.find_one()

print(x)

#Devolverá el documento del ultimo documento ingresado
```

Búsquedas, limites y Filtros

Buscar todos los documentos

Para seleccionar datos de una tabla en MongoDB, también podemos usar el método find().

El método find() devuelve todas las apariciones de la selección.

El primer parámetro del método find() es un objeto de consulta. En este ejemplo usamos un objeto de consulta vacío, que selecciona todos los documentos de la colección.

Ningún parámetro en el método find() le da el mismo resultado que SELECT * en MySQL.

Búsquedas, limites y Filtros

```
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

for x in mycol.find():
  print(x)



# Devuelva todos los documentos de la colección " customers " e
# imprima cada documento
```

Búsquedas, limites y Filtros

Devolver solo algunos campos

El segundo parámetro del método find() es un objeto que describe qué campos incluir en el resultado.

Este parámetro es opcional y, si se omite, todos los campos se incluirán en el resultado.

```
for x in mycol.find({},{ "_id": 0, "name": 1, "address": 1 }):
  print(x)
```

Búsquedas, limites y Filtros

Filtrar el resultado

Al buscar documentos en una colección, puede filtrar el resultado mediante un objeto de consulta.

El primer argumento del método find() es un objeto de consulta y se utiliza para limitar la búsqueda.

```
myquery = { "address": "Park Lane 38" }
mydoc = mycol.find(myquery)
for x in mydoc:
  print(x)
```

Búsquedas, limites y Filtros

Filtrar el resultado

Al buscar documentos en una colección, puede filtrar el resultado mediante un objeto de consulta.

El primer argumento del método find() es un objeto de consulta y se utiliza para limitar la búsqueda.

```
myquery = { "address": "Park Lane 38" }
mydoc = mycol.find(myquery)
for x in mydoc:
  print(x)
```

Búsquedas, limites y Filtros

Consulta avanzada

Para realizar consultas avanzadas, puede utilizar modificadores como valores en el objeto de consulta.

Por ejemplo, para encontrar los documentos donde el campo "dirección" comienza con la letra "S" o superior (alfabéticamente), utilice el modificador mayor que {"$gt": "S"}:

```
myquery = { "address": { "$gt": "S" } }
mydoc = mycol.find(myquery)
for x in mydoc:
  print(x)
```

#Busque documentos donde la dirección comience con la letra "S" o superior

Búsquedas, limites y Filtros

Filtrar con expresiones regulares

También puede utilizar expresiones regulares como modificador.

Para buscar solo los documentos donde el campo "dirección" comienza con la letra "S", utilice la expresión regular {"$regex": "^S"}:

```
myquery = { "address": { "$regex": "^S" } }
mydoc = mycol.find(myquery)
for x in mydoc:
  print(x)
```

Búsquedas, limites y Filtros

Filtrar con expresiones regulares

También puede utilizar expresiones regulares como modificador.

Para buscar solo los documentos donde el campo "dirección" comienza con la letra "S", utilice la expresión regular {"$regex": "^S"}:

```
myquery = { "address": { "$regex": "^S" } }
mydoc = mycol.find(myquery)
for x in mydoc:
  print(x)
```

Búsquedas, limites y Filtros

Limite el resultado

Para limitar el resultado en MongoDB, usamos el método limit().

El método limit() toma un parámetro, un número que define cuántos documentos devolver.

Considere que tiene una colección de "clientes":

```
myresult = mycol.find().limit(5)

#print the result:
for x in myresult:
  print(x)
```

Ordenar

Ordenar el resultado

Utilice el método sort() para ordenar el resultado en orden ascendente o descendente.

El método sort() toma un parámetro para "nombre de campo" y un parámetro para "dirección" (ascendente es la dirección predeterminada).

```
mydoc = mycol.find().sort("name")  #.sort("name“, -1)
for x in mydoc:
  print(x)
```

```
#sort ("nombre", 1) # ascendente
#sort ("nombre", -1) # descendente
```

Ordenar

Ordenar el resultado

Utilice el método sort() para ordenar el resultado en orden ascendente o descendente.

El método sort() toma un parámetro para "nombre de campo" y un parámetro para "dirección" (ascendente es la dirección predeterminada).

```
mydoc = mycol.find().sort("name")
for x in mydoc:
  print(x)
```

Eliminar y Actualizar

Eliminar documento

Para eliminar un documento, usamos el método delete_one().

El primer parámetro del método delete_one() es un objeto de consulta que define qué documento eliminar.

Si la consulta encuentra más de un documento, solo se elimina la primera aparición.

```
myquery = { "address": "Mountain 21" }
mycol.delete_one(myquery)
```

Eliminar y Actualizar

Eliminar documento

Para eliminar un documento, usamos el método delete_one().

El primer parámetro del método delete_one() es un objeto de consulta que define qué documento eliminar.

Si la consulta encuentra más de un documento, solo se elimina la primera aparición.

```
myquery = { "address": "Mountain 21" }
mycol.delete_one(myquery)
```

Eliminar y Actualizar

Eliminar muchos documentos

Para eliminar más de un documento, use el delete_many()método.

El primer parámetro del delete_many()método es un objeto de consulta que define qué documentos eliminar.

```
myquery = { "address": {"$regex": "^S"} }
x = mycol.delete_many(myquery)
print(x.deleted_count, "documents deleted.")
```

Eliminar y Actualizar

Eliminar todos los documentos de una colección

Para eliminar todos los documentos de una colección, pase un objeto de consulta vacío al método delete_many():

```
x = mycol.delete_many({})
print(x.deleted_count, " documents deleted.")
```

Eliminar y Actualizar

Eliminar colección

Puede eliminar una tabla o colección como se llama en MongoDB, utilizando el método drop().

```
mycol.drop()
```

El método drop() devuelve verdadero si la colección se eliminó correctamente y falso si la colección no existe.

Eliminar y Actualizar

Actualizar colección

Puede actualizar un registro o documento como se llama en MongoDB, utilizando el método update_one().

El primer parámetro del método update_one() es un objeto de consulta que define qué documento actualizar.

Si la consulta encuentra más de un registro, solo se actualiza la primera aparición.

Eliminar y Actualizar

```
myquery = { "address": "Valley 345" }
newvalues = { "$set": { "address": "Canyon 123" } }

mycol.update_one(myquery, newvalues)

#print "customers" after the update:
for x in mycol.find():
  print(x)
```

Eliminar y Actualizar

Actualizar muchos

Para actualizar todos los documentos que cumplen con los criterios de la consulta, use el método update_many().

```
myquery = { "address": { "$regex": "^S" } }
newvalues = { "$set": { "name": "Minnie" } }

x = mycol.update_many(myquery, newvalues)

print(x.modified_count, "documents updated.")
```

RESUMEN DE SESIÓN

GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN