Flask ile MongoDB'ye Bağlanma

Gerekli tahmini süre: 30 dakika

Son videoda, Python'dan MongoDB'ye nasıl bağlanacağınızı ve sorgulayacağınızı öğrendiniz. Flask Python ile yazıldığından, `PyMongo` modülünü kullanarak Flask ile MongoDB'yi kullanmak oldukça kolaydır.

Hedefler

Bu okumada şunları öğreneceksiniz:

1. Flask'ta bir PyMongo istemcisi başlatmak
2. MongoDB veritabanına bağlanmak
3. Flask'tan MongoDB'deki veritabanı ve koleksiyonlarla çalışmak
4. Flask rotalarından uygun HTTP durumunu döndürmek

Dizin yapısı

PyMongo'yu Flask'ta bir veritabanıyla çalışacak şekilde nasıl ayarlayabileceğimizi görmek için örnek bir dizin yapısına bakalım.

```
.
├── __init__.py
├── app.py
├── db.py
└── venv
```

1. `__ini__.py`: bir python modülünü başlatan boş dosya
2. `app.py`: Flask rotalarını içeren ana uygulama
3. `db.py`: veritabanını başlatmak için python dosyası
4. `venv`: python sanal ortamı

Flask'tan MongoDB istemcisini Başlatma

Bu alıştırmada, bir MongoDB istemcisi başlatacaksınız. MongoDB'nin localhost'ta çalıştığını ve `mongouser` kullanıcı adı ile `password` şifresiyle bağlanabileceğinizi varsayın. Aşağıdaki komut, bu kimlik bilgilerini kullanarak bir istemci başlatacaktır:

```
client = MongoClient('mongodb://%s:%s@127.0.0.1' % ('mongouser', 'password'))
```

Veritabanını ilk kez doldurun

`db.py` dosyasında bir `PyMongo` istemcisi oluşturacaksınız. Bu daha sonra ana uygulama dosyasına aktarılabilir.

```
from pymongo import MongoClient
```

Sonraki adımda, aşağıdaki fonksiyon `MongoClient` istemcisini başlatır ve `tododb` veritabanına ait `todo` koleksiyonuna üç Todo ekler.

```
def init_db():
    client = MongoClient('mongodb://%s:%s@127.0.0.1' % ('mongouser', 'password'))
    client.tododb.todo.drop()
    client.tododb.todo.insert_many(
        [
            {"priority": "high",
            "title": "Get milk"},
            {"priority": "medium",
            "title": "Get gasoline"},
            {"priority": "low",
            "title": "Water plants"}
        ]
    )
    return client
```

1. `Line1`: İlk olarak, `mongodb://%s:%s@127.0.0.1` biçimindeki URL kullanarak yeni bir PyMongo istemcisi oluşturun. Kullanıcı adı ve şifreyi değiştireceksiniz. Bu gizli bilgi kodun içine eklenmemelidir. Bu öğrenme etkinliğinde, bilginin kolay anlaşılması için kodun içine dahil edilmiştir.
2. `Line 2`: Yukarıda açıklandığı gibi MongoDB'ye bir bağlantı oluşturur.
3. `Line 3`: `tododb` veritabanından `todo` koleksiyonunu siler. Bu, Flask sunucusu her yeniden başlatıldığında, bu örnek uygulamada veritabanının yeni bir kopyasını alacağınız anlamına gelir.
4. `Line 4`: Koleksiyona üç todo ekler.

MongoDB Veritabanı ile Etkileşim

Artık Flask uygulamasındaki `db.py` dosyasını kullanacaksınız. Şimdi koda bakalım:

```
app = Flask(__name__)
from . import db
client = db.init_db()
```

1. `Line 1`: yeni bir Flask uygulaması başlatır.
2. `Line 3`: mevcut `.` modülünden db dosyasını içe aktarır.
3. `Line 4`: `db.init_db()` yöntemi kullanarak veritabanını başlatır ve doldurur.

JSON verisi ve HTTP durumu döndür

Artık bazı rotalar yazmaya başlayabiliriz. İlk rota, `/todos` URL'si için olacak ve koleksiyondaki tüm belgeleri basitçe döndürecek.

```
@app.route("/todos")
def index():
    result = client.tododb.todo.find({})
    return json_util.dumps(list(result)), 200
```

1. `Line 1`: `/todos` URL'sini `@app.route` dekoratörü ile tanımlar.
2. `Line 2`: Bu rotayı işlemek için metodu oluşturur.
3. `Line 3`: `tododb` veritabanındaki `todo` koleksiyonundaki tüm belgeleri almak için `PyMongo`'daki `find()` metodunu kullanır.
4. `Line 4`: Ayın döndürdüğü `bson` yapısını `json_util` modülünü kullanarak `json`'a dönüştürür. Bu modül, BSON nesnelerini JSON'a ve tersine dönüştürmek için yöntemler sağlar.
5. `Line 4`: 200 HTTP kodu ile JSON listesini döndürür.

Farklı bir HTTP kodu döndürmeniz gereken duruma bakalım. Rota, `/todos/<priority>` URL'si içindir ve verilen önceliğe sahip tüm todo'ları döndürmelidir. Öncelik **düşük**, **orta** veya **yüksek** olabilir. Eğer istemci `/todos/yüksek` adresine bir GET isteği gönderirse, metod `yüksek` önceliğine sahip tüm öğeleri döndürecektir. Eğer istemci `yüksek`, `orta` veya `düşük` dışında bir öncelik gönderirse, kod `HTTP 404 NOT FOUND` döndürecektir.

```
@app.route("/todos/<priority>")
def get_by_priority_better(priority):
    result = client.tododb.todo.find({"priority": priority})
    result_list = list(result)
    if not result or len(result_list) < 1:
        return json_util.dumps(result_list), 404
    return json_util.dumps(result_list), 200
```

1. `Line 1`: `/todos/<priority>` URL'sunu `@app.route` dekoratörü ile tanımlar.
2. `Line 2`: Bu rotayı işlemek için bir yöntem oluşturur. Yöntem, önceliği bir girdi olarak alır. Flask, dekoratörde sağlanan URL'den bunu otomatik olarak çıkarır.
3. `Line 3`: Yukarıdaki koda benzer şekilde `find()` metodunu kullanır, ancak URL'de istemci tarafından geçirilen `priority` değişkenine sahip todos'ları filtreler.
4. `Line 4`: `find` metodu bir cursor döner. Bu satır, cursor'ı bir listeye dönüştürür.
5. `Line 5`: Listenin boş olup olmadığını kontrol eder. Eğer öyleyse, kod boş liste ile birlikte 404 HTTP durumu döner.
6. `Line 8`: Listeyi 200 HTTP durumu ile istemciye geri döner. Yukarıdaki koda benzer şekilde, `bson` yapısını `json`'a dönüştürmek için `json_util` modülünü kullanırsınız.

Yukarıdaki kodda hatırlanması gereken önemli bir nokta, `find` metodundan dönen `cursor` nesnesi bir Python `list`'e dönüştürüldüğünde, cursor'ın boş hale gelmesidir. Artık ileride `list` nesnesi ile çalışmalısınız, çünkü cursor'daki öğelere erişemezsiniz.

Artık `Flask` içinde `PyMongo` kullanarak `MongoDB` veritabanı ile nasıl etkileşimde bulunacağınızı bildiğinize göre, capstone projesinde kendi başınıza `PUT`, `POST` ve `DELETE` uç noktaları oluşturabilirsiniz.

Test Etme

Yukarıdaki iki uç noktayı curl komutuyla test edelim.

1. Uygulamayı aşağıdaki komutla çalıştırabiliriz:

   ```
   flask run --reload --debugger
   ```

2. `/todos` uç noktasını test et:

   ```
   curl -i -w '\n' localhost:5000/todos
   ```

   Sonuç şöyle görünecektir:

   ```
   HTTP/1.1 200 OK
   Server: Werkzeug/2.2.2 Python/3.7.16
   Date: Fri, 10 Feb 2023 04:18:31 GMT
   Content-Type: text/html; charset=utf-8
   Content-Length: 273
   Connection: close
   [{"_id": {"$oid": "63e5c594f32480ef62a901ab"}, "priority": "high", "title": "Get milk"}, {"_id": {"$oid": "63e5c594f32480ef62a901ac"}, "priority": "medium", "title": "Get gasoline"}, {"_id": {"$oid": "63
   ```

   Biçimlendirilmiş JSON çıktısı:

   ```
   [
     {
       "_id": {
         "$oid": "63e5c88077d86b4ad7dfa072"
       },
       "priority": "high",
       "title": "Get milk"
     },
     {
       "_id": {
         "$oid": "63e5c88077d86b4ad7dfa073"
       },
       "priority": "medium",
       "title": "Get gasoline"
     },
     {
       "_id": {
         "$oid": "63e5c88077d86b4ad7dfa074"
       },
       "priority": "low",
       "title": "Water plants"
     }
   ]
   ```

3. `/todos/high` uç noktasını test et:

```
curl -i -w '\n' localhost:5000/todos/high
```

Komut şu sonucu verecektir:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Fri, 10 Feb 2023 04:19:19 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 88
Connection: close
[{"_id": {"$oid": "63e5c594f32480ef62a901ab"}, "priority": "high", "title": "Get milk"}]
```

Biçimlendirilmiş JSON çıktısı:

```
[
  {
    "_id": {
      "$oid": "63e5c88077d86b4ad7dfa072"
    },
    "priority": "high",
    "title": "Get milk"
  }
]
```

4. `/todos/low` uç noktasını test et:

```
    HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Fri, 10 Feb 2023 04:21:07 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 91
Connection: close
[{"_id": {"$oid": "63e5c62b3d484ba15fb5184e"}, "priority": "low", "title": "Water plants"}]
```

Biçimlendirilmiş JSON çıktısı:

```
[
  {
    "_id": {
      "$oid": "63e5c88077d86b4ad7dfa074"
    },
    "priority": "low",
    "title": "Water plants"
  }
]
```

Sonraki Adımlar

Artık `MongoDB` veritabanı sunucusuyla etkileşimde bulunmak için `Flask` uygulamasında `PyMongo`'yu nasıl kullanacağınızı biliyorsunuz. Şimdi, bu okumada edindiğiniz becerileri kullanarak capstone projesi için ikinci servisi yazacağınız bir sonraki modüle geçebilirsiniz.

Yazar(lar)

CF