MongoDB + PyMongo

Introduction

MongoDB is a popular NoSQL database that stores data in flexible, JSON-like documents. PyMongo is the official Python library for interacting with MongoDB databases, allowing Python applications to easily perform database operations.

Presented by:
Mark Fontenot, PhD
Northeastern University

PyMongo Overview

PyMongo simplifies interactions between Python and MongoDB instances by abstracting database operations into intuitive Python methods.

Installation

To install PyMongo:
pip install pymongo

Connecting to MongoDB

Below is an example of how to establish a connection to a local MongoDB instance using PyMongo:

```
from pymongo import MongoClient

client = MongoClient('mongodb://user_name:pw@localhost:27017')
```

Explanation:

- MongoClient is a class imported from the PyMongo library used to establish a connection to a MongoDB database.
- The connection string (mongodb://) includes a username (user_name), password (pw), hostname (localhost), and port number (27017).

Getting a Database and Collection

MongoDB organizes data into databases, and each database contains collections (similar to tables in relational databases).

```
from pymongo import MongoClient

client = MongoClient('mongodb://user_name:pw@localhost:27017')

db = client['ds4300']  # or client.ds4300
collection = db['myCollection']  # or db.myCollection
```

Explanation:

- db represents a specific database instance.
- collection is a collection within the database, where documents (records) will be stored.

Inserting a Single Document

Here's how to insert a single JSON-like document into the collection:

```
db = client['ds4300']
collection = db['myCollection']

post = {
   "author": "Mark",
   "text": "MongoDB is Cool!",
   "tags": ["mongodb", "python"]
}

post_id = collection.insert_one(post).inserted_id
print(post_id)
```

Explanation:

- insert_one() inserts a single document into the collection.
- The inserted document automatically receives a unique _id.
- The returned inserted_id is useful for referencing this document later.

Finding Documents

The following example retrieves documents matching specific criteria (e.g., movies released in the year 2000):

```
from bson.json_util import dumps

# Find all movies released in 2000
movies_2000 = db.movies.find({"year": 2000})

# Pretty-print the results
print(dumps(movies_2000, indent=2))
```

Explanation:

- find() method queries the database based on specified criteria ({"year": 2000}).
- Results are returned as a cursor that can be iterated through.
- bson.json_util.dumps helps print MongoDB cursor results in readable JSON format.

Setting Up Environment with Jupyter

To work with PyMongo in a Jupyter notebook environment:

1. Activate your Conda environment or create a virtual Python environment.
2. Install required libraries:

```
pip install pymongo
pip install jupyterlab
```

1. Download provided notebooks and unzip:

- A provided zip file (this) contains Jupyter notebooks for practice.

1. Launch Jupyter Lab:

jupyter lab

Navigate to the unzipped folder in the Jupyter interface and open the provided notebooks to begin interacting with MongoDB using PyMongo.

This guide covers basic operations for MongoDB interactions using PyMongo, clearly laying out installation, connection, document insertion, querying, and environment setup.