Advanced Programming with Python

Connecting to DBs in Flask

Pepe García jgarciah@faculty.ie.edu

Plan for today

- SQLAlchemy
- Connecting to Databases

SQLAlchemy

SQLAlchemy is a Python library to interact with SQL databases. It is included in Anaconda.

SQLAlchemy. Engine

The entry point to SQLAlchemy is the `engine`. It allows us to specify where is our database, and how do we connect to it.

SQLAlchemy. Engine

The entry point to SQLAlchemy is the `engine`. It allows us to specify where is our database, and how do we connect to it.

SQLite

The database we're going to use ourselves is called SQLite. It's a good database for testing purposes, and local storage, since all data is contained on a single file

SQLAlchemy. Engine

SQLAlchemy. Engine

In order to create an engine, we need to import the `create_engine` function.

```
from sqlalchemy import create_engine
```

SQLAlchemy. Engine

Then, we can specify how do we connect to our DB. In our case we just specify the file, but we would need to specify server, host, port, username, pass in other database engines.

```
from sqlalchemy import create_engine


engine = create_engine("sqlalchemy:///paypalme.db")
```

SQLAlchemy. Connection

With the engine, we specify how does one connect to the underlying DB, but we don't yet create the connection.

In order to create the connection, we need to call **`engine.connect()`**

SQLAlchemy. Connection

Something very important is that, as with files, the **connection** must be always closed. So we can either:

```
from sqlalchemy import create_engine

engine = create_engine("sqlalchemy:///paypalme.db")

connection = engine.connect()

# more interesting stuff

connection.close()
```

SQLAlchemy. Connection

Or we can use a **with** block! (this is what we usually do)

```
from sqlalchemy import create_engine

engine = create_engine("sqlalchemy:///paypalme.db")

with  engine.connect() as connection:

    # more interesting stuff

# connection.close() Not needed anymore, with closes it automagically
```

SQLAlchemy. Fetching data

Finally, once we have the connection, we can start executing SQL queries!

```
# ...
query = "SELECT * FROM invoices"

invoices = connection.execute(query)

for invoice in invoices
    print(invoice[0]) # Results are represented as Python tuples
```

Example

Let's explore our data!

Exercise

Exercise 1

Let's implement login in paypalme using databases!

Exercise

Exercise 2

In the private area, show the transactions for the logged user.

Exercise

Exercise 3

Make it possible to create new transactions! Create a form at the bottom of the private area that receives the data for the transaction and shop.

Additional materials

Take a look at https://github.com/app-2020/app-async-9/ (also, forgive my pandemic looks)

It contains the materials for an async session we did last year to learn more about sql in Python, so it's probably useful.