**Chapter 02**

Extending from Chapter 01, we intorduce basic Daatabase connection and session concepts in this chapter.

The pre-requisites:

* Python 2.x (this particular tutorial code is developed using 2.7.3)
* pip and virtualenv installed. (<http://www.pip-installer.org/en/latest/installing.html>, <http://docs.python-guide.org/en/latest/dev/virtualenvs/>)
* Having access to a note-pad or similar text editor to edit/create files.(recommended Sublime Text2)
* Git versioning system. (<http://git-scm.com/book/en/Getting-Started-Installing-Git>)
* MongoDB. (This is the NoSQL DB that we are going to use in this project to sore and operate on the DB. Please visit <http://docs.mongodb.org/manual/installation/> and <http://www.mongodb.org/>)

Setup the environment:

if you have followed the chapter 01's documents then there is nothing much that is needed here. But before you begin please run once

(venv)$ pip install -r requirments.txt

And also, check once that you can run mongo client

$ mongo

You do not have to worry about creating a Database or a table or anything like that now. Unlike many other dbs Mongo does not complain about all those and creates them for you as soon as your application stores something in the Database.

Start Exploring the project:

As stated in the earlier point, if you have followed the chapter 01 doc properly then the only one more directory needs to be explained.

* db: This directory contains the Database related files and all. This directory has two files \_\_init\_\_.py as you know we need this file to make it package. And settings.py where we store our DB related settings. This is a good practice to create a settings.py file instead of storing it in plain text or XML files.

Explaining db/\_\_init\_\_.py file:

This file is the entry point and our single point communication road to the Database. Everything we do with the DB will go through this file eventually. Here we are using pymongo as our library to connect and talk with mongodb (<http://api.mongodb.org/python/current/>) We start the file by importing the necessary files

from pymongo import MongoClient

from settings import MONGO\_URL, DB\_NAME, COLLECTION\_NAME

import datetime

import sys

This is followed by

try:

client = MongoClient(MONGO\_URL)

db = client[DB\_NAME]

collection = db[COLLECTION\_NAME]

except Exception, ex:

sys.exit()

Here we are creating a connection with MongoDB and storing the collection in collection object. With the help of thhis we will perform every operation in the DB.

Finally we define our only function so far that is called “add\_entry”. This function acts as the middleware for inserting an entry to the databse. Check how we are calling it from app.py -> post\_entry method. This function returns True or False based on the success of it. And we duely reflect that in the fornt end by adding a Flask flash message in our application. (check the base.html for the Jinja2 way of adding a flask message container for all the pages. And see how we are assigning it with “flash” at “post\_entry”).

There is one more thing we are introduced to, that is, session. (To know in details what is a session you can check – <http://en.wikipedia.org/wiki/Session_(computer_science>[)](http://en.wikipedia.org/wiki/Session_(computer_science))). We use Flask's session to manage session inside our application. If you see the app.py file at line 90 you will see that we are supplying a secret\_key to Flask so that it signs all the sessions with it. That measn, though user will be able to see them when they view the cookie they will not be able to change the values. In a production environment, however you should use something more random to make the key secure. After that is set we can use flask's session as a normal python dict like session['key\_name'] = 'value' and to get a value from it, my\_val = session['key\_name'] (check out, post\_name, post\_entry and add\_entry methods in app.py to see some examples of the session handling.)