

1 Utilisation de “pymysql” dans le “framework” : “flask”

- fonction pour créer puis supprimer une connexion

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
from flask import Flask, request, render_template, redirect, flash

app = Flask(__name__)
app.secret_key = 'une cle(token) : grain de sel(any random string)'

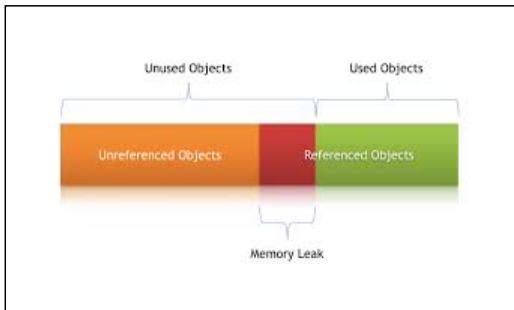
                                ## à ajouter
from flask import session, g
import pymysql.cursors

def get_db():
    if 'db' not in g:
        g.db = pymysql.connect(
            host="localhost",          # à modifier
            user="login",              # à modifier
            password="secret",         # à modifier
            database="BDD_votrelogin", # à modifier
            charset='utf8mb4',
            cursorclass=pymysql.cursors.DictCursor
        )
    return g.db

@app.teardown_appcontext
def teardown_db(exception):
    db = g.pop('db', None)
    if db is not None:
        db.close()
```

L’instruction `@app.teardown_appcontext` permet d’appeler la fonction `close_connection` lorsque que le serveur a fini de renvoyer une réponse.

Il faut recréer et fermer la connexion à chaque requête HTTP, sinon il y a un risque de [fuite de mémoire](#)



- [doc flask database - doc flask appcontext](#)
- [ancienne documentation flask -](#)
- [doc sqlalchemy](#)

2 Exemple : CRUD type_article,

Exemple de code pour ajouter/modifier/supprimer et afficher le contenu de la table “type_article” et début du CRUD sur la table “article”

- Pour afficher, ajouter, supprimer, modifier un type d’article

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
from flask import Flask, request, render_template, redirect, url_for, abort, flash, session, g

import pymysql.cursors

app = Flask(__name__)
app.secret_key = 'une cle(token) : grain de sel(any random string)'

def get_db():
    if 'db' not in g:
        g.db = pymysql.connect(
            host="localhost",          # à modifier
            user="login",              # à modifier
            password="secret",         # à modifier
            database="BDD_votrelogin", # à modifier
            charset='utf8mb4',
            cursorclass=pymysql.cursors.DictCursor
        )
    return g.db

@app.teardown_appcontext
def teardown_db(exception):
    db = g.pop('db', None)
    if db is not None:
        db.close()

@app.route('/')
def show_accueil():
    return render_template('layout.html')
```

CRUD type article

```

@app.route('/type-article/show')
def show_type_article():
    mycursor = get_db().cursor()
    sql = "SELECT * FROM type_article ORDER BY libelle"
    mycursor.execute(sql)
    types_articles = mycursor.fetchall()
    return render_template('type_article/show_type_article.html', types_articles=types_articles)

@app.route('/type-article/add', methods=['GET'])
def add_type_article():
    return render_template('type_article/add_type_article.html')

@app.route('/type-article/add', methods=['POST'])
def valid_add_type_article():
    mycursor = get_db().cursor()
    libelle = request.form.get('libelle', '')
    tuple_insert = (libelle,)
    sql = "INSERT INTO type_article(libelle) VALUES (%s);"
    mycursor.execute(sql, tuple_insert)
    get_db().commit()
    message = u'type ajouté , libellé :'+libelle
    flash(message, 'alert-success')
    return redirect('/type-article/show')

@app.route('/type-article/delete', methods=['GET'])
def delete_type_article():
    mycursor = get_db().cursor()
    id_type_article = request.args.get('id', '')
    tuple_delete = (id_type_article,)
    sql = "DELETE FROM type_article WHERE id = %s;"
    mycursor.execute(sql, tuple_delete)
    get_db().commit()
    flash(u'un type d\'article supprimé, id : ' + id_type_article)
    return redirect('/type-article/show')

```

(1) création du curseur

(2) exécution de la requête

(3) récupération de tous les enregistrements dans un tableau de dictionnaires

(4) validation de la transaction « commit »

CRUD type article

```

@app.route('/type-article/edit', methods=['GET'])
def edit_type_article():
    mycursor = get_db().cursor()
    id_type_article = request.args.get('id', '')
    sql = "SELECT id,libelle FROM type_article where id=%s"
    mycursor.execute(sql, (id_type_article,))
    type_article = mycursor.fetchone()
    return render_template('type_article/edit_type_article.html', type_article=type_article)

@app.route('/type-article/edit', methods=['POST'])
def valid_edit_type_article():
    mycursor = get_db().cursor()
    libelle = request.form['libelle']
    id_type_article = request.form.get('id', '')
    tuple_update = (libelle, id_type_article)
    sql = "UPDATE type_article SET libelle =%s WHERE id = %s;"
    mycursor.execute(sql, tuple_update)
    get_db().commit()
    flash(u'type article modifié, id : ' + id_type_article + " libelle : " + libelle)
    return redirect('/type-article/show')

```

(5) récupération du premier enregistrement dans un dictionnaire

3 Exercice

Compléter sur le même principe, le CRUD pour ajouter/modifier/supprimer et afficher les données de la table article

4 Pour les plus rapides et pour le mini projet : un filtre

on désire faire une interface avec des filtres sur les articles

Les filtres

Articles dont le nom est composé par :

Types articles

☒ Arrosage
☐ Divers
☐ Divers2
☐ Divers3
☒ Fourniture de bureau
☒ Mobilier
☐ Mobilier Jardin
☐ Outils

Prix :


Filtrer

Supprimer filtre

Les articles

Boite de rangement

3 €




Ajouter

Stock restant : 7
note : 3.0 (3 avis)

Enveloppes (50p)

2 €




Ajouter

article momentanément indisponible

Stylo noir

1 €



Ajouter

Stock restant : 10
note : 1.9 (4 avis)

filtre : exemple de formulaire

- exemple de code pour mettre en session le contenu du formulaire

https://cours-info.iut-bm.univ-fcomte.fr/upload/perso/77/S1_BDD/tp_python/S1_BDD_pymysql_tp1_flask.html

4/6

```

@app.route('/client/panier/filtre', methods=['POST'])
def client_panier_filtre():
    filter_word = request.form.get('filter_word', None)
    filter_prix_min = request.form.get('filter_prix_min', None)
    filter_prix_max = request.form.get('filter_prix_max', None)
    filter_types = request.form.getlist('filter_types', None)
    print("word:" + filter_word + str(len(filter_word)))
    if filter_word or filter_word == "":
        if len(filter_word) > 1:
            if filter_word.isalpha():
                session['filter_word'] = filter_word
            else:
                flash(u'votre Mot recherché doit uniquement être composé de lettres')
        else:
            if len(filter_word) == 1:
                flash(u'votre Mot recherché doit être composé de au moins 2 lettres')
            else:
                session.pop('filter_word', None)
    if filter_prix_min or filter_prix_max:
        if filter_prix_min.isdecimal() and filter_prix_max.isdecimal():
            if int(filter_prix_min) < int(filter_prix_max):
                session['filter_prix_min'] = filter_prix_min
                session['filter_prix_max'] = filter_prix_max
            else:
                flash(u'min < max')
        else:
            flash(u'min et max doivent être des numériques')
    if filter_types and filter_types != []:
        print("filter_types:", filter_types)
        if isinstance(filter_types, list): # type(filter_types) = list :
            check_filter_type = True
            for number_type in filter_types:
                print('test', number_type)
                if not number_type.isdecimal():
                    check_filter_type = False
            if check_filter_type:
                session['filter_types'] = filter_types
    return redirect(url_for('client_index'))

```

filtre : mettre en session le contenu du formulaire

- exemple de code pour supprimer les variables en session

```

@app.route('/client/panier/filtre/suppr', methods=['POST'])
def client_panier_suppr_filtre():
    session.pop('filter_word', None)
    session.pop('filter_prix_min', None)
    session.pop('filter_prix_max', None)
    session.pop('filter_types', None)
    return redirect(url_for('client_index'))

```

filtre : supprimer les variables en session

- exemple de code pour utiliser les variables en session pour former la requête

```

#sql = "SELECT * FROM article "
list_param = []
condition_and = ""
if "filter_word" in session or "filter_prix_min" in session or "filter_prix_max" in session or "filter_types" in session:
    sql = sql + " WHERE "
    if "filter_word" in session:
        sql = sql + " nom LIKE %s "
        recherche = "%" + session["filter_word"] + "%"
        list_param.append(recherche)
        condition_and = " AND "
    if "filter_prix_min" in session or "filter_prix_max" in session:
        sql = sql + condition_and + " prix BETWEEN %s AND %s "
        list_param.append(session["filter_prix_min"])
        list_param.append(session["filter_prix_max"])
        condition_and = " AND "
    if "filter_types" in session:
        sql = sql + condition_and + "("
        last_item = session['filter_types'][-1]
        for item in session['filter_types']:
            sql = sql + " type_article_id = %s "
            if item != last_item:
                sql = sql + " or "
            list_param.append(item)
        sql = sql + ")"
tuple_sql = tuple(list_param)

```

filtre : utiliser les variables en session pour former la requête

- exemple de code pour utiliser les variables en session dans le code HTML

```

<form method="post" action="{{ url_for('client_panier_filtre') }}" style="display: inline-block">
    <div class="form-group">
        <input name="filter_word" type="text" placeholder="Recherche" size="10" class="form-control"
            value="{% if session['filter_word'] %}{{ session['filter_word'] }}{% endif %}" >
    </div>
    {% if itemsFiltre is defined %}
    <div class="form-group">
        <h4>Types articles</h4>
        {% for itemFiltre in itemsFiltre %}
        <input type="checkbox" name="filter_types" value="{{ itemFiltre.id }}"
            {% set idItem = itemFiltre.id | string %}
            {% if session['filter_types'] %}
            {% if idItem in session['filter_types'] %} checked {% endif %} #}
            {{ ' checked ' if (idItem in session['filter_types']) else '' }}
            {% endif %}
            <a href="https://stackoverflow.com/questions/58433775/how-to-access-the-value-of-each-checkbox">
            > {{ itemFiltre.libelle }}
            <br/>
        {% endfor %}
    </div>
    {% endif %}

```

filtre : exemple de formulaire HMTL