TD 2 *API REST Flask : Quiz*

Nous souhaitons modéliser une API REST pour des questionnaires et pour modifier la manière dont nous définissons les données.

Exercice 1 *Modèle*

1.1 Faites un nouveau projet en suivant la même structure que le TD précédent.

1.2 Dans le fichier `models.py`, définir une classe `Questionnaire`, avec un identifiant (entier), et un nom. Pour gérer les identifiants utiliser une variable globale qui sera incrémentée à chaque création de nouveau questionnaire.

1.3 Toujours dans ce fichier ajouter une variable globale `questionnaires`, représentant la liste de tous les questionnaires.

1.4 Ajouter les méthodes pour récupérer tous les questionnaires, un questionnaire à partir de son id, pour créer un questionnaire et l'ajouter à la liste (à partir de son nom), et pour supprimer un questionnaire avec son id

1.5 Créer plusieurs questionnaires

Exercice 2 *Vues*

Nous allons maintenant créer les différentes routes pour gérer ces questionnaires

2.1 Importer votre liste de questionnaires dans le fichier `views.py`

2.2 Faites les routes pour obtenir tous les questionnaires, un questionnaire à partir de son id, la création d'un questionnaire, la modification, et la suppression. Avant de passer à la création d'une nouvelle route penser à la tester. Conserver dans un fichier vos requêtes curl. Penser à ajouter une méthode `to_json` dans votre classe `Questionnaire`

Exercice 3 *Modification du modèle*

Nous allons maintenant intégrer des questions aux questionnaires. Chaque questionnaire a une liste de question. Une question est définie par son numéro et son énoncé. Attention son numéro n'est pas un identifiant (il peut exister plusieurs questions n°1 si elles sont dans des questionnaires différents)

3.1 Créer la classe `Question` dans `models.py`, et ajouter une liste de questions dans la classe `Questionnaire`.

3.2 Ajouter dans la classe `Questionnaire` les méthodes permettant d'ajouter une question (à partir de son énoncé), de supprimer une question à partir de son id, et de récupérer toutes les questions.

3.3 Ajouter les nouvelles routes nécessaires pour ces questions. Attention, la route doit refléter le fait qu'une question appartient forcément a un questionnaire (donc pas de `/quiz/api/v1.0/question`. Tester chaque nouvelle route, et conserver les requêtes curl.

Exercice 4 *Base de données*

Nous souhaitons maintenant que les données soient persistantes, et donc les stocker en base de données. Nous allons utiliser SQLAlchemy.

4.1 Faites une copie de votre projet

4.2 Modifier votre modèle pour intégrer SQLAlchemy (la liste des questionnaires disparait donc)

4.3 Ajouter un fichier `commands.py` pour initier la base de données, et faire les modifications suivantes :

`commands.py`

```
from .app import app,db
from .models import create_questionnaire

@app.cli.command()
def syncdb():

    db.create_all()
    qz1 = create_questionnaire('Maths')
    ... # A completer
```

app.py

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
import os

def mkpath(p):
    return os.path.normpath(
            os.path.join(
                os.path.dirname(__file__),p)
            )
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = ('sqlite:///'+mkpath('../quiz.db'))
app.config["SQLALCHEMY_ECHO"] = True

db = SQLAlchemy(app)
```

Il est nécessaire d'intégrer le nouveau fichier à `__init__.py`

```
from .app import app
import quiz.views
import quiz.models
import quiz.commands
```

Vous avez désormais accès à une nouvelle commande dans Flask. Entrer la commande `flask syncdb` pour initier votre base de données. Un nouveau fichier est créé à la racine de votre projet contenant vos tables.

4.4 Tester vos routes pour vérifier que votre application fonctionne toujours

Exercice 5 *Extension du modèle*

Nous allons maintenant enrichir notre modèle pour intégrer deux nouveaux types de questions, les question ouvertes et les questions à choix multiples. Pour cela, faire deux nouvelles classes qui héritent de questions. Intégrer un champ réponse pour les questions ouvertes, et pour les questions fermées, deux propositions et le numéro de la bonne réponse.

5.1 Ajouter ces deux nouvelles classes à votre modèle.

5.2 Faites les modifications nécessaires à vos routes pour pouvoir créer et modifier ces nouveaux types de questions. Il n'est pas nécessaire de créer des nouvelles routes

5.3 Tester vos modifications