

FLASK- DB

לפני שניכנס למסדי נתונים של flask נסדר את הספרייה בצורה שתקל עלינו בהמשך, ולא נצטרך להכניס את כל הקוד לקובץ אחד. ראשית ניצור תיקייה חדשה שתכיל קובץ `__init__.py`, תזכורת קובץ `__init__.py` מגדיר תיקייה כ-namespace עבור כמה מודולים. לצורך הדוגמא נקרא לתיקייה `flask_example`. נעביר את החלק הראשון של הקוד (ייבוא הספריות) מסקריפט הראשי לסקריפט `__init__` כלומר בנתיים הוא יראה כך:

```
from flask import Flask, render_template
app = Flask(__name__)
```

נכניס את templates לתיקייה, נוסיף סקריפט חדש עם השם `routes.py`, ונעביר את כל שקשור לניתוב (לא כולל הפונקציית main) מהסקריפט הראשי לסקריפט הזה:

```
users = [
    {'name': 'Joe Javany',
     'email': 'joo@example.com',
     'phone': '111-1111'},
    {'name': 'Tom Pythonovitch',
     'email': 'python_is_cool@example.com',
     'phone': '222-2222'},
]

@app.route('/')
def hello_world():
    return render_template('home.html', users = users)
```

ונשנה את השם של המודול הראשי להיות `run.py` כדי שיהיה קל לזהות אותו. כדי שנוכל להשתמש במשתנה `app` צריך לייבא אותו מהתיקייה שיצרנו. בסוף הסקריפט `run.py` צריך להיראות ככה:

```
from flask_example import app

if __name__ == '__main__':
    app.run(debug = True)
```

נצטרך לייבא את הפונקציה `render_template` ואת המשתנה `app` גם לסקריפט `routes.py` כדי שנוכל להשתמש בו:

```
from flask import render_template
from flask_example import app
...
```

ולבסוף נצטרך גם לייבא את `routes` לסקריפט `__init__` אבל את זה נעשה אחרי שניצור את המשתנה `app`, כי אם נייבא אותו לפני תזרק שגיאה, שכן הקוד מנסה להפעיל את `routes`, אבל `routes` מסתמך על אובייקט שעדיין לא הוגדר (על `app`):



ד"ר סגל הלוי דוד אראל

```
from flask import Flask, render_template
app = Flask(__name__)
from flask_example import routes
```

ונריץ את run.py כדי לוודא שהכל עובד כשורה.

- FLASK-SQLALCHEMY

אנחנו נשתמש במסד נתונים sqlalchemy שכבר ראינו בשיעור הקודם. יש הרחבה מיוחדת עבור flask שמספקת הרבה מאוד ברירות מחדל שימושיות עבור השרת נתקין אותה:

```
pip install flask-sqlalchemy
```

לאחר ההתקנה נחזור לקובץ __init__ נייבא את המחלקה SQLAlchemy מהספרייה flask-sqlalchemy. עכשיו צריך להגדיר את המסד נתונים בו משתמשים, בשביל להגדיר אותו צריך להשתמש באובייקט של flask.config האובייקט מגדיר את הקונפיגורציה של השרת בזמן היצירה שלו, אנחנו נשמש בו כדי להגדיר את SQLAlchemy_DATABASE_URI שהוא קונפיגורציה הקישוריות של מסד הנתונים של השרת. האובייקט עובד בצורה של מילון, הוא מקבל כמפתח את שם הקונפיגורציה והערך שלו, במקרה זה לפחות, הוא מחרחט עם ה-URI למסד. בינתיים נשתמש ב-sqlite בתור המסד, לכן בתחילת ה-uri נכתוב 'sqlite:///'. השלוש קווים מסמנים path יחסי של המסד נתונים מהקובץ הנוכחי (relative path), ואם המסד נתונים לא מוגדר בכלל הוא יוגדר באותה תיקייה של הסקריפט; אחר כך נוסיף את שם המסד נתונים, נניח קוראים site.db. ולבסוף ניצור אינסטנס של המסד נתונים עם המחלקה שייבאנו:

```
from flask-sqlalchemy import SQLAlchemy
from flask import Flask, render_template
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)

from flask_example import routes
```

מה שמעולה ב-sqlalchemy שאנחנו יכולים לייצג את טבלאות המסד נתונים שלנו כמחלקות מונחות עצמים שנקראות מודלים (models), כל מודל הוא מחלקה שיוורשת מאובייקט שנקרא Model ששייך למסד נתונים (db.Model), כל עמודה מוגדרת כמחלקה שנקראת Column וחייב להכיל את טיפוס הנתונים שאמור לייצג אותה, למשל אם העמודה מייצגת גיל נצטרך להגדיר אותה כאינטגרל (db.Integer), ושמות כמחרחט (db.String()) ואפשר להגדיר גם גודל מקסימלי, למשל 20 תווים: (db.String(20)) וכו'. פרמטרים נוספים שניתן להגדיר הם אם העמודה היא primary key כלומר כזאת שניתן להשתמש בה כדי להסיק על שאר הערכים מטבלאות חיצוניות, למשל תעודת זהות היא primary-key כי היא מספר ייחודי לכל בן אדם, ואם יש לי אותו אני אז יכול לקבל ספציפית את הערכים של האדם הזה, לעומת שם שיכול להיות שיש שני אנשים עם אותו שם, ואז אני אקבל את שניהם ולא דווקא את הבן אדם הספציפי שרציתי לקבל; עוד משהו שניתן להגדיר זה האם העמודה ייחודית, כלומר כזאת שאין לה חזרות, האם היא nullable כלומר ניתן לתת לה ערך null, וכו'. ניצור שתי טבלאות - אחת שתשמש להכלה של המשתמשים, ואחת להכלה של הפוסטים שלהם. הטבלה של המשתמשים תכלול - איזשהו primary-key, מייל, טלפון ותמונת פרופיל. הטבלה של הפוסטים תכלול - primary-key כלשהו, כותרת, תאריך ותוכן

כמו כן צריך להגדיר איזשהו קשר בין משתמש לרשימה של פוסטים שלו, לכן ניצור משתנה חדש מקבל את הערך החוזר מהפונקציה relationship(), הפונקציה מקבלת שם של מודל אחר, אצלנו היא תהיה Post, ומגדירה קשר ישיר בין



ד"ר סגל הלוי דוד אראל

המחלקה הזאת למחלקה השנייה. יש לה פרמטר `backref` שמוסיף לאובייקט של המודל `Post` עוד עמודה שמייצגת את המשתמש שכתב את הפוסט. עוד פרמטר שנשתמש הוא `lazy` שמגדיר שכאשר מטעינים את הנתונים מהמסד, הם לא ייטענו בבת אחת (in one go).

שימוש לב שהפוסטים הם `relationship` ולא עמודה של המודל `User`.

בנוסף נרצה להוסיף מפתח זר לפוסטים שייצג את `id` של הכותב שלו. בשביל להגדיר מפתח זר נצטרך להשתמש באובייקט `ForeignKey` שמקבל את הטבלה והעמודה בה הוא משתמש, הטבלה במקרה זה היא כשם המודל רק באותיות קטנות, למשל עבור `User.id` נכתוב `user.id`.

כדי שלא נתבלבל נשים את הקוד בסקריפט חדש שנקרא `models.py` ושלא נשכח לייבא אליו גם את `db` מהמודול `flask-example`.

```
from flask_example import db
from datetime import datetime

class User(db.Model):
    id = db.Column(db.Integer , primary_key= True)
    username = db.Column(db.String(20) , unique=True, nullable = False)
    email = db.Column(db.String(20) , unique=True, nullable = False)
    phone = db.Column(db.Integer , unique = True)
    profile_img = db.Column(db.String(20), nullable= False , default='default.jpg')
    posts = db.relationship('Post' , backref = 'author' , lazy = True)

    def __repr__(self):
        return f'User({self.username!r} , {self.email!r}, {self.phone!r}, {self.profile_img!r})'

class Post(db.Model):
    id = db.Column(db.Integer , primary_key= True)
    title = db.Column(db.String(100), nullable =False)
    date_posted = db.Column(db.DateTime, nullable = False , default = datetime.utcnow)
    content = db.Column( db.Text, nullable = False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable =False)

    def __repr__(self):
        return f'Post({self.title} , {self.date_posted})'
```

בואו נבחן את המסד נתונים ונראה אם הוא עובד כמו שצריך.

נפתח את המצב האינטרקטיבי של פייתון ונייבא אליו את `db` ממודול `flask_example`. אם ביצענו את התהליך כמו שצריך בחלק הראשון אנחנו אמורים לא לקבל שגיאות, אבל לקבל אזהרה מ-`SQLALCHEMY`. אין מה לדאוג מזה זה סה"כ מידע. אח"כ נייבא את המחלקות `User` ו-`Post` מ-`flask_example.models`.

כדי ליצור את המסד נתונים נשתמש בפקודה `db.create_all()` זה אמור ליצור את כל המתודות של המסד. אם קיבלנו שגיאה זה אמור ליצור לנו מסד חדש עם השם `site.db` בתיקייה `flask_example`:

```
>>> from flask_example import db
.../__init__.py:833: FSADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant
overhead and will be disabled by default in the future. Set it to True or False to suppress
this warning.
  warnings.warn(FSADeprecationWarning(
>>> from flask_example.models import User , Post
```



ד"ר סגל הלוי דוד אראל

```
>>> db.create_all()
>>>
```

כדי לבחון את המערכת ניצור שני user ונכניס להם כל מיני נתונים, ההגדרה של ה-id תיווצר אוטומטית ולא נצטרך להגדיר אותה ביצירת המשתמשים. כרגע נשמש בתמונה הדיפולטית. אחרי שהגדרנו כל משתמש צריך להשתמש בפונקציה db.session.add() והארגומנט אמור להיות שם המשתמש שמוסיפים. ולבסוף צריך לעשות גם session.commit() לפעולות שביצענו:

```
>>> user1 = User(username = 'Tom Pythonovitch', email = 'Tom@mail.com', phone = 55555555)
>>> db.session.add(user1)
>>> user2 = User(username = 'joe javany', email = 'Joe@mail.com', phone = 44444444)
>>> db.session.add(user2)
>>> db.session.commit()
```

ועכשיו הנתונים אמורים להיות במסד-הנתונים. כדי לבדוק אם הנתונים באמת נמצאים במסד נוכל להשתמש בquery פשוט על המודול שנבחר. למשל כדי לראות את כל הנתונים של הטבלה User נשתמש ב-query.all() על הטבלה:

```
>>> User.query.all()
[User('Tom Pythonovitch' , 'Tom@mail.com', 55555555, 'default.jpg'), User('joe javany'
, 'Joe@mail.com', 44444444, 'default.jpg')]
```

יש כמה שאילתות שניתן לבצע על המסד ולא ניכנס לפרטים, כרגע מה שבעיקר חשוב לנו הוא לדעת איך לבקש משתמש ספציפי לפי איזה ערך של הטבלה, למשל לבקש את המשתמש שקוראים לו Tom Pythonovitch. בשביל זה נשתמש ב-filter.query() ובשאלתא נכניס לפי איזה פרמטר נרצה להשתמש. נוכל להשתמש ב-dll() בשביל לקבל את כל המשתמשים שעונים על השאלתא וב-first() בשביל לקבל את הראשון וכו':

```
>>> User.query.filter_by(username = 'Tom Pythonovitch').all()
[User('Tom Pythonovitch' , 'Tom@mail.com', 55555555, 'default.jpg')]
```

נשמור את המשתמש במשתנה כדי לגשת לשדות שפציפים שלו למשל id :

```
>>> user_1 = User.query.filter_by(username = 'Tom Pythonovitch').first()
>>> user_1.username
'Tom Pythonovitch'
>>> user_1.id
1
```

באותו אופן אפשר להשתמש בפונקציה query.get() ולהכניס לה את מספר ה-id של המשתמש ולקבל את התוצאה.

```
>>> user_1 = User.query.get(1)
>>> user_1.id
1
```

כרגע למשתמש אין שום פוסט, אם ננסה לגשת ל-user_1.post נקבל רשימה ריקה. בואו נוסיף שני פוסטים חדשים, ונראה איך הם מחוברים למשתמש user_1 שה-id שלו הוא 1:

```
>>> user_1.posts
[]
>>> post1 = Post(title = 'First', content = '...' , user_id = user_1.id)
>>> post2 = Post(title = 'Second', content = '...' , user_id = user_1.id)
>>> db.session.add(post1)
>>> db.session.add(post2)
>>> db.session.commit()
>>> Post.query.all()
```

ד"ר סגל הלוי דוד אראל

```
[Post('First', '2021-02-13 23:17:12.928932'), Post('Second', '2021-02-13 23:17:12.930933')]  
>>> user_1.posts  
[Post('First', '2021-02-13 23:17:12.928932'), Post('Second', '2021-02-13 23:17:12.930933')]
```

כדי למחוק את המסד נתונים נשתמש בפונקציה `drop_all()`, ואז כדי ליצור את המסד נתונים שוב נשתמש פעם נוספת בפונקציה `:create_all()`

```
>>> db.drop_all()  
>>> db.create_all()
```