05 – Packages

သေးငယ်တဲ့ web applications တွေကို script file တစ်ခုတည်းနဲ့ store လုပ်ထားတာ အဆင်ပြေသင့်လျော်ပေမဲ့ ကောင်းမွန်တဲ့ အစီအစဉ်တော့မဟုတ်ပေဘူး။ ကြီးထွား ရှုတ်ထွေးလာတာနဲ့အမျှ ကြီးမားလာတဲ့ application တွေအတွက် source file တစ်ခုတည်းနဲ့ အလုပ်လုပ်တယ်ဆိုတာ အခက်အခဲ ပြဿနာတွေ ဖြစ်ပေါ်လာစေပါတယ်။

အခြားမြောက်မြားလှစွာသော web framework တွေနဲ့ မတူတဲ့ Flask မှာ ကြီးမားတဲ့ projects တွေအတွက် သီးခြားဖွဲ့စည်းမှုတွေ ပြဋ္ဌာန်းမထားတဲ့အတွက် developer ကသာလျှင် application တစ်ခုလုံးရဲ့ structure ကို လမ်းကြောင်းချပေးရဖို့ ဖြစ်လာပါတယ်။အဲဒီအတွက် ဖြစ်တန်ရာသော လမ်းကြောင်းတစ်ခုနဲ့ application တစ်ခုကို packages and modules တွေထဲမှာ စုဖွဲ့ဖော်ပြကြရအောင်။

အခု FlaskBlog application ရဲ့ Package structure ကို ပုံစံချကြည့်ပါမယ်။

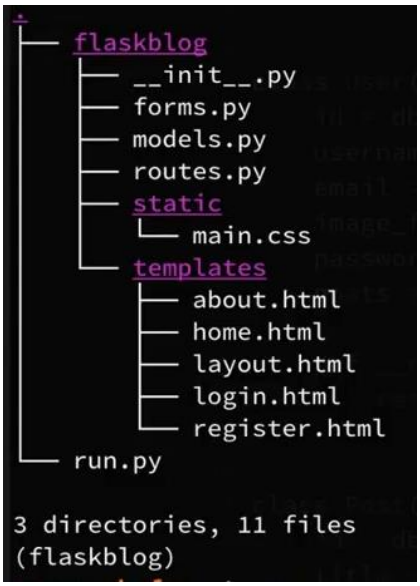
Structure of flaskblog

_____

– flaskblog

1. __init__
2. forms.py
3. models.py
4. routes.py
5. static\
6. templates\

– run.py



project အစမှာ ဖန်တီးခဲ့တဲ့ flaskblog folder က project folder ပါ ။ အခု အဲဒီ folder အောက်မှာ application package folder တစ်ခုကို flaskblog အမည်နဲ့ပဲ တည်ဆောက်ပါမယ်။ application package တိုင်းမှာ __init__.py ဆိုတဲ့ application package constructor ပါရပါမယ်။ ဒီ constructor မှာ လက်ရှိ application က အသုံးပြုနေတဲ့ Flask extensions များကို import လုပ်ထားရပါမယ်။ နောက် template files တွေအတွက် templates folder နဲ့ static files တွေအတွက် static folder

ရှိပါမယ်။ database models တွေအတွက် models.py၊ form class တွေအတွက် forms.py၊ နဲ့ routing patterns တွေအတွက် routes.py အစရှိတဲ့ module files တွေလဲ အသီးသီးခွဲထုတ်ပါမယ်။

ယခင်အသုံးပြုလာတဲ့ application ရဲ့ main script ဖြစ်တဲ့ flaskblog.py ကိုတော့ run.py အမည်နဲ့ refactoring ပြန်လုပ်ပေးပါမယ်။ အဲဒီအတွက် environment variable တွေကိုတော့ လိုအပ်ရင် setting ပြန်ပြင်ပေးရပါလိမ့်မယ်။ လိုတာတွေ ပြောင်းလဲရေးသားပြီးရင်တော့ application ကို ပြန် run ကြည့်ရအောင်ပါ။

```
# run flask project
(venv)python run.py
```

run.py

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

__init__.py

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)
from flaskblog import routes
```

forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, BooleanField
from wtforms.validators import DataRequired, Length, Email, EqualTo

class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=2,
max=20)])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Sign Up')

class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')
```

models.py

```
from datetime import datetime
from flaskblog import db

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(120), unique=True, nullable=False)
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    password = db.Column(db.String(60), nullable=False)
    posts = db.relationship('Post', backref='author', lazy=True)

    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image_file}')"

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}')"
```

routes.py

```
from flask import render_template, url_for, flash, redirect
from flaskblog import app
from flaskblog.forms import RegistrationForm, LoginForm
from flaskblog.models import User, Post

posts = [
    {
        ...
    },
    {
            ...
    },
]

@app.route("/")
@app.route("/home")
def home():
    return render_template('home.html', posts=posts)

@app.route("/about")
def about():
    return render_template('about.html', title='About')

@app.route("/register", methods=['GET', 'POST'])
```

```
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        flash(f'Account created for {form.username.data}!', 'success')
        return redirect(url_for('home'))
    return render_template('register.html', title='Register', form=form)


@app.route("/login", methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        if form.email.data == 'admin@blog.com' and form.password.data == 'password':
            flash('You have been logged in!', 'success')
            return redirect(url_for('home'))
        else:
            flash('Login Unsuccessful. Please check username and password', 'danger')
    return render_template('login.html', title='Login', form=form)
```

Teminal ကနေ Python shell ကိုသွားပြီးလဲ စမ်းသုံးကြည့်ပါအုံးမယ်။

```
# Change the working directory
>>> from flaskblog import db

# Import User and Post models
>>> from flaskblog.models import User, Post

>>> db.create_all()
```

```
>>> User.query.all()
```