전북대 소중해유(You)

프로젝트:
Python과 Flask를 활용한 간단한 백엔드 구축

전북대학교 JEONBUK NATIONAL UNIVERSITY
전북대학교 SOFTWARE중심대학사업단
SW중심대학
과학기술정보통신부
IITP 정보통신기획평가원

목차

Python 백엔드 구축

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, Flask!"

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

```

Python 백엔드 구축

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html', name="Flask")

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

```

Python 백엔드 구축

```python
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('form.html')

@app.route('/greet', methods=['POST'])
def greet():
    name = request.form['name']
    return f"Hello, {name}!"

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

Python 백엔드 구축

```python
from flask import Flask, request, jsonify, abort
from models import db, Student

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db.init_app(app)


def create_tables():
    db.create_all()

app.before_request(create_tables)


```

Python 백엔드 구축

```python
@app.route('/students', methods=['GET'])
def get_students():
    students = Student.query.all()
    return jsonify([student.to_dict() for student in students])


@app.route('/students/<int:student_id>', methods=['GET'])
def get_student(student_id):
    student = Student.query.get_or_404(student_id)
    return jsonify(student.to_dict())


```

Python 백엔드 구

```python
@app.route('/students', methods=['POST'])
def create_student():
    print(request.json)

    if not request.json or not 'name' in request.json:
        abort(400)


    student = Student(
        name=request.json['name'],
        age=request.json.get('age', ""),
        grade=request.json.get('grade', "")
    )

    try:
        db.session.add(student)
        db.session.commit()

    except Exception as e:
        print(e)
        return jsonify({'error': 'Student already exists'}), 409

    return jsonify(student.to_dict()), 201


```

Python 백엔드 구축

```python
@app.route('/students/<int:student_id>', methods=['PUT'])
def update_student(student_id):

    try:
        student = Student.query.get_or_404(student_id)
        if not request.json:
            abort(400)

        student.name = request.json.get('name', student.name)
        student.age = request.json.get('age', student.age)
        student.grade = request.json.get('grade', student.grade)

        db.session.commit()
    except Exception as e:
        print(e)
        return jsonify({'error': 'Student does not exist'}), 404

    return jsonify(student.to_dict()), 200


```

Python 백엔드 구축

```python
@app.route('/students/<int:student_id>', methods=['DELETE'])
def delete_student(student_id):

    try:
        student = Student.query.get_or_404(student_id)
        db.session.delete(student)
        db.session.commit()
    except Exception as e:
        print(e)
        return jsonify({'error': 'Student does not exist'}), 404


    return jsonify({'result': True, 'message': f'Student {student_id} deleted'}), 200


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

```

Python 백엔드 구축

```python
from flask import Flask, render_template, url_for, flash, redirect, request
from models2 import db, User, Post
from forms import RegistrationForm, LoginForm, PostForm
from flask_login import login_user, current_user, logout_user, login_required, LoginManager

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blog.db'
db.init_app(app)

login_manager = LoginManager(app)
login_manager.login_view = 'login'

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

def create_tables():
    db.create_all()

app.before_request(create_tables)
```

Python 백엔드 구축

```python
@app.route('/')
@app.route('/home')
def home():
    posts = Post.query.all()
    return render_template('home.html', posts=posts)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        user = User(username=form.username.data, email=form.email.data, password=form.password.data)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', form=form)
```

Python 백엔드 구축

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and user.password == form.password.data:
            login_user(user)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('home'))
        else:
            flash('Login Unsuccessful. Please check email and password', 'danger')
    return render_template('login.html', form=form)

@app.route('/logout')
def logout():
    logout_user()
    return redirect(url_for('home'))
```

Python 백엔드 구축

```python
@app.route('/post/new', methods=['GET', 'POST'])
@login_required
def new_post():
    form = PostForm()
    if form.validate_on_submit():
        post = Post(title=form.title.data, content=form.content.data, author=current_user)
        db.session.add(post)
        db.session.commit()
        flash('Your post has been created!', 'success')
        return redirect(url_for('home'))
    return render_template('new_post.html', form=form)

@app.route('/post/<int:post_id>')
def post(post_id):
    post = Post.query.get_or_404(post_id)
    return render_template('post.html', title=post.title, post=post)


```

```python
@app.route('/post/<int:post_id>/update', methods=['GET', 'POST'])
@login_required
def update_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    form = PostForm()
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        db.session.commit()
        flash('Your post has been updated!', 'success')
        return redirect(url_for('post', post_id=post.id))
    elif request.method == 'GET':
        form.title.data = post.title
        form.content.data = post.content
    return render_template('edit_post.html', form=form)

@app.route('/post/<int:post_id>/delete', methods=['POST'])
@login_required
def delete_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    db.session.delete(post)
    db.session.commit()
    flash('Your post has been deleted!', 'success')
    return redirect(url_for('home'))

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

감사합니다.

- 본 온라인 콘텐츠는 2024년도 과학기술 정보통신부 및 정보통신기획평가원의 'SW중심대학사업' 지원을 받아 제작되었습니다.
- 본 결과물의 내용을 전재할 수 없으며, 인용(재사용)할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원이 지원한 'SW중심대학'의 결과물이라는 출처를 밝혀야 합니다.

전북대학교 JEONBUK NATIONAL UNIVERSITY

전북대학교 SOFTWARE중심대학사업단

SW중심대학

과학기술정보통신부

IITP 정보통신기획평가원