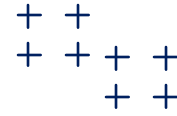
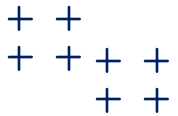


전북대 소중해유(You)



프로젝트:
Python과 Flask를 활용한 간단한 백엔드 구축



전북대학교
SOFTWARE중심대학사업단

SW중심대학




정보통신기획평가원



목차

1 / 백엔드 구축 및 설명



```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello():
7     return "Hello, Flask!"
8
9 if __name__ == '__main__':
10     app.run(debug=True)
11
```



```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def index():
7     return render_template('index.html', name="Flask")
8
9 if __name__ == '__main__':
10     app.run(debug=True)
11
```



```
1  from flask import Flask, render_template, request
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def index():
7      return render_template('form.html')
8
9  @app.route('/greet', methods=['POST'])
10 def greet():
11     name = request.form['name']
12     return f"Hello, {name}!"
13
14 if __name__ == '__main__':
15     app.run(debug=True)
16
```



```
1 from flask import Flask, request, jsonify, abort
2 from models import db, Student
3
4 app = Flask(__name__)
5 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.db'
6 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
7
8 db.init_app(app)
9
10
11 def create_tables():
12     db.create_all()
13
14 app.before_request(create_tables)
15
16
```



```
1 @app.route('/students', methods=['GET'])
2 def get_students():
3     students = Student.query.all()
4     return jsonify([student.to_dict() for student in students])
5
6
7 @app.route('/students/<int:student_id>', methods=['GET'])
8 def get_student(student_id):
9     student = Student.query.get_or_404(student_id)
10    return jsonify(student.to_dict())
11
12
```

```
1 @app.route('/students', methods=['POST'])
2 def create_student():
3     print(request.json)
4
5     if not request.json or not 'name' in request.json:
6         abort(400)
7
8
9     student = Student(
10         name=request.json['name'],
11         age=request.json.get('age', ''),
12         grade=request.json.get('grade', '')
13     )
14
15     try:
16         db.session.add(student)
17         db.session.commit()
18
19     except Exception as e:
20         print(e)
21         return jsonify({'error': 'Student already exists'}), 409
22
23     return jsonify(student.to_dict()), 201
24
25
```



```
1 @app.route('/students/<int:student_id>', methods=['PUT'])
2 def update_student(student_id):
3
4     try:
5         student = Student.query.get_or_404(student_id)
6         if not request.json:
7             abort(400)
8
9         student.name = request.json.get('name', student.name)
10        student.age = request.json.get('age', student.age)
11        student.grade = request.json.get('grade', student.grade)
12
13        db.session.commit()
14    except Exception as e:
15        print(e)
16        return jsonify({'error': 'Student does not exist'}), 404
17
18    return jsonify(student.to_dict()), 200
19
20
```

```
1 @app.route('/students/<int:student_id>', methods=['DELETE'])
2 def delete_student(student_id):
3
4     try:
5         student = Student.query.get_or_404(student_id)
6         db.session.delete(student)
7         db.session.commit()
8     except Exception as e:
9         print(e)
10        return jsonify({'error': 'Student does not exist'}), 404
11
12
13    return jsonify({'result': True, 'message': f'Student {student_id} deleted'}), 200
14
15
16 if __name__ == '__main__':
17     app.run(debug=True)
18
```

```
1 from flask import Flask, render_template, url_for, flash, redirect, request
2 from models2 import db, User, Post
3 from forms import RegistrationForm, LoginForm, PostForm
4 from flask_login import login_user, current_user, logout_user, login_required, LoginManager
5
6 app = Flask(__name__)
7 app.config['SECRET_KEY'] = 'your_secret_key'
8 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blog.db'
9 db.init_app(app)
10
11 login_manager = LoginManager(app)
12 login_manager.login_view = 'login'
13
14 @login_manager.user_loader
15 def load_user(user_id):
16     return User.query.get(int(user_id))
17
18 def create_tables():
19     db.create_all()
20
21 app.before_request(create_tables)
```

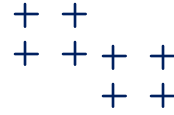
```
1 @app.route('/')
2 @app.route('/home')
3 def home():
4     posts = Post.query.all()
5     return render_template('home.html', posts=posts)
6
7 @app.route('/register', methods=['GET', 'POST'])
8 def register():
9     if current_user.is_authenticated:
10         return redirect(url_for('home'))
11     form = RegistrationForm()
12     if form.validate_on_submit():
13         user = User(username=form.username.data, email=form.email.data, password=form.password.data)
14         db.session.add(user)
15         db.session.commit()
16         flash('Your account has been created! You are now able to log in', 'success')
17         return redirect(url_for('login'))
18     return render_template('register.html', form=form)
19
20
```

```
1 @app.route('/login', methods=['GET', 'POST'])
2 def login():
3     if current_user.is_authenticated:
4         return redirect(url_for('home'))
5     form = LoginForm()
6     if form.validate_on_submit():
7         user = User.query.filter_by(email=form.email.data).first()
8         if user and user.password == form.password.data:
9             login_user(user)
10            next_page = request.args.get('next')
11            return redirect(next_page) if next_page else redirect(url_for('home'))
12        else:
13            flash('Login Unsuccessful. Please check email and password', 'danger')
14    return render_template('login.html', form=form)
15
16 @app.route('/logout')
17 def logout():
18     logout_user()
19     return redirect(url_for('home'))
20
21
```

```
1 @app.route('/post/new', methods=['GET', 'POST'])
2 @login_required
3 def new_post():
4     form = PostForm()
5     if form.validate_on_submit():
6         post = Post(title=form.title.data, content=form.content.data, author=current_user)
7         db.session.add(post)
8         db.session.commit()
9         flash('Your post has been created!', 'success')
10        return redirect(url_for('home'))
11    return render_template('new_post.html', form=form)
12
13 @app.route('/post/<int:post_id>')
14 def post(post_id):
15     post = Post.query.get_or_404(post_id)
16     return render_template('post.html', title=post.title, post=post)
17
18
```

Python 백엔드 구축

```
1 @app.route('/post/<int:post_id>/update', methods=['GET', 'POST'])
2 @login_required
3 def update_post(post_id):
4     post = Post.query.get_or_404(post_id)
5     if post.author != current_user:
6         abort(403)
7     form = PostForm()
8     if form.validate_on_submit():
9         post.title = form.title.data
10        post.content = form.content.data
11        db.session.commit()
12        flash('Your post has been updated!', 'success')
13        return redirect(url_for('post', post_id=post.id))
14    elif request.method == 'GET':
15        form.title.data = post.title
16        form.content.data = post.content
17    return render_template('edit_post.html', form=form)
18
19 @app.route('/post/<int:post_id>/delete', methods=['POST'])
20 @login_required
21 def delete_post(post_id):
22     post = Post.query.get_or_404(post_id)
23     if post.author != current_user:
24         abort(403)
25     db.session.delete(post)
26     db.session.commit()
27     flash('Your post has been deleted!', 'success')
28     return redirect(url_for('home'))
29
30 if __name__ == '__main__':
31     app.run(debug=True)
32
```

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.

