2025-05-17

2025254008 곽소영

IoT 플랫폼과 IoT 장치 설계
은행 웹서버 개발

충북대학교 산업인공지능학과

CONTENTS

목차.

01
프로젝트 기능 및 폴더 구조

02
코드 설명

프로젝트 기능

- **로그인 기능**
  - 사용자 ID/비밀번호로 인증
  - 대시보드
  - 로그인 후 접근 가능
- **입출금 기능**
  - 잔액 및 입출금 기능 제공
  - 입금 / 출금
  - 금액 입력 후 잔액 반영 및 메세지 출력
- **잔액 유지**
  - json 파일로 잔액 저장
  - 서버 재시작 후 복원
- **로그아웃**
  - 세션 종료 및 로그인 페이지로 이동
- **404 에러 헨들링**
  - 존재하지않는 url 접근 시 사용자 정의 404 페이지 표시

폴더 구조

```
project_bank/
├── app.py
├── balance_data.json
└── templates/
    ├── login.html
    ├── dashboard.html
    └── 404.html
```

코드 설명 app.py

```python
app = Flask(__name__)
app.secret_key = 'ChungbuckUniv'
```

- Flask() 객체 생성
- app.secret_key는 세션을 암호화하기 위한 key

```python
BALANCE_FILE = 'balance_data.json'
```

- 잔액 데이터는 json 파일로 저장 및 관리

```python
# 사용자 정보 (패스워드만 메모리에 유지)
users = {
    'user1': {
        'password': generate_password_hash('pass123')
    }
}
```

- 사용자 정보는 메모리에 저장
- 패스워드는 generate_password_hash로 암호화 처리

코드 설명 app.py

```python
# 초기 잔액 생성 함수
def load_balance(userid):
    balances = {}
    if os.path.exists(BALANCE_FILE):
        with open(BALANCE_FILE, 'r') as f:
            balances = json.load(f)

    # 사용자 잔액이 없으면 기본 잔액으로 생성
    if userid not in balances:
        balances[userid] = 1000
        save_balance(balances)

    return balances[userid]


# 잔액 저장 함수
def save_balance(data):
    with open(BALANCE_FILE, 'w') as f:
        json.dump(data, f)
```

- load_balance() : json 파일이 있으면 읽고, 없으면 기본값 반환
- save_balance(): 현재 잔액을 json 파일로 관리

코드 설명 app.py

```python
# 로그인 페이지
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        userid = request.form['userid']
        password = request.form['password']
        user = users.get(userid)

        if user and check_password_hash(user['password'], password):
            session['user'] = userid
            return redirect(url_for('dashboard'))
        else:
            flash('아이디 또는 비밀번호를 확인해주세요.')
    return render_template('login.html')
```

- GET 요청시 : 로그인 폼으로 렌더링
- POST 요청시 : 폼에서 받은 아이디/비밀번호 확인
  - check_password_hash()로 암호화된 패스워드 비교
- 인증 성공 -> 세션 저장
- 인증 실패 -> flash() 메세지 사용

```python
# 로그아웃
@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('login'))
```

- 세션에서 user 키를 삭제하고 로그인 페이지로 리다이렉트

코드 설명 app.py

```python
def dashboard():
    if 'user' not in session:
        return redirect(url_for('login'))

    userid = session['user']
    balances = {}
    user_balance = load_balance(userid)
    message = ""

    if os.path.exists(BALANCE_FILE):
        with open(BALANCE_FILE, 'r') as f:
            balances = json.load(f)

    if request.method == 'POST':
        action = request.form['action']
        amount = int(request.form['amount'])

        if action == 'deposit':
            user_balance += amount
            message = f"{amount}원이 입금되었습니다."
        elif action == 'withdraw':
            if amount > user_balance:
                message = "잔액이 부족합니다."
            else:
                user_balance -= amount
                message = f"{amount}원이 출금되었습니다."

        # 변경된 잔액 저장
        balances[userid] = user_balance
        save_balance(balances)

    return render_template('dashboard.html', user=userid, balance=user_balance, message=message)
```

- 로그인이 되지 않았으면 강제 로그인 페이지 리다이렉트
- 사용자의 잔액을 불러와 보여줌
- POST 요청 시 입금 또는 출금 처리
  - 출금 시 잔액 부족 여부 확인
  - 처리 후 변경된 금액 저장
- 결과 메세지를 템플릿에 전달

app.py | dashboard.html

templates > dashboard.html > html > body > a

```html
<!DOCTYPE html>
<html>
  <head>
    <title>대시보드</title>
  </head>
  <body>
    <h2>{{ user }}님의 계좌</h2>
    <p>현재 잔액: {{ balance }}원</p>
    {% if message %}
    <p><strong>{{ message }}</strong></p>
    {% endif %}
    <form method="POST">
      <label>금액:</label><br />
      <input type="number" name="amount" required /><br />
      <button type="submit" name="action" value="deposit">입금</button>
      <button type="submit" name="action" value="withdraw">출금</button>
    </form>
    <br />
    <a href="{{ url_for('logout') }}">로그아웃</a>
  </body>
</html>

```

코드 설명 app.py

```python
@app.errorhandler(404)
def page_not_found(e):
    return render_template("error.html"), 404
```

- 정의되지 않은 url 접근 시 error.html을 보여줌

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

- 개발 모드로 Flask 서버를 실행