플라스크 웹 프로그램

Flash 메시지

유효성 검증(밸리데이션) 추가하기

- Flash 메시지는 동작 실행 후에 간단한 메시지를 표시하는 기능
- 완료 시나 오류 발생 시 등 일시적으로 메시지를 표시할 때 이용
- 문의 폼 화면에 유효성 검증 (입력 체크 처리) 을 추가
- 입력 체크 시 오류가 있으면 Flash 메시지를 사용하여 한 번만 오류 정보를 표시
- Flash 메시지는 flash 함수를 사용하여 설정하고, 템플릿에서 get_flashed_messages 함수를 사용
- Flash 메시지를 이용하려면 세션이 필요하므로 config의 SECRET_KEY
- 를 설정

Flash 메시지

유효성 검증(밸리데이션) 추가하기

- config는 앱을 이용하는 데 필요한 설정
- config에 값을 설정하려면

```
app.config["config_key"] = config_value
```

Flash 메시지

SECRET_KEY 설정하기

- 세션을 사용하려면 세션 정보 보안을 위해 비밀 키 (SECRET_KEY)를 설정
- 비밀 키는 랜덤한 값으로 해야 함
- C:₩flaskbook₩apps₩minimalapp₩app.py 수정

app.py - Windows 메모장

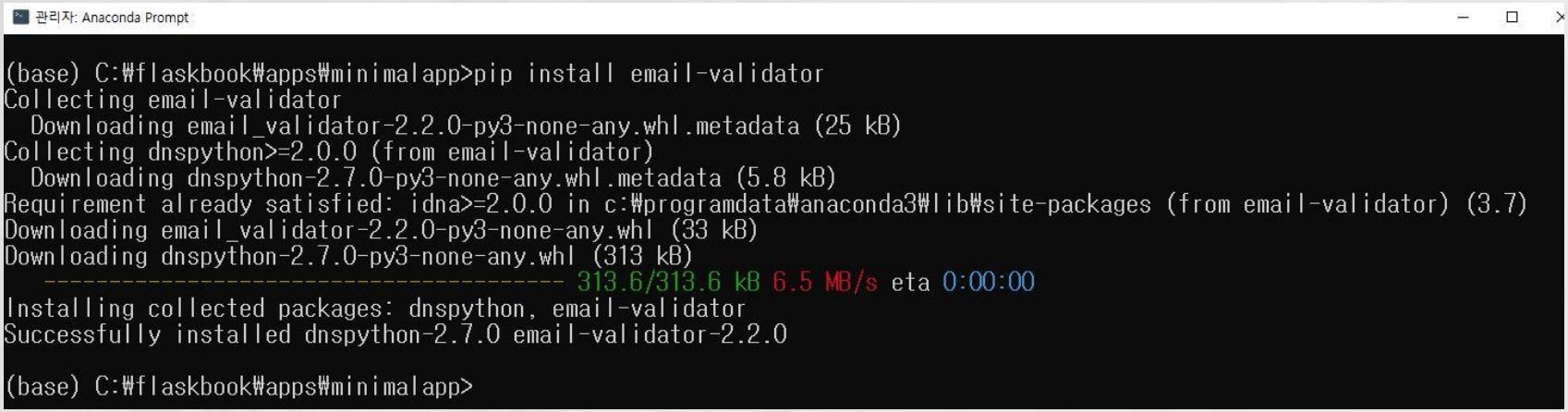
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# flask 클래스를 import 한다
from flask import Flask, render_template, request, url_for, redirect, make_response, session

# flask 클래스를 인스턴스화한다
app = Flask(__name__)

# SECRET_KEY를 추가한다
app.config["SECRET_KEY"] = "2AZSMss3p5QPbcY2hBsJ"
```

Flash 메시지

POST 값의 입력 체크를 추가

- 이메일 주소(email)가 올바른 형식인지 여부를 체크하기 위해서 email-validator 패키지를 설치
- pip install email-validator



Flash 메시지

POST 값의 입력 체크를 추가

- C:₩flaskbook₩apps₩minimalapp₩app.py의 import추가 및 contact_complete 갱신

app.py - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# flask 클래스를 import 한다
from email_validator import EmailNotValidError, validate_email

from flask import (
    Flask,
    current_app,
    make_response,
    redirect,
    render_template,
    request,
    session,
    url_for,
    flash,
)
```

Flash 메시지

POST 값의 입력 체크를 추가

- C:₩flaskbook₩apps₩minimalapp₩app.py의 import추가 및 contact_complete 갱신

app.py - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
@app.route("/contact/complete", methods=["GET", "POST"])
def contact_complete():
    if request.method == "POST":
        # form 속성을 사용해서 폼의 값을 취득한다
        username = request.form["username"]
        email = request.form["email"]
        description = request.form["description"]

        # 입력 체크
        is_valid = True
        if not username:
            flash("사용자명은 필수입니다")
            is_valid = False

        if not email:
            flash("메일 주소는 필수입니다")
            is_valid = False
    try:
        validate_email(email)
    except EmailNotValidError:
        flash("메일 주소의 형식으로 입력해 주세요")
        is_valid = False

    if not description:
        flash("문의 내용은 필수입니다")
        is_valid = False

    if not is_valid:
        return redirect(url_for("contact"))

    # 문의 완료 엔드포인트로 리다이렉트한다
    flash("문의 내용은 메일로 송신했습니다. 문의해 주셔서 감사합니다.")

    return redirect(url_for("contact_complete"))
return render_template("contact_complete.html")
```

Flash 메시지

POST 값의 입력 체크를 추가

- 이메일 주소 형식 체크용 validate_email과 EmailNotValidError를 import
- flash 를 추가로 import
- 사용자명, 이메일 주소, 문의 내용 입력란이 비어 있으면 flash에 오류 메시지를 설정
- email이 이메일 주소의 형식인지 여부를 validate_email 함수로 확인
- 형식이 정확하지 않은 경우는 예외가 발생하므로 try-except로 사용
- POST 값에 문제가 없는 경우는 Flash 메시지에 “문의해 주셔서 감사합니다.”를 설정
- 문의 완료 화면으로 리다이렉트

Flash 메시지

POST 값의 입력 체크를 추가

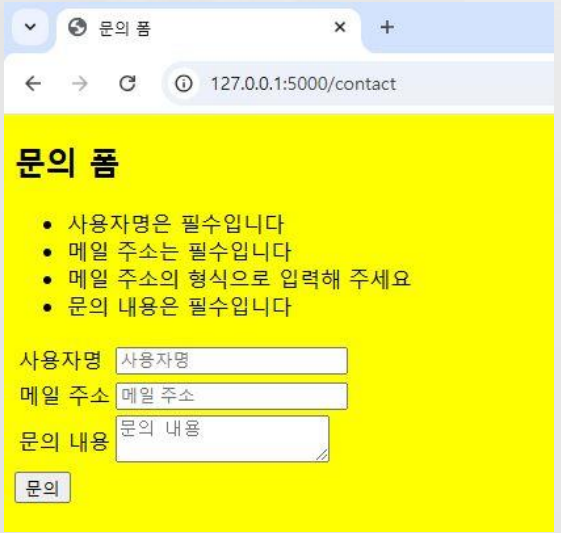
- C:₩flaskbook₩apps₩minimalapp₩templates₩contact.html 의

  get_flashed_messages 함수 추가

contact.html - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
<body>
    <h2>문의 폼</h2>

    {% with messages = get_flashed_messages() %}
    {% if messages %}
    <ul>
        {% for message in messages %}
        <li class="flash">{{ message }}</li>
        {% endfor %}
    </ul>
    {% endif %}
    {% endwith %}
```



Flash 메시지

POST 값의 입력 체크를 추가

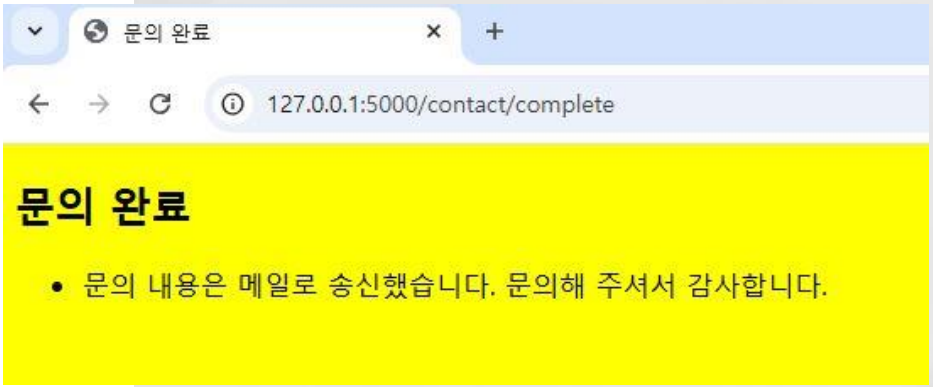
- C:₩flaskbook₩apps₩minimalapp₩templates₩contact_complete.html

  에도 flash 메시지 추가

contact_complete.html - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

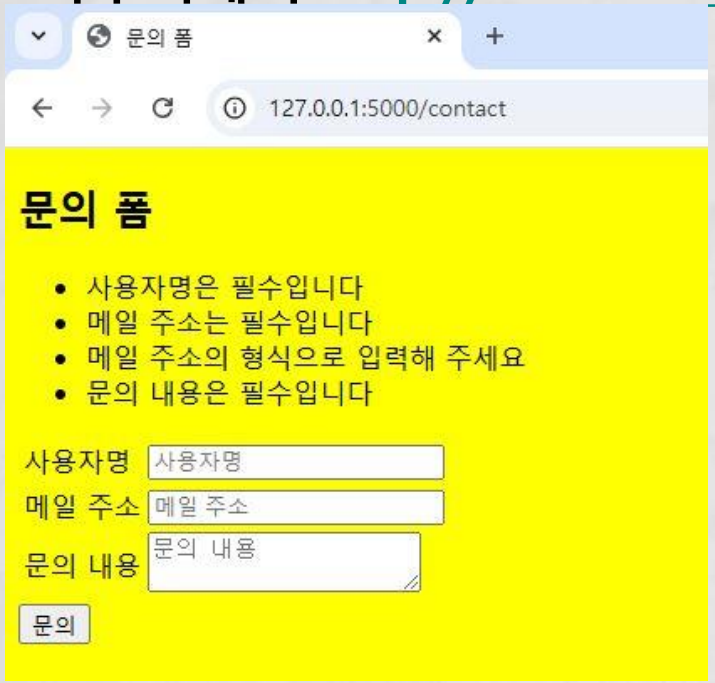
```
<body>
    <h2>문의 완료</h2>

    {% with messages = get_flashed_messages() %}
    {% if messages %}
    <ul>
        {% for message in messages %}
        <li>{{ message }}</li>
        {% endfor %}
    </ul>
    {% endif %}
    {% endwith %}

</body>
```

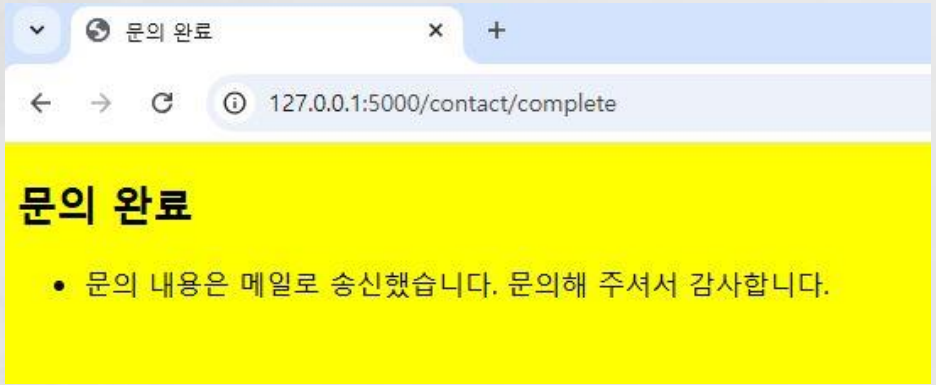


Flash 메시지

Flask 실행 및 확인

- (base) C:₩flaskbook₩apps₩minimalapp>flask run
- 브라우저에서 http://127.0.0.1:5000/contact URL에 접근
- 브라우저에서 http://127.0.0.1:5000/contact/complete URL에 접근



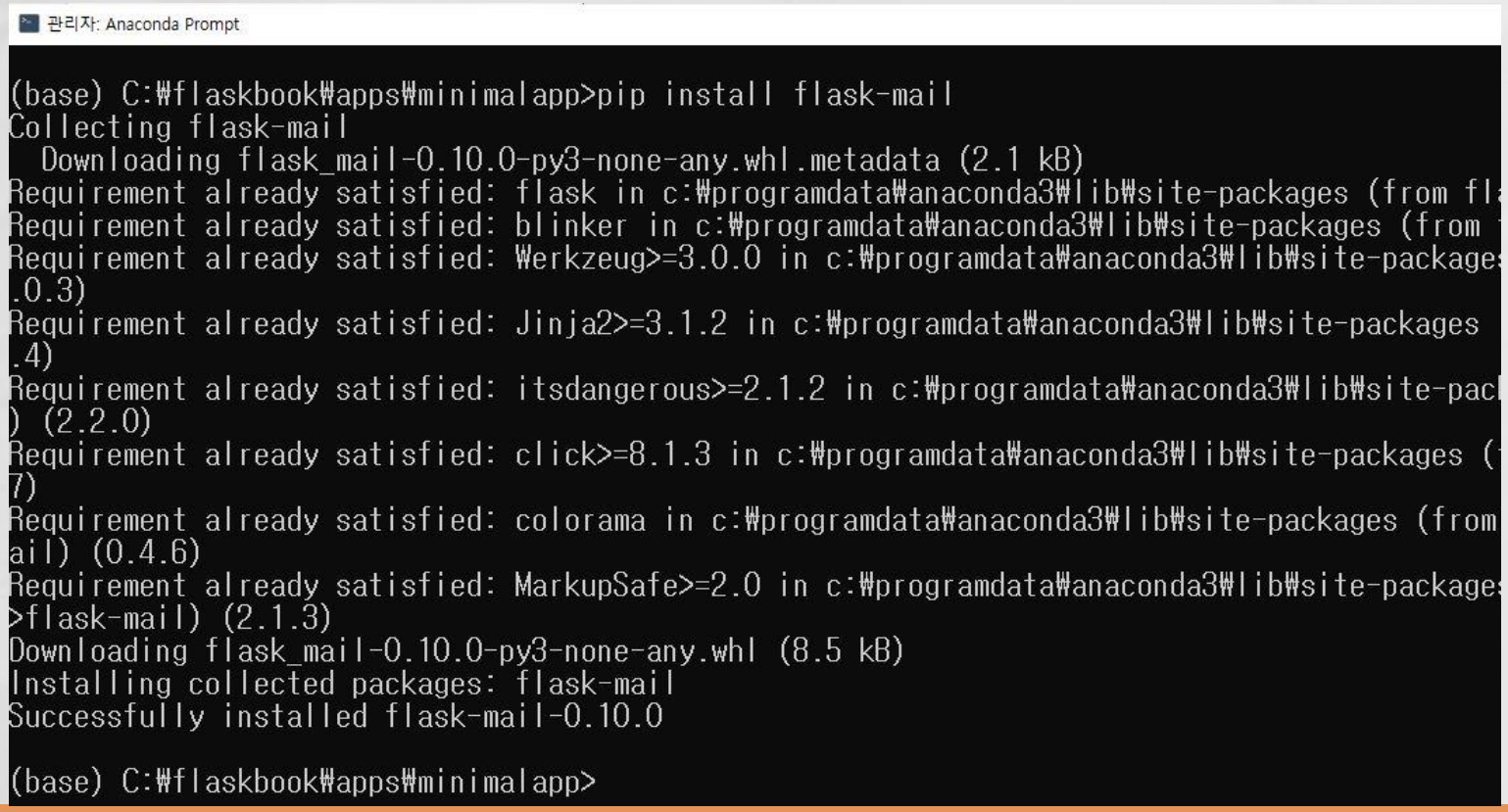


이메일 보내기

Flask-mail 설치

- 이메일을 보내기 위해서 플라스크의 flask-mail 확장을 설치

- pip install flask-mail



이메일 보내기

config 설정

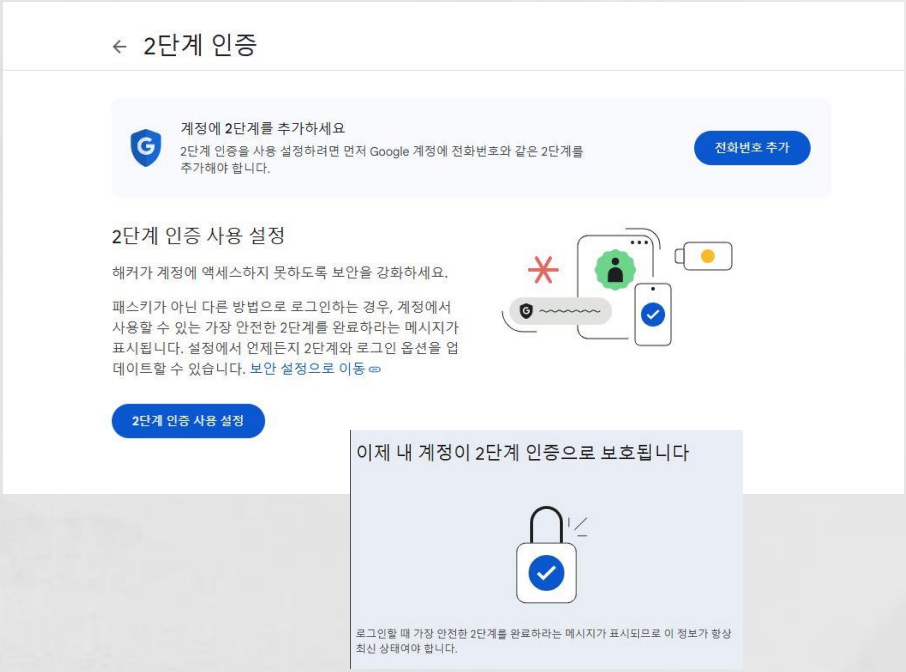
- config의 설정

설정	기본값	설명
MAIL_SERVER	localhost	이메일 서버의 호스트명
MAIL_PORT	25	이메일 서버의 포트
MAIL_USE_TLS	False	TLS를 유효로 하는가
MAIL_USE_SSL	False	SSL을 유효로 하는가
MAIL_DEBUG	app.debug	디버그 모드
MAIL_USERNAME	None	송신자 이메일 주소
MAIL_PASSWORD	None	송신자 이메일 주소의 비밀번호
MAIL_DEFAULT_SENDER	None	이메일의 송신자명과 이메일 주소

이메일 보내기

애플리케이션에서 Gmail로 이메일 보내기 준비

- Gmail을 사용하여 앱에서 이메일을 보내려면 먼저 다음의 Gmail 2단계 인증 프로세스 페이지에서 2단계 인증을 설정
- https://myaccount.google.com/signinoptions/two-step-verification/enroll-welcome

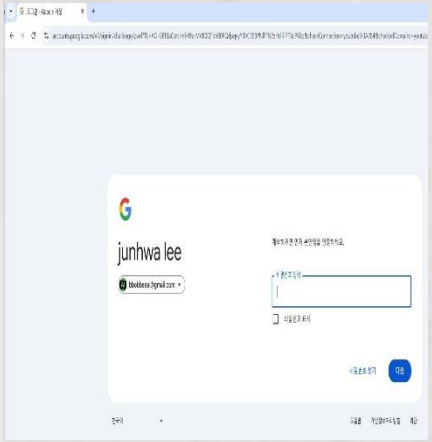


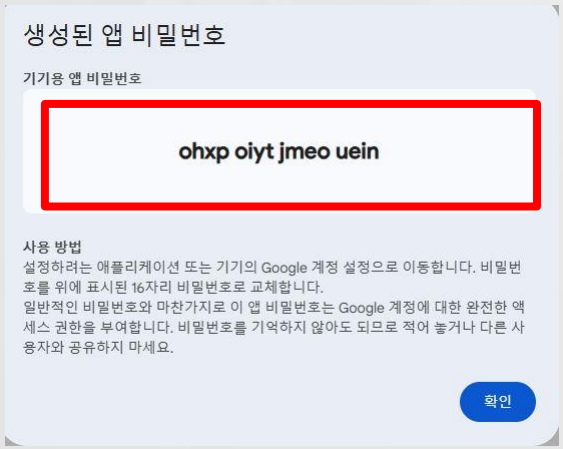
이메일 보내기

애플리케이션에서 Gmail로 이메일 보내기 준비

- 앱용 비밀번호 만들기

- https://https://security.google.com/settings/security/apppasswords





이메일 보내기

이메일 송신 기능 구현

- 환경 변수를 취득하기 위해서 os를 import
- Mail 클래스를 import
- Mail 클래스의 config를 환경 변수로부터 설정
- flask-mail 확장을 앱에 등록

이메일 보내기

이메일 송신 기능 구현

- C:₩flaskbook₩apps₩minimalapp₩app.py에 flask-mail 처리 추가

app.py - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# flask 클래스를 import 한다
import os


from flask_mail import Mail, Message
```

이메일 보내기

이메일 송신 기능 구현

- C:₩flaskbook₩apps₩minimalapp₩app.py에 flask-mail 처리 추가

*app.py - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# Mail 클래스의 컨피그를 추가한다
app.config["MAIL_SERVER"] = os.environ.get("MAIL_SERVER")
app.config["MAIL_PORT"] = os.environ.get("MAIL_PORT")
app.config["MAIL_USE_TLS"] = os.environ.get("MAIL_USE_TLS")
app.config["MAIL_USERNAME"] = os.environ.get("MAIL_USERNAME")
app.config["MAIL_PASSWORD"] = os.environ.get("MAIL_PASSWORD")
app.config["MAIL_DEFAULT_SENDER"] = os.environ.get("MAIL_DEFAULT_SENDER")

# flask-mail 확장을 등록한다
mail = Mail(app)

# URL과 실행할 함수를 매핑한다
@app.route("/")
```

이메일 보내기

이메일 송신 기능 구현

- C:₩flaskbook₩apps₩minimalapp₩.env

.env - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
FLASK_ENV=development

# flask-mail의 config 설정
MAIL_SERVER=smtp.gmail.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=bbobbeeai@gmail.com
MAIL_PASSWORD=ohxp oiyt jmeo uein
MAIL_DEFAULT_SENDER=Flaskbook <bbobbeeai@gmail.com>
```

이메일 보내기

이메일 송신 기능 구현

- MAIL_SERVER는Gmail의 송신 이메일 서버 smtp.gmail.com을 설정
- MAIL_PORT는 TLS(STARTLS : 이메일 암호화)를 사용하기 위해 포트 번호 587을 설정
- MAIL_USE_TLS는 TLS를 유효로 하기 위해 True를 설정
- MAILJJSERNAME은 이용하는 Gmail의 이메일 주소를 설정
- MAIL PASSWORD는 새로 생성한 앱용 비밀번호를 설정
- MAIL_DEFAULT_SENDER는 이메일의 송신자명 Flaskbook <Gmail의 이메일 주소>를 설정

이메일 보내기

이메일 보내기

- C:₩flaskbook₩apps₩minimalapp₩app.py에 문의 완료 엔드포인트 contact_complete에 이메일을 보내는 처리를 추가

app.py - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# 문의 완료 엔드포인트로 리다이렉트한다
flash("문의 내용은 메일로 송신했습니다. 문의해 주셔서 감사합니다.")

# 메일을 보낸다
send_email(
    email,
    "문의 감사합니다.",
    "contact_mail",
    username=username,
    description=description,
)
```

이메일 보내기

이메일 보내기

- C:₩flaskbook₩apps₩minimalapp₩app.py에 문의 완료 엔드포인트 contact_complete에 이메일을 보내는 처리를 추가

```
            "contact_mail",
            username=username,
            description=description,
        )

        return redirect(url_for("contact_complete"))
    return render_template("contact_complete.html")


def send_email(to, subject, template, **kwargs):
    """메일을 송신하는 함수"""
    msg = Message(subject, recipients=[to])
    msg.body = render_template(template + ".txt", **kwargs)
    msg.html = render_template(template + ".html", **kwargs)
    mail.send(msg)
```

이메일 보내기

이메일 템플릿 만들기

- C:₩flaskbook₩apps₩minimalapp₩templates₩contact_mail.txt
- 텍스트 이메일의 템플릿 contact_mail.txt를 작성
- 변수로 치환하는 부분은 {{ }}로 지정

이메일 보내기

이메일 템플릿 만들기

- C:₩flaskbook₩apps₩minimalapp₩templates₩contact_mail.html

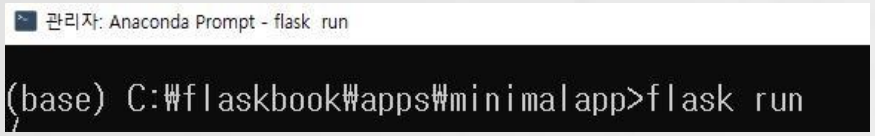
*contact_mail.html - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8" />
    <title>문의 완료</title>
</head>

<body>
    <p>{{ username }} 씨</p>
    <p>문의 감사합니다. 문의 내용은 다음과 같습니다.</p>
    <p>문의 내용</p>
    <p>{{ description }}</p>
</body>

</html>
```

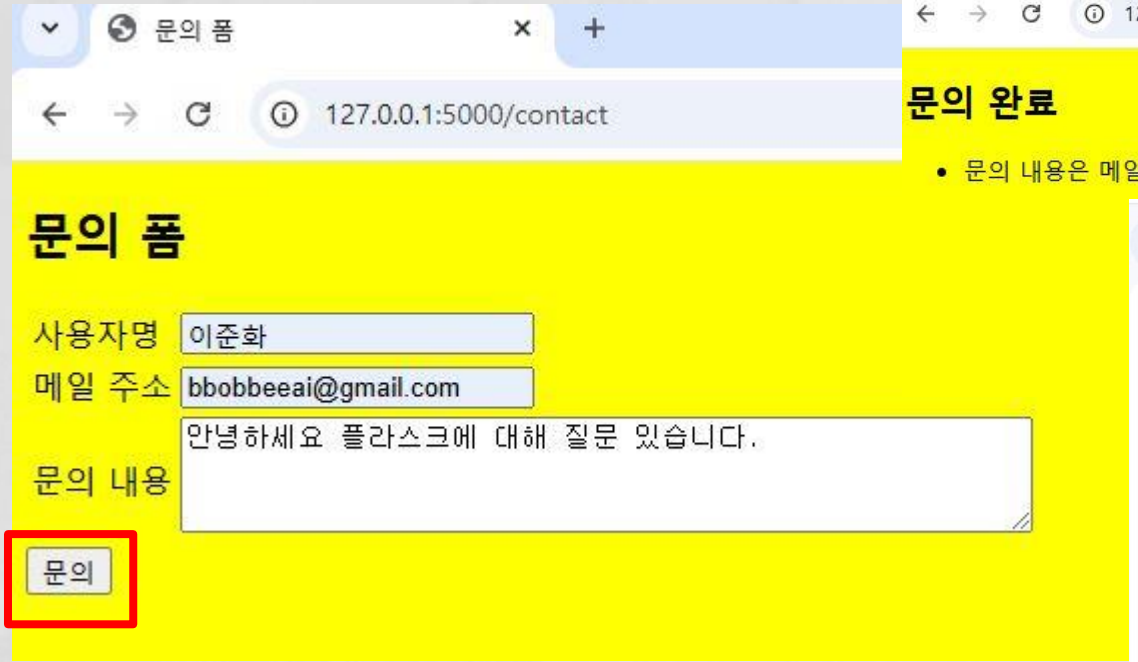
Flash 메시지

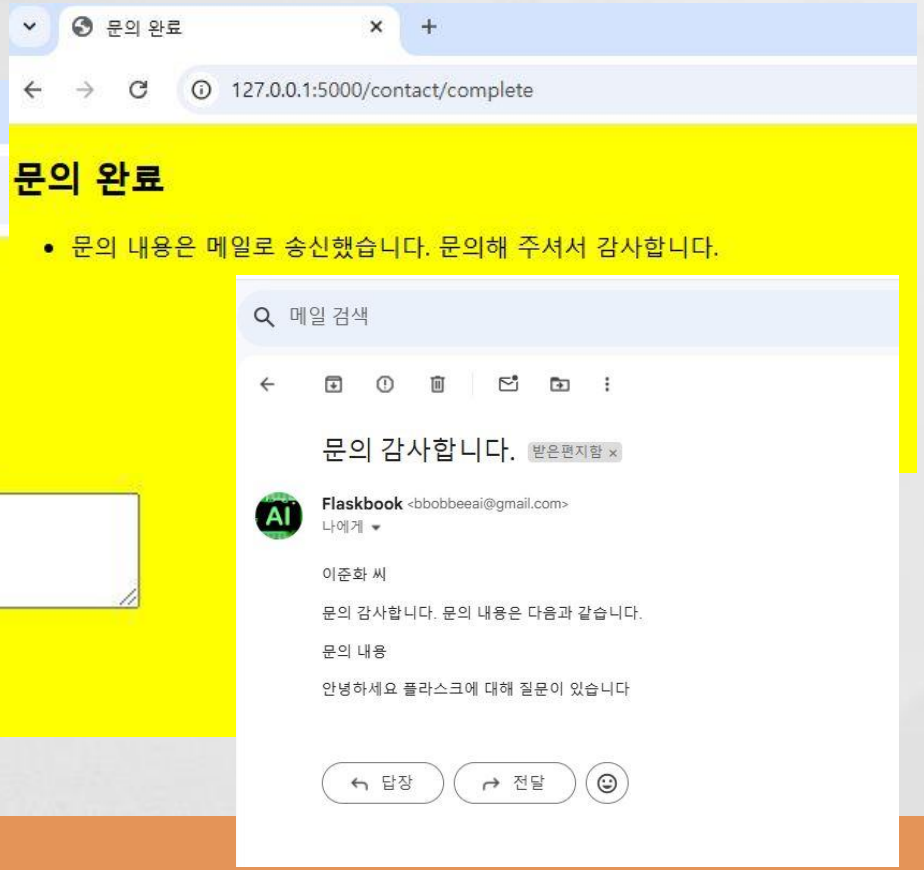
Flask 실행 및 확인



- (base) C:₩flaskbook₩apps₩minimalapp>flask run

- 브라우저에서 http://127.0.0.1:5000/contact URL에 접근

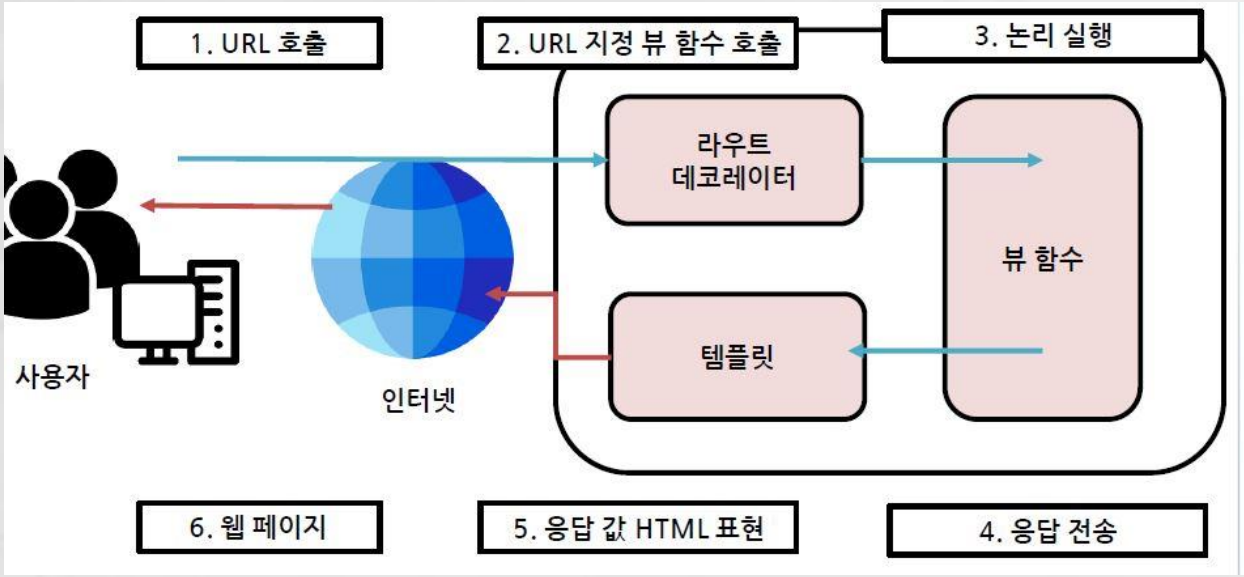




플라스크 앱 구조

플라스크 앱 구조

- 플라스크 앱은 다음 과정을 통해 호출
  - 특정 URL 호출
  - 뷰 함수 호출
  - 논리 실행
  - 논리 결과 응답 전송
  - 응답 값 HTML 표현
  - 클라이언트 전달

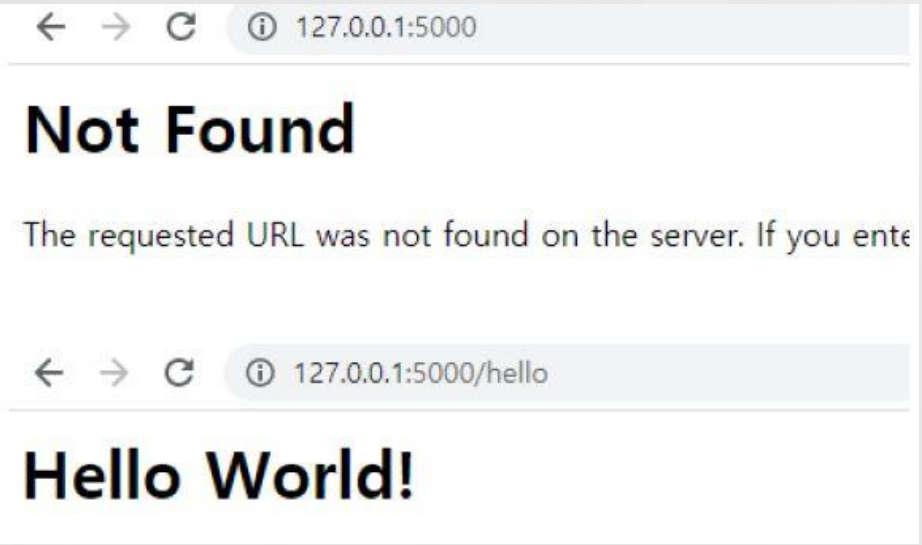


플라스크 앱 구조

플라스크 정적 라우팅

- 플라스크는 복잡한 URI 를 쉽게 함수로 연결하는 방법 제공
- route() 함수 사용 앱 객체 제어

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

@app.route("/hello")
def hello():
  return "<h1>Hello World!</h1>"

if __name__ == "__main__":
  app.run()
```



플라스크 앱 구조

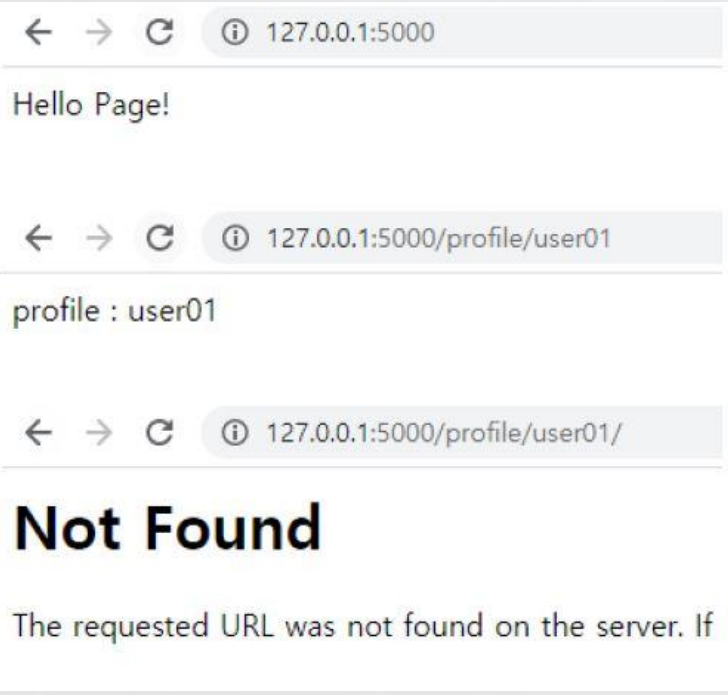
플라스크 동적 라우팅

- 동적 라우팅 상황에 따라 변화하는 URI 적용

- URL 을 변수 형태로 사용

➢ 입력 URL 인자에 따라 출력이 바뀐다. (/)

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

@app.route("/")
def hello():
  return "Hello Page!"

@app.route('/profile/<username>/')
def get_profile(username):
  return 'profile : ' + username

if __name__ == "__main__":
  with app.test_request_context():
    print (url_for('get_profile', username='flask'))
  app.run()
```

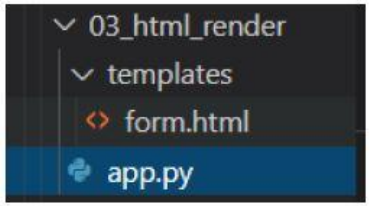


플라스크 앱 구조

플라스크 HTML 렌더링

- 플라스크는 기본 HTML 코드를 반환하면 페이지로 렌더링함

```
<form>
  <p>이름: <input type="text" id="input"></p>
  <p>이름 입력 후 제출버튼을 누르세요.
    <input type="button" value="제출 " onclick="alert('입력')"/>
  </p>
</form>
```

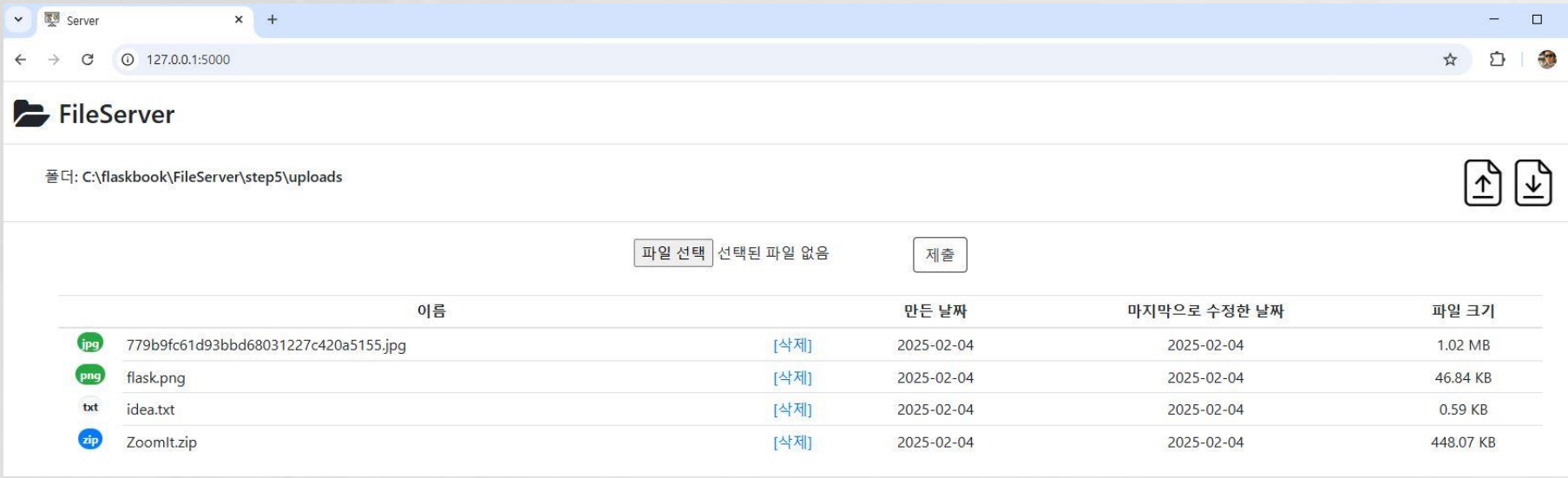



```
from flask import Flask, render_template
app = Flask(__name__)

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

if __name__ == "__main__":
  app.run()
```

파일 서버(File Server) 구현

File Server 사이트 설계

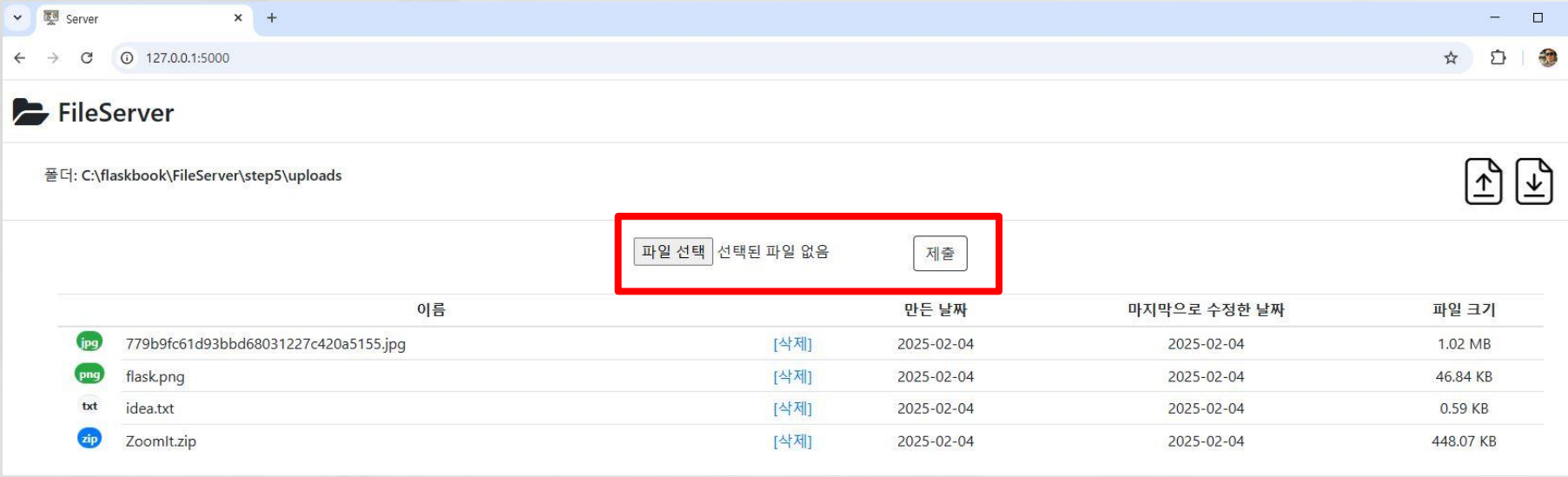
- 전체

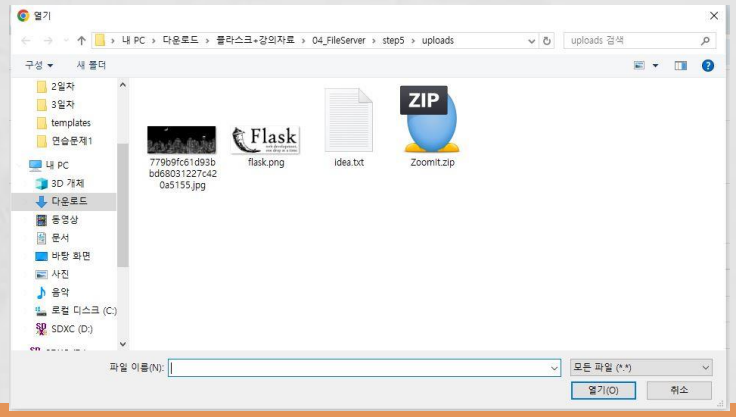


파일 서버(File Server) 구현

File Server 사이트 설계

- 파일 업로드

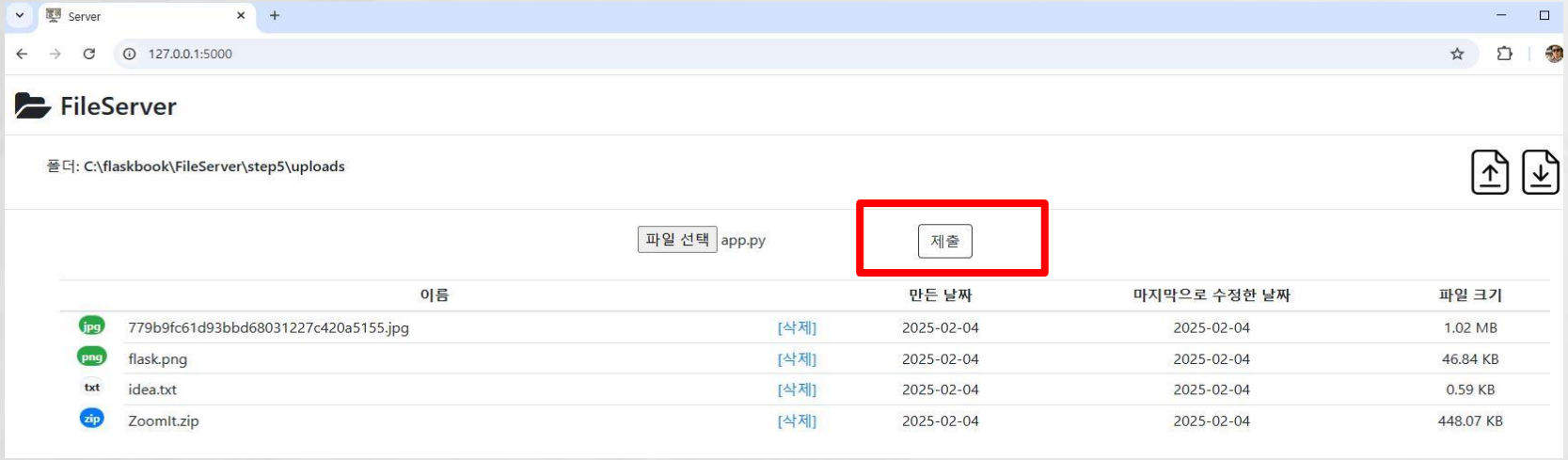


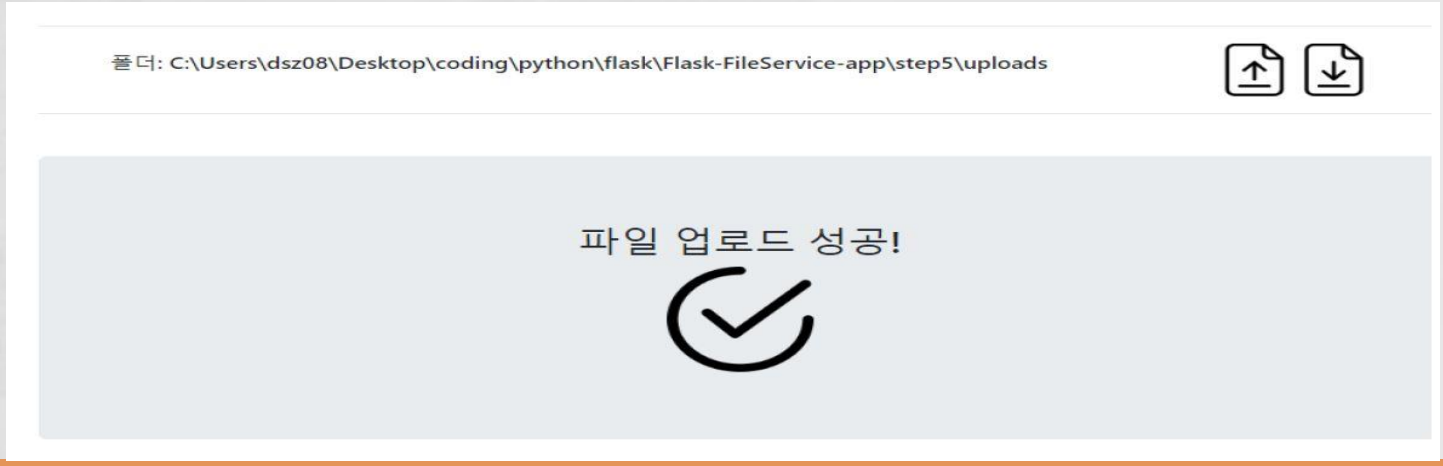


파일 서버(File Server) 구현

File Server 사이트 설계

- 파일 업로드

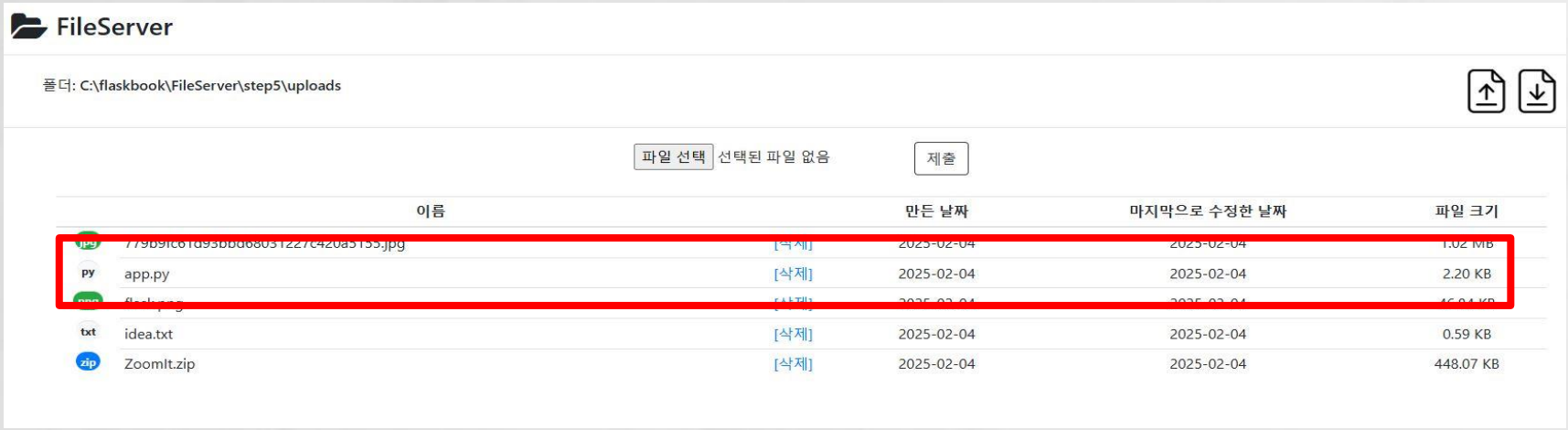




파일 서버(File Server) 구현

File Server 사이트 설계

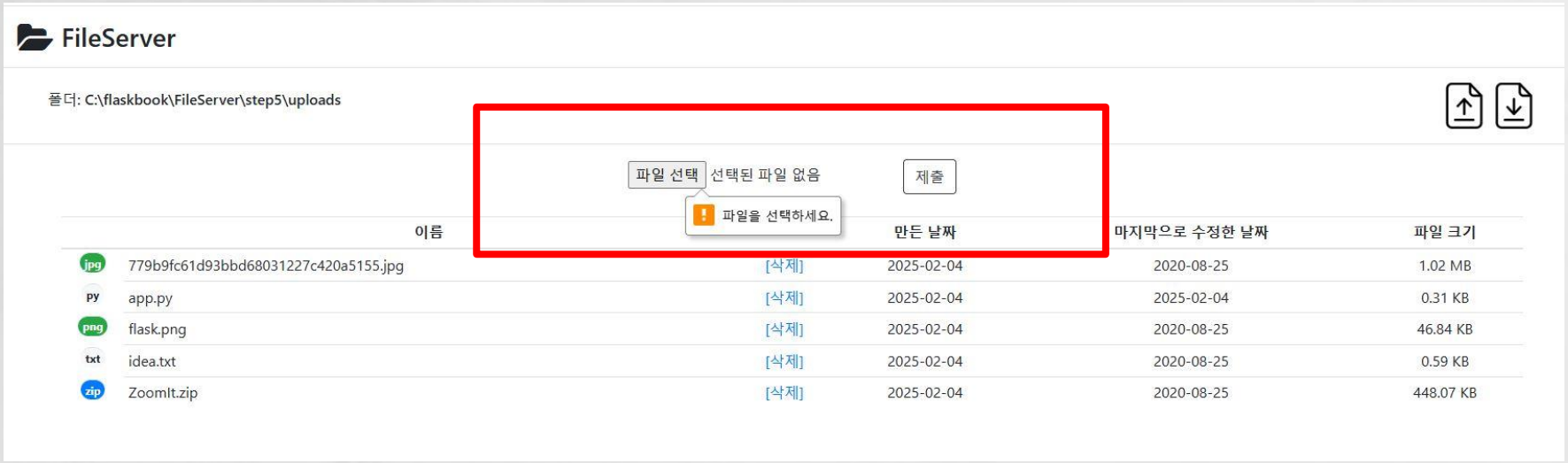
- 파일 업로드



파일 서버(File Server) 구현

File Server 사이트 설계

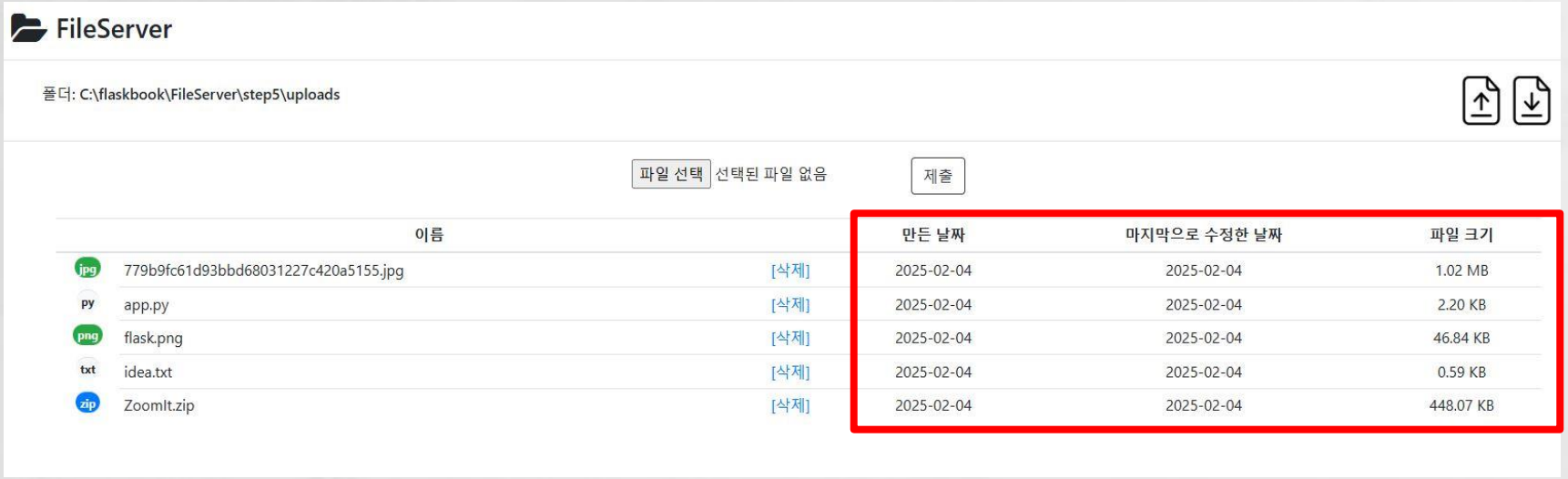
- 파일 업로드 검증



파일 서버(File Server) 구현

File Server 사이트 설계

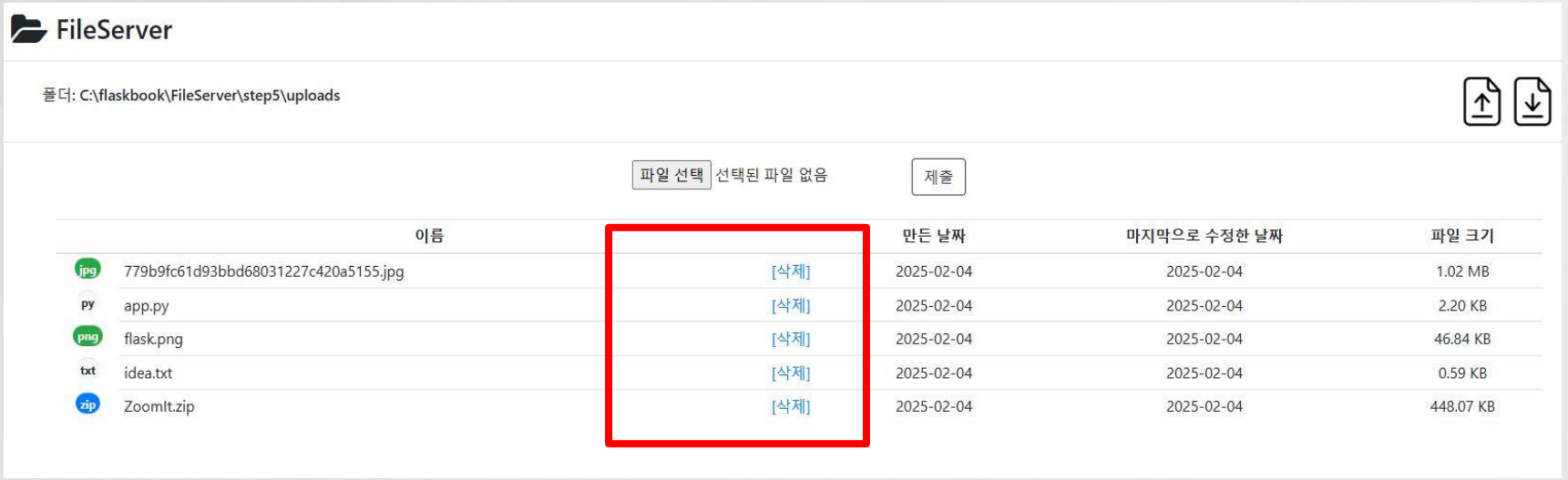
- 파일 정보 확인 및 나열

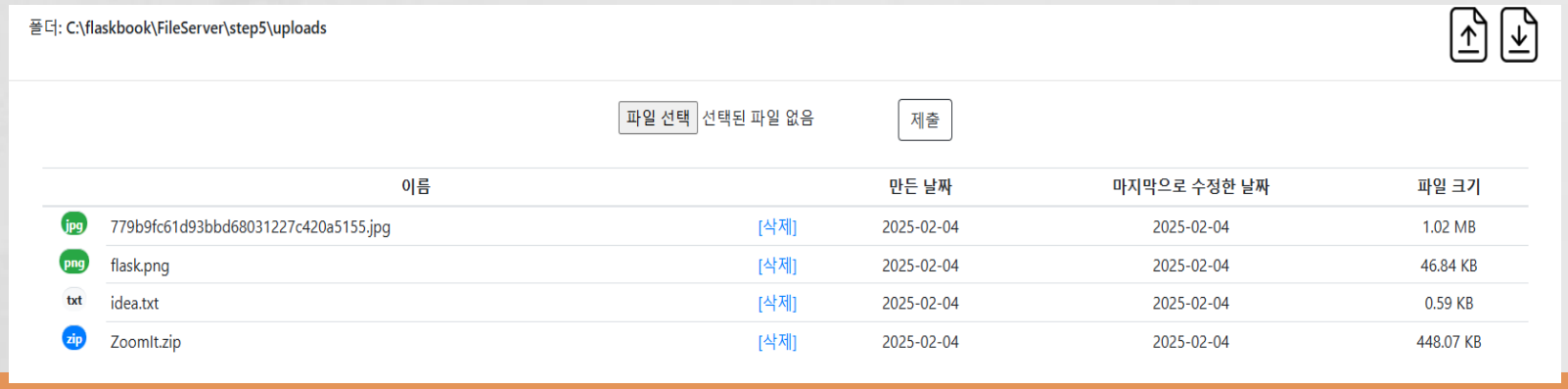


파일 서버(File Server) 구현

File Server 사이트 설계

- 파일 삭제





정리

정리

- Flash 메시지
- 이메일 보내기
- File Server 만들기