

플라스크 웹 프로그램

플라스크 작성

url_for 함수를 사용해서 URL 생성하기

- 엔드포인트의 URL을 이용하려면 url_for 함수를 사용하면 편리
- HTML 파일이나 View 파일에 /name과 같이 기술
- 이것을 url_for("name")와 같이 기술
- 엔드포인트에 대응하는 Rule이 바뀐다고 해도 HTML 파일이나 View에 기술하는 URL을 변경할 필요가 없음
- test_request_context 함수를 사용하여 현재의 루트 정보를 url_for 함수로 출력

플라스크 작성

url_for 함수를 사용해서 URL 생성하기

- flask routes 명령어로 현재의 루트 정보를 확인
- url_for 를 추가로 import
- app.py의 맨 아래 부분에 with app.test_request_context():를 추가하고

url_for() 출력

- url_for의 1번째 인수에는 엔드포인트를 지정(flask run을 실행하면 콘솔에 URL이 출력)
- URL 규칙의 변수에 값을 설정하는 경우는 2번째 인수에 key=value의 타입으로 지정
- URL 규칙의 변수에 값을 설정 후 그 다음 인수에 key=value를 지정하면 GET 파라미터

플라스크 작성

url_for 함수를 사용해서 URL 생성하기

o C:\flaskbook\apps\minimalapp\app.py

app.py - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# flask 클래스를 import 한다
from flask import Flask, render_template, url_for
```

#url_for 함수를 사용해서 URL 생성하기

```
with app.test_request_context():
    # /
    print(url_for("index"))
    # /hello/world
    print(url_for("hello-endpoint", name="world"))
    # /name/AK?page=1
    print(url_for("show_name", name="AK", page="1"))
```

플라스크 작성

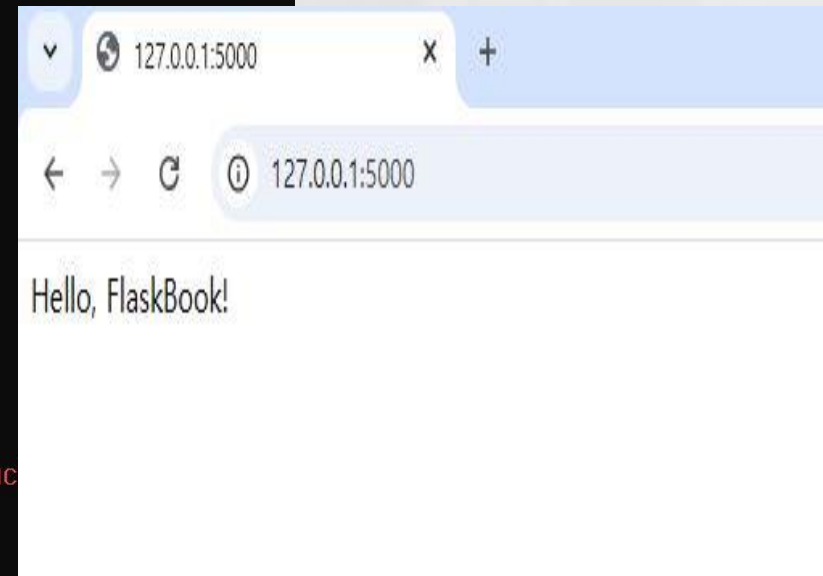
url_for 함수를 사용해서 URL 생성하기

- (base) C:\flaskbook\apps\minimalapp> flask run
- 브라우저에서 <http://127.0.0.1:5000> URL에 접근

관리자: Anaconda Prompt - flask run

```
(base) C:\flaskbook\apps\minimalapp>flask routes
/
/hello?name=world
/name/AK?page=1
Endpoint      Methods  Rule
-----
hello-endpoint GET      /hello
index         GET      /
show_name     GET      /name/<name>
static        GET      /static/<path:filename>
```

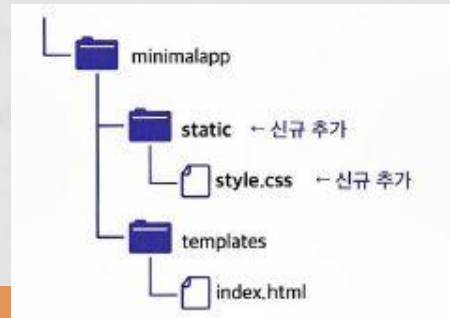
```
(base) C:\flaskbook\apps\minimalapp>flask run
/
/hello?name=world
/name/AK?page=1
* Debug mode: off
WARNING: This is a development server. Do not use it in a production
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



플라스크 작성

정적 파일 이용하기

- 웹사이트에서는 HTML 과 함께 이미지나 JavaScript, CSS (Cascading Style Sheets) 를 이용
- 요청 내용에 관계없이 항상 같은 내용이 나타나므로 정적 파일
- CSS는 HTML 의 외형인 UI를꾸미기 위해 이용하고, JavaScript는 주로 HTML의 동작을처리하기 위해 이용
- 플라스크에서는 정적 파일을 이용할 때 기본적으로 static 디렉터리 배치



플라스크 작성

정적 파일 이용하기

- `url_for('static', filename='style.css')` 으로 활용
- `C:\flaskbook\apps\minimalapp\static\style.css` 수정
- `C:\flaskbook\apps\minimalapp\templates/index.html`

```
*style.css - Windows 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)

body {
    background-color: #ffff00;
}
h1 {
    margin-top: 40px;
}
```

*index.html - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <title>Name</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" />
  </head>
  <body>
    <h1>Name: {{ name }}</h1>
  </body>
</html>
```

플라스크 작성

정적 파일 이용하기

- (base) C:\flaskbook\apps\minimalapp>flask run
- 브라우저에서 <http://127.0.0.1:5000> URL에 접근

```
관리자: Anaconda Prompt - flask run
127.0.0.1 - - [02/Feb/2025 21:28:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 21:28:39] "GET /hello HTTP/1.1" 200 -

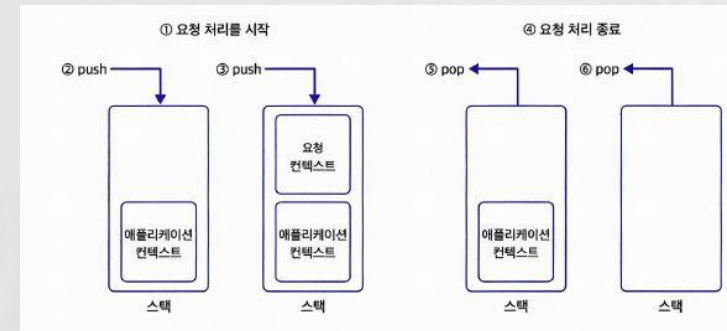
(base) C:\flaskbook\apps\minimalapp>flask run
/
/hello?name=world
/name/AK?page=1
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [02/Feb/2025 21:35:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 21:35:16] "GET /name/AK HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 21:35:16] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 21:38:08] "GET /name/AK HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 21:38:08] "GET /static/style.css HTTP/1.1" 200 -
```



플라스크 작성

컨텍스트의 라이프 사이클

- 요청 처리를 시작
- 애플리케이션 컨텍스트를 작성 (스택으로 push)
- 요청 컨텍스트를 작성 (스택으로 push)
- 요청 처리를 종료
- 요청 컨텍스트를 삭제 (스택으로부터 pop)
- 애플리케이션 컨텍스트를 삭제 (스택으로부터 pop)



문의 폼 만들기

문의 폼의 사양

- 문의 폼 화면
- 문의 완료 화면
- 문의 폼 화면으로부터 문의를 하면
- 입력한 이메일 주소에 문의 내용을 송신하고 문의 완료 화면을 표시

문의 폼 화면(문의의 입력)

문의폼

사용자명

이메일 주소

문의 내용

이메일 송신



문의 완료 화면(완료의 표시)

문의 완료

• 문의해 주셔서 감사합니다.

문의 폼 만들기

PRG 패턴

- PRG 패턴이란 POST/REDIRECT/GET 패턴의 약어
- 폼 데이터를 POST하면 REDIRECT (리다이렉트) 하여 GET한 페이지를 표시하는 패턴
- PRG 패턴을 사용하지 않는 경우
- 폼 데이터를 POST한 다음에 리로드하면 본래 POST한 콘텐츠가 재송신되어 폼 데이터가 이중으로 전송될 가능성

문의 폼 만들기

PRG 패턴

- 문의 폼 화면을 표시 (GET)
- 문의 내용을 이메일로 송신(POST)
- 문의 완료 화면으로 리다이렉트(REDIRECT)
- 문의 완료 화면을 표시 (GET)
- 문의 폼 (contact) 과 문의 완료 (contact.complete) 의 경로 정보

Endpoint	Methods	Rule
contact	GET	/contact
contact_complete	GET, POST	/contact/complete

문의 품 만들기

요청과 리다이렉트

- 요청 정보를 취득하려면 플라스크 모듈에서 요청 객체(request)를 import
- 다른 엔드포인트로 리다이렉트하려면 redirect 함수를 사용
- C:\flaskbook\apps\minimalapp\app.py 에 request의 import

app.py - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

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

문의 품 만들기

요청과 리다이렉트

- 대표적인 요청 객체 (request)의 속성 또는 메서드

속성 또는 메서드	설명
method	요청 메서드
form	요청 품
args	쿼리 파라미터
cookies	요청 쿠키 (Cookie)
files	요청 파일
environ	환경 변수
headers	요청 헤더
referrer	요청의 리퍼러 (링크 참조 페이지)
query_string	요청 쿼리 문자열
Scheme	요청의 프로토콜 (http/https)
url	요청 URL

문의 폼 만들기

문의 폼의 엔드포인트 만들기

- 문의 폼 화면을 표시하는 엔드포인트
- 이메일을 보내 문의 완료 화면을 표시하는 엔드포인트

← → ↻ 127.0.0.1:5000/contact

문의폼

사용자명

메일 주소

문의 내용

← → ↻ 127.0.0.1:5000/contact/complete

문의 완료

문의 폼 만들기

문의 폼의 엔드포인트 만들기

- o C:\flaskbook\apps\minimalapp\app.py

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

@app.route("/contact")
def contact():
    #이메일을 보낸다.

    #contact 엔드포인트로 리다이렉트 한다.
    return render_template("contact.html")

@app.route("/contact/complete", methods=["GET", "POST"])
def contact_complete():
    if request.method == "POST":
        return redirect(url_for("contact_complete"))
    return render_template("contact_complete.html")
```


문의 폼 만들기

문의 폼의 엔드포인트 만들기

- 문의 폼 화면을 반환하는 `contact` 엔드포인트를 만듦
- 문의 폼 처리•문의 완료 화면을 반환하는 `contact_complete` 엔드포인트를 만듦 `@app.route` 데코레이터의 2번째 인수에 `methods=[" GET " , " POST "]`를 지정하여 GET과 POST 메서드를 허가
- `Request.method` 속성을 이용하여 요청된 메서드를 확인
- GET의 경우는 문의 완료 화면(`contact_complete.html`)을 반환하고, POST의 경우는 문의 완료 엔드포인트(`contact_complete`)로 리다이렉트

문의 폼 만들기

문의 폼의 템플릿 만들기

- templates 디렉터리 바로 아래에 문의 폼 화면(contact.html)과 문의 완료 화면(contact_complete.html) 을 작성
- 문의 폼 화면(contact.html) 작성
- C:\flaskbook\apps\minimalapp\templates\contact.html

```
contact.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 폼</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css' ) }}" />
</head>
```

문의 폼 만들기

문의 폼의 템플릿 만들기

o C:\flaskbook\apps\minimalapp\templates\contact.html

```
contact.html - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

<body>
  <h2>문의 폼</h2>

  <form action="{{ url_for('contact_complete') }}" method="POST" novalidate="novalidate">
    <table>
      <tr>
        <td>사용자명</td>
        <td>
          <input type="text" name="username" value="{{ username }}" placeholder="사용자명" />
        </td>
      </tr>
      <tr>
        <td>메일 주소</td>
        <td>
          <input type="text" name="email" value="{{ email }}" placeholder="메일 주소" />
        </td>
      </tr>
    </table>
  </form>
</body>
```

문의 폼 만들기

문의 폼의 템플릿 만들기

- o C:\flaskbook\apps\minimalapp\templates\contact.html

```
<td>문의 내용</td>
<td>
    <textarea name="description" placeholder="문의 내용">{{ description }}</textarea>
</td>
</tr>
</table>
<input type="submit" value="문의" />
</form>
</body>
```

문의 폼 만들기

문의 폼의 템플릿 만들기

- 문의 완료 화면(contact_complete.html) 을 작성
- C:\flaskbook\wapps\minimalapp\templates\contact_complete.html

```
contact_complete.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>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css' ) }}" />
</head>

<body>
  <h2>문의 완료</h2>

</body>

</html>
```

문의 폼 만들기

문의 폼의 템플릿 만들기

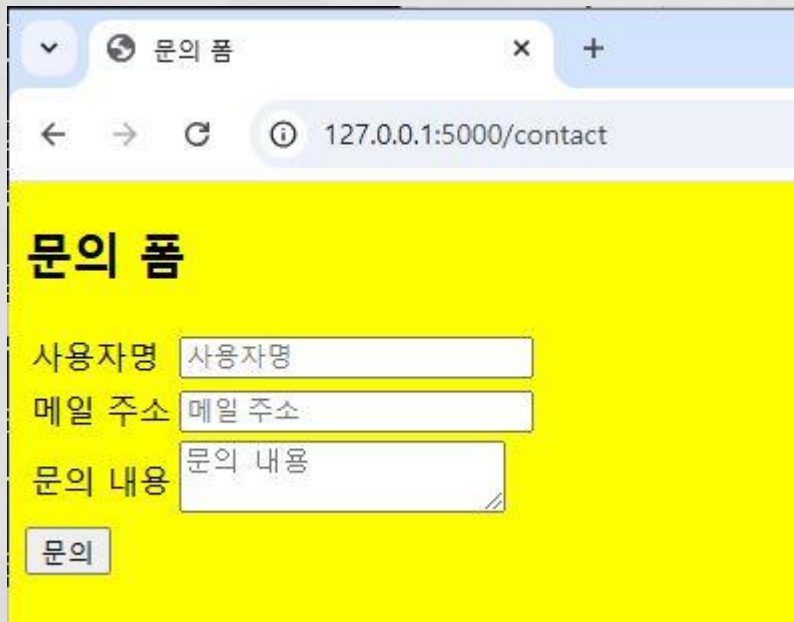
- C:\flaskbook\apps\minimalapp\flask routes

```
(base) C:\flaskbook\apps\minimalapp>flask routes
/
/hello?name=world
/name/AK?page=1
Endpoint                Methods      Rule
-----
contact                 GET          /contact
contact_complete        GET, POST    /contact/complete
hello-endpoint          GET          /hello
index                   GET          /
show_name               GET          /name/<name>
static                  GET          /static/<path:filename>
```

문의 폼 만들기

문의 폼의 템플릿 만들기

- (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

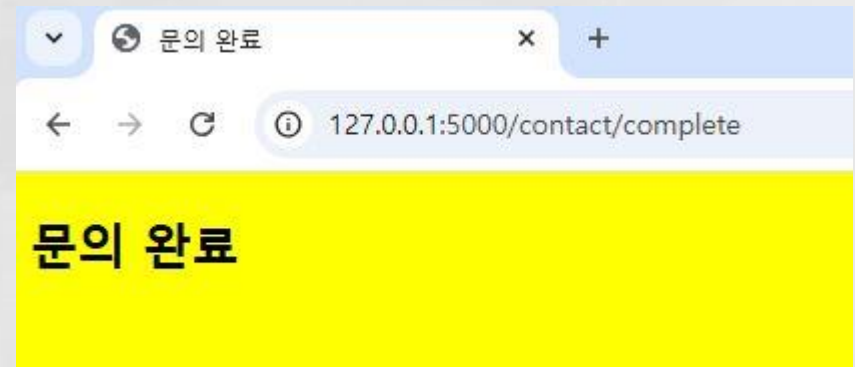


문의 폼

사용자명

메일 주소

문의 내용



문의 완료

문의 폼 만들기

POST 된 폼의 값 얻기

- POST된 폼의 값을 얻으려면 request의 form 속성을 이용
- C:\wflaskbook\wapps\minimalapp\app.py 수정

```
@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"]

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


쿠키(cookie)

쿠키(cookie)

- 쿠키(Cookie)는 클라이언트의 브라우저와 웹 서버와의 사이에서 상태를 관리하기 위해서 브라우저에 저장된 정보와 그 구조
- 서버에서 받아 브라우저에 보존된 쿠키 정보는 요청과 함께 쿠키를 보냈던 서버로 보냄
- 플라스크에서 쿠키로부터 값을 취득하는 데는 **요청 객체(request)**를 사용
- 또한 값의 설정에는 **make_response**로 얻은 **응답 객체(response)**를 사용

쿠키(cookie)

쿠키(cookie)

- 쿠키로부터 값을 취득하기 (request 객체)

```
from flask import request

# key를 지정한다
username = request.cookies.get("username")
```

- 쿠키로 값을 설정하기(response 객체)

```
from flask import make_response, render_template

response = make_response(render_template("contact.html"))
# key와 value를 설정한다
response.set_cookie("username", "AK")
```

- 쿠키로부터 값을 삭제하기(response 객체)

```
from flask import make_response, render_template, response

response = make_response(render_template("contact.html"))
# key를 지정한다
response.delete_cookie("username")
```

세션(Session)

세션(session)

- 세션은 사용자의 로그인 정보 등을 서버에 유지하고, 일련의 처리를 계속적으로 실시할 수 있도록 하는 구조
- HTTP는 스테이트리스이므로 상태를 유지할 수 없음
 - 스테이트리스(stateless)란 서버가 클라이언트의 정보를 유지하지 않아 웹 서버 측에서 상태를 관리할 수 없는 것
- 쿠키를 사용한 세션 관리 구조를 이용함으로써 사용자가 일련의 처리를 연속적으로 할 수 있음
- 플라스크에서 세션을 다루려면 `session`을 import

세션(Session)

세션(session)

- 세션에 값 설정하기

```
from flask import session  
  
session["username"] = "AK"
```

- 세션으로부터 값을 취득하기

```
from flask import session  
  
username = session["username"]
```

- 세션으로부터 값을 삭제하기

```
from flask import session  
  
session.pop("username", None)
```

응답(response)

응답(resoponse)

- 응답(response)은 브라우저로부터 온 요청에 대해 서버가 클라이언트에 게 반환하는 응답
- `return render_template("contact_complete.html")`
- 쿠키에 값을 설정하는 등 응답의 내용을 갱신해야 하는 경우는 `make_response` 함수를 이용

속성 또는 메서드	설명
<code>status_code</code>	응답 상태 코드
<code>headers</code>	응답 헤더
<code>set_cookie</code>	쿠키를 설정한다.
<code>delete_cookie</code>	쿠키를 삭제한다.

응답(response)

응답(resoponse)

o C:\w\flaskbook\wapps\wminimalapp\wapp.py

```
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

@app.route("/contact")
def contact():
    # 응답 오브젝트를 취득한다
    response = make_response(render_template("contact.html"))

    # 쿠키를 설정한다
    response.set_cookie("flaskbook key", "flaskbook value")

    # 세션을 설정한다
    session["username"] = "AK"

    # 응답 오브젝트를 반환한다
    return response
```

응답(response)

응답(resoponse)

- `make_response` 와 `session` 을 추가로 import
- `render_template`을 `make_response` 함수에 건네고, 응답 객체를 취득
- `set_cookie` 함수의 `key`에 flaskbook key, `value`에 flaskbook value를
설정
- `session["username"]`에 값을 설정
- 응답 객체를 반환

Jinja2 템플릿 엔진 파악

동적 웹 페이지 구성을 위한 Jinja 2 템플릿 엔진

- jinja 2 (Jinja 2 템플릿 엔진 동적 HTML 구성 특수문자 웹 보안 처리)
- jinja 2 변수 처리

➤ In python file,

```
return render_template('test.html', msg=msg)
```

➤ In HTML file,

```
{{ msg }}
```

✓ 콤마(,)로 데이터 구분

```
return render_template('test.html', msg=msg, name=name)
```

● 주석

```
{# ... #}
```


Jinja2 템플릿 엔진 파악

동적 웹 페이지 구성을 위한 Jinja 2 템플릿 엔진

- 조건문
- if 구문으로 변수나 변수 이용 표현을 사용해 존재 여부 변수 값 부울 값으로 판단
- 반드시 조건문이 끝나고 endif 를 붙여줘야 함

```
{% if <조건> %}  
  <코드>  
{% elseif <조건> %}  
  <코드>  
{% else %}  
  <코드>  
{% endif %}
```

Jinja2 템플릿 엔진 파악

동적 웹 페이지 구성을 위한 Jinja 2 템플릿 엔진

- 반복문
- 일반적인 for 반복 구조와 유사 리스트로 값 호출

```
{% for n in files %}  
  {{ n }}  
{% endfor %}
```

- 반복문, 인덱스
- 리스트 반복문 중 loop index 를 사용하면 인덱스를 반환함

```
<ul>  
  {% for item in items %}  
    <li>  
      {{ loop.index }} 번째 줄입니다  
    </li>  
  {% endfor %}  
</ul>
```

Jinja2 템플릿 엔진 파악

동적 웹 페이지 구성을 위한 Jinja 2 템플릿 엔진

◦ 반복문 표현식 사용

```
{% for i in range(1,10) %}  
  {{ i }}  
{% endfor %}
```

Loop 속성	설명
---------	----

loop.index	루프내 순서로 1부터 표시
------------	----------------

loop.index0	루프내 순서로 0부터 표시
-------------	----------------

loop.first	루프 첫 순서면 True
------------	---------------

loop.last	루프 마지막 순서면 True
-----------	-----------------

loop.length	전체 반복 횟수
-------------	----------

Jinja2 템플릿 엔진 파악

동적 웹 페이지 구성을 위한 Jinja 2 템플릿 엔진

- 기본 줄 끝 개행 문자 제외 공백은 제거하지 않음
- Jinja 2 공백
- + : 태그 앞 공백 제거 시작 태그
- - : 태그 시작과 끝 공백 제거

```
<ul>
  {% for user in users -%}
  <li><a href="{{ user.href }}">{{ user.caption
}}</a></li>
  {% endfor %}
</ul>
```

Jinja2 템플릿 엔진 파악

동적 웹 페이지 구성을 위한 Jinja 2 템플릿 엔진

- Jinja 2 템플릿 내부에서도 형 변환 및 포매팅(formatting) 가능
- Jinja 2 형 변환
- 데이터 변환할 자료형

```
{{ number|int }}
```

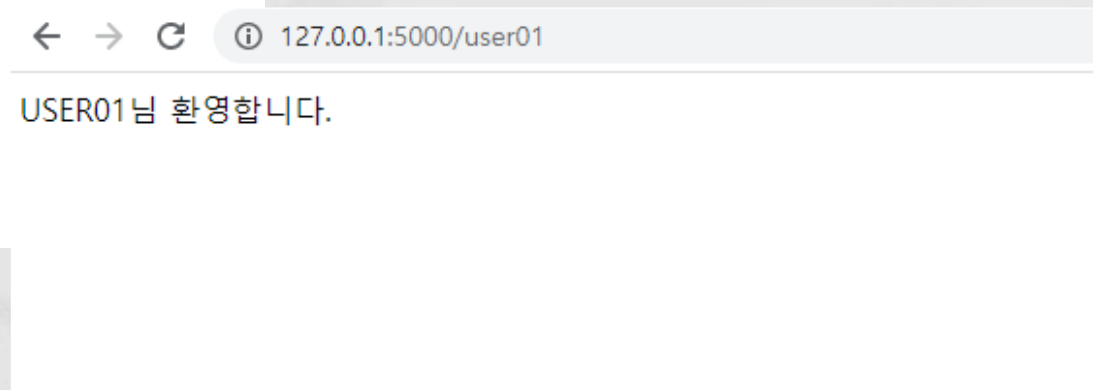
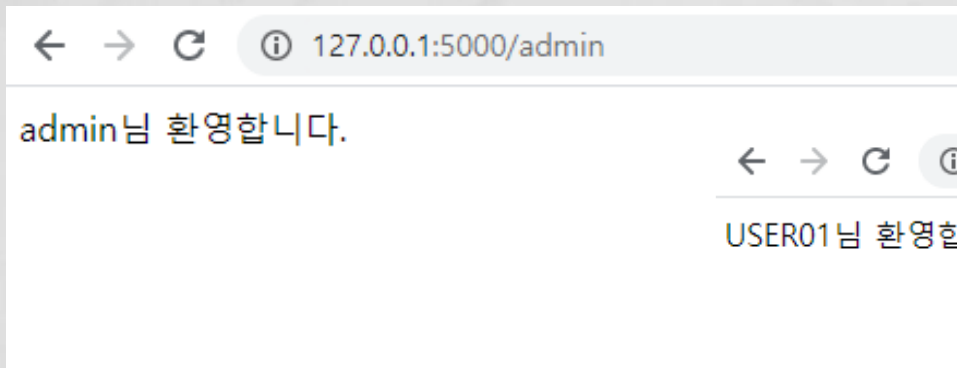
- 포매팅 세 자릿수마다 숫자 콤마 추가

```
{{ '{:,.}'.format(number[0]) }}
```

Jinja2 템플릿 엔진 파악

Jinja 2 템플릿 엔진 파악 연습문제

- 다음과 같은 페이지를 만들어보세요!
 - url 로 <username> 을 입력 받아
 - Jinja 2 템플릿 문법 사용 조건문으로 값이 문자열이 홀수개면 모두 소문자 짝수면 대문자로 표현



Jinja2 템플릿 엔진 파악

Jinja 2 템플릿 엔진 파악 연습문제

◦ 다음과 같은 페이지를 만들어보세요!

- GET 을 사용해 입력 받은 값을 버튼이 눌리면 POST 로 전달 후
- Jinja 2 템플릿 문법 사용 조건문으로 값 입력이 없으면 없다고 출력하고 있다면 반복문

사용 구구단 출력

출력 예 :

127.0.0.1:5000/3

구구단을 외자

출력하고자 하는 구구단을 입력하세요.

3 구구단 확인하기

구구단 확인하기

아직 아무 숫자도 입력이 안되었습니다.

값이 없다? => None

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 9 = 27

<종료>

정리

정리

- url_for 함수를 사용해서 URL 생성하기
- 정적파일 사용하기
- 문의 폼 만들기
- 쿠키
- 세션
- 응답
- Jinja 2 템플릿 엔진 파악 연습문제