

Jinja Template

```
└── main.py
└── templates/
    ├── index.html
    └── bs4_doc.html
    └── bootstrap_doc.html
└── static/
    ├── style.css
    └── script.js
```

작동 흐름 정리 (요청 → 응답)

1. 사용자가 루트 URL (/)에 접속하면 --> GET 요청을 보냄.
2. FastAPI가 비동기 핸들러 (read_root)를 실행.
3. index.html을 찾아 렌더링하고, request 객체를 템플릿으로 사전형 데이터로 전달.
4. HTML 페이지를 클라이언트에게 반환.

```
# 정적 파일 경로 설정
app.mount("/static", StaticFiles(directory="static"), name="static")

# 템플릿 경로 설정
templates = Jinja2Templates(directory="templates")

# @app.get("/", response_class=HTMLResponse)
# async def read_root(request: Request):
#     return templates.TemplateResponse("index.html", {"request": request})

@app.get("/", response_class=HTMLResponse)
async def read_root(request: Request):
    # Request 객체는 클라이언트의 요청 정보
    # - 클라이언트가 FastAPI 서버에 요청을 보낼 때 자동 생성
    # "index.html"을 랜더링하고 templates에 {"request": request}를 전달
    return templates.TemplateResponse("index.html", {"request": request})
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for('static', path='style.css') }}"/>
    <title>FastAPI with Jinja2</title>
<body>
    <h1>Welcome to FastAPI with Jinja2!</h1>
    <a href="/bs4">View BeautifulSoup Example</a>
    <button id="myButton">Click Me!</button>
    <p id="message"></p>
    <script src="{{ url_for('static', path='script.js') }}"></script>

    <!-- 현재 요청된 URL 표시 -->
    <p><strong>Current URL:</strong> {{ request.url }}</p>
```



작동 흐름 정리 (요청 → 응답)

🚀 FastAPI에서 `templates.TemplateResponse()`를 사용할 때는 `request` 객체를 반드시 전달

① 클라이언트 요청

웹 브라우저에서 `http://127.0.0.1:8000/`로 요청

② FastAPI 서버에서 요청 처리

FastAPI의 `read_root(request: Request)` 실행

`request` 객체 생성 및 `index.html` 템플릿에 전달

③ Jinja2 템플릿에서 `request` 객체 사용

`{% request.url %}`을 사용하여 현재 요청된 URL을 HTML에 삽입

④ 클라이언트에게 HTML 응답 반환

FastAPI가 렌더링된 HTML을 응답으로 보내고, 브라우저에서 결과를 표시

