영 진 전 문 대 학 교

글로벌시스템융합과 - GSC

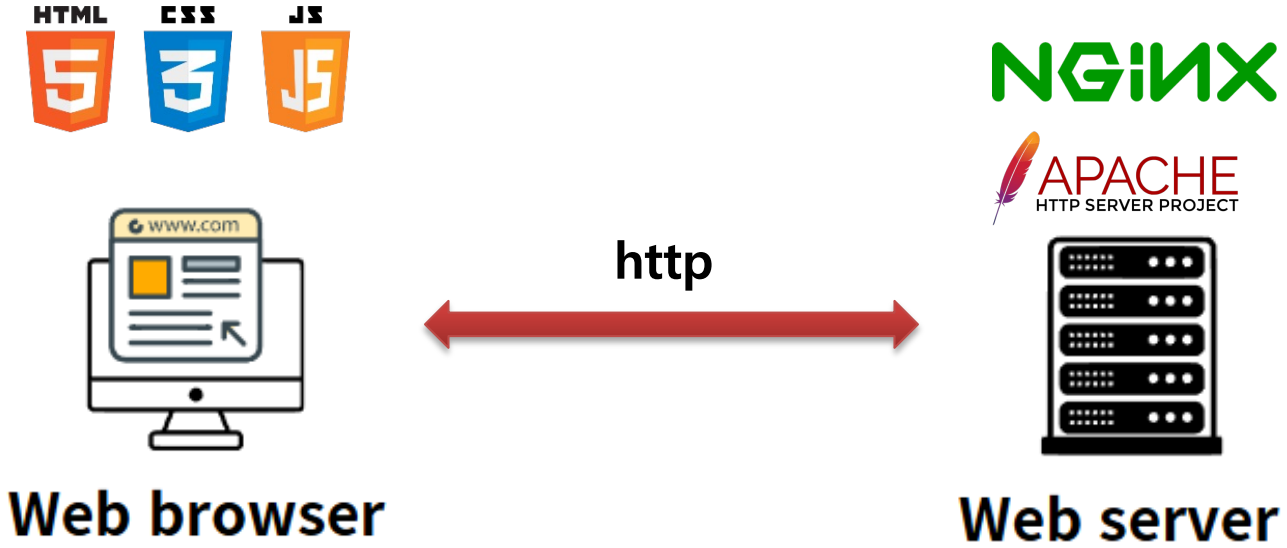
Backend in Web Application

정영철 교수

글로벌시스템융합과

A.I WITH BETTER LIFE

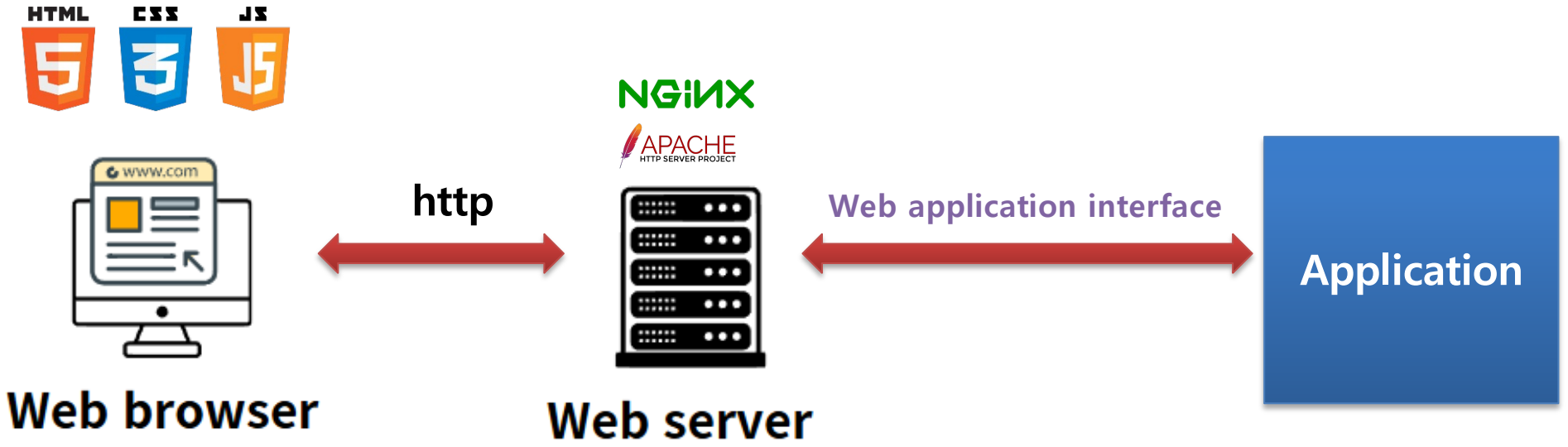
Static Website vs Dynamic Website : Static Website 특징



제공 기능	설명
콘텐츠 제공	• 미리 작성된 HTML 파일 그대로 전송
사용자 상호작용	• JS로 클라이언트에서 처리 가능
페이지 전환	• 여러 HTML 파일을 하이퍼링크로 연결

Static 웹사이트는 백엔드(Back-end) 기술이 필요하지 않음

Static Website vs Dynamic Website : Web Application

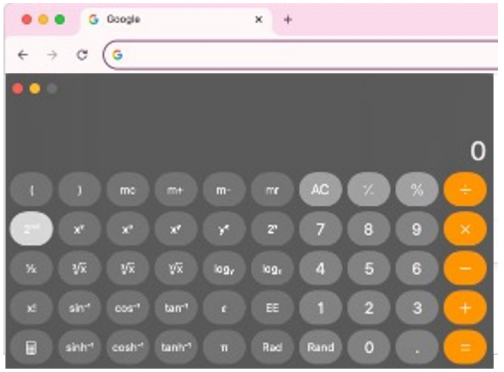


요청에 따라 서버에서
데이터를 처리해 생성

✓ **Application : 특정 목적을 수행하기는 위한 프로그램**

- 예 : 계산기 프로그램

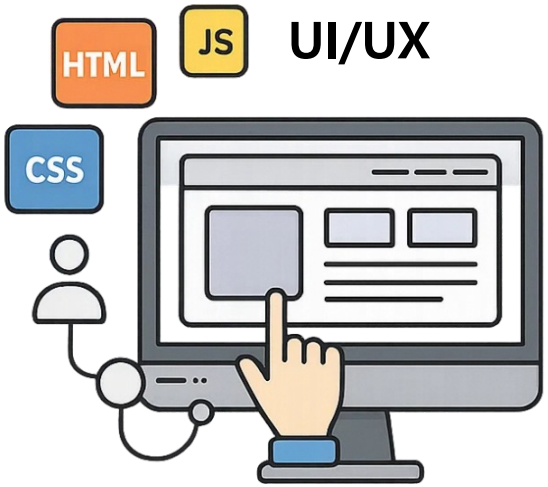
- **Web Application**
  - 예 : Web기반 계산기 프로그램



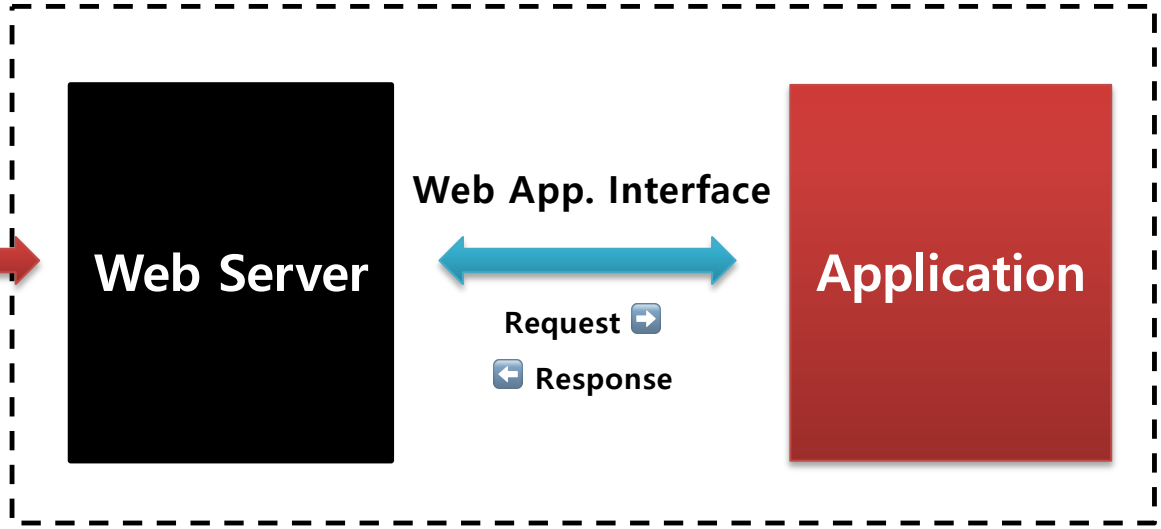
Web Application의 구성 요소 : Front-end and Back-end

Full stack

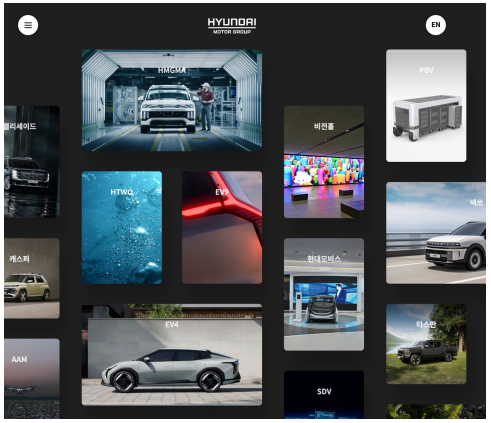
Front-end



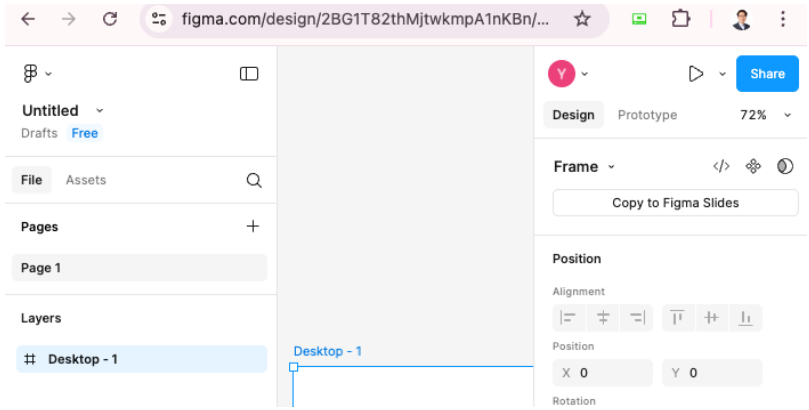
Back-end



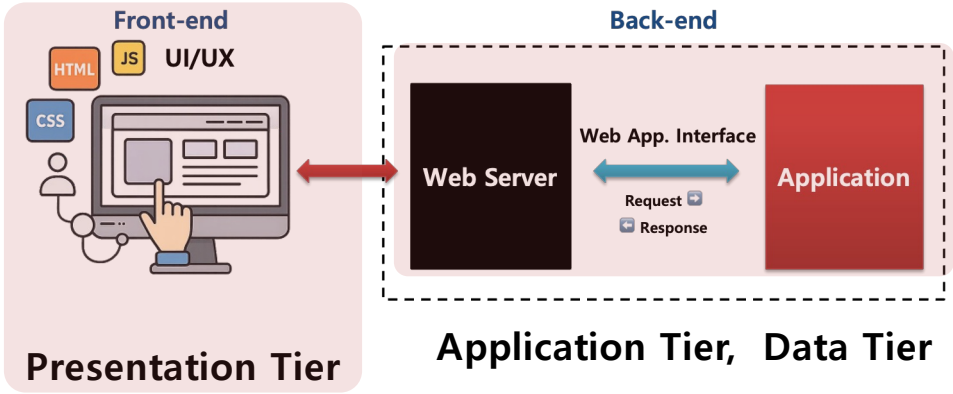
Static web site Only front-end



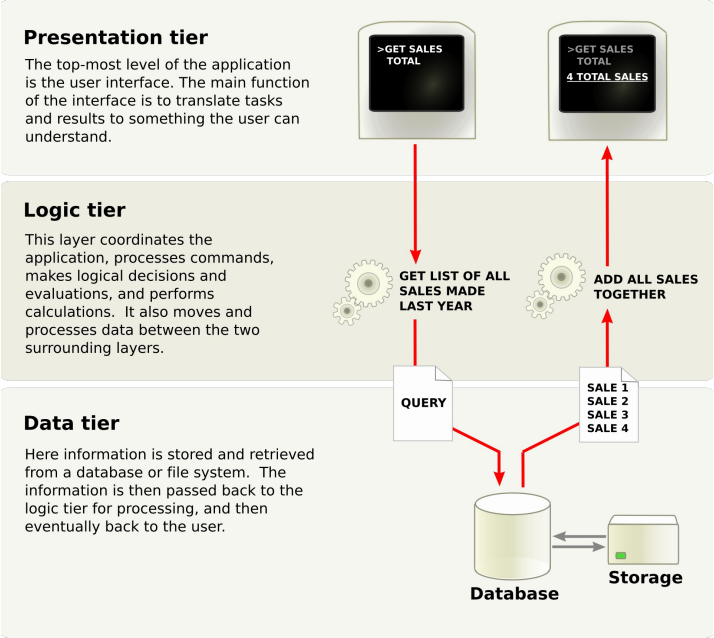
Dynamic web site front-end + back-end



Web Application의 구성 요소 : Software Design Architecture



3 Tier-Architecture (Physical)

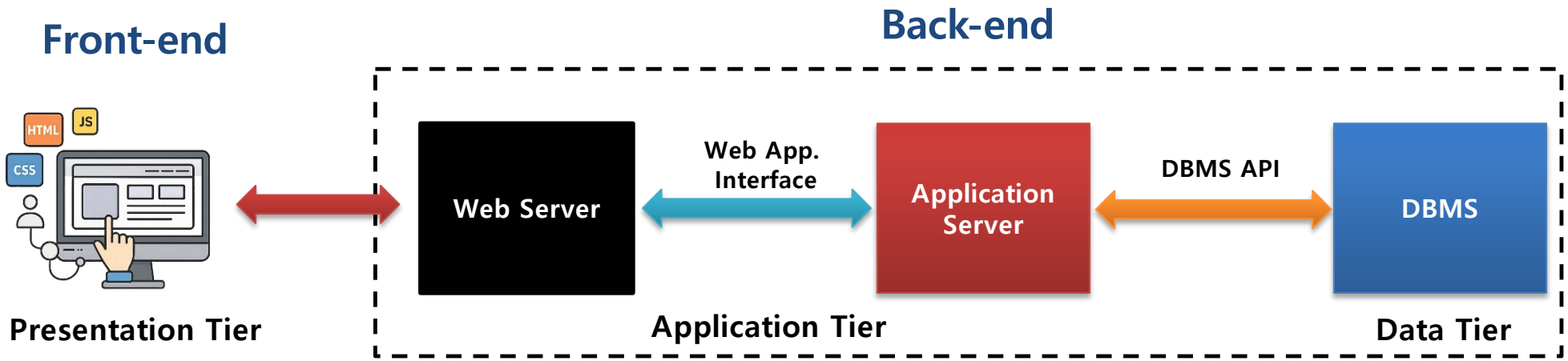


- **Layered Architecture (Logical)**
  - ① **Presentation Layer**
  - ② **Business Logic Layer**
  - ③ **Data Access Layer**

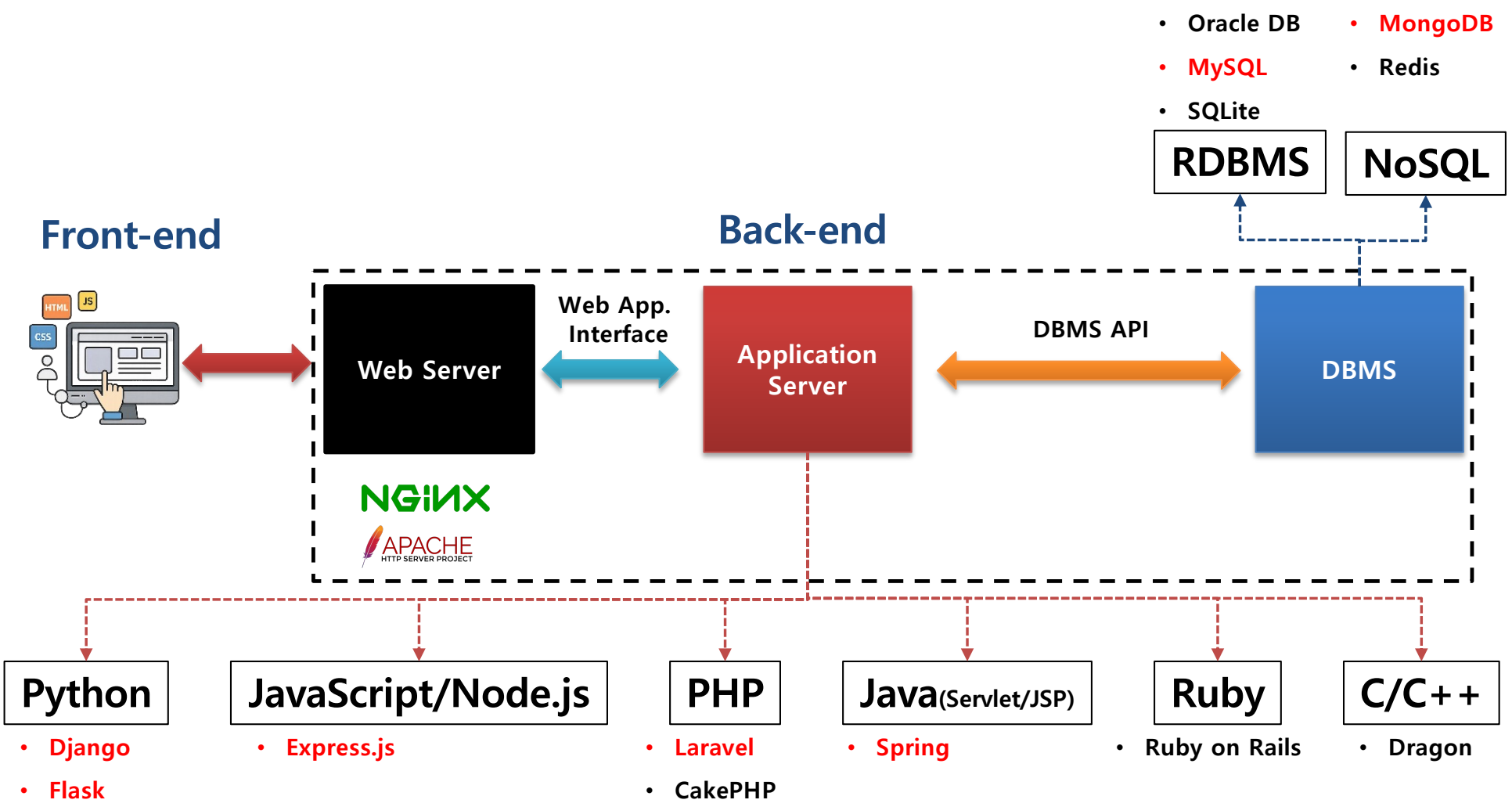
장점	설명
역할 분리	• 사용자 인터페이스, 비즈니스 로직, 데이터 저장 기능을 **기능별로 나눔**
재사용성	• 각 기능은 독립적으로 수정·테스트 가능
유지보수성	• 특정 기능 수정이 전체 시스템에 영향을 덜 줌

Web Application의 구성 요소 : 3-tier Architecture (1)

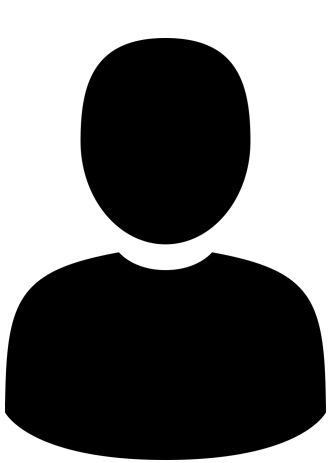
Web application 개발을 위한 일반적인 3-Tier 구조 예



Web Application의 구성 요소 : 3-tier Architecture (2)



Web Application의 특징 : 일반 어플리케이션 관점



사용자

실행 + 입력 값

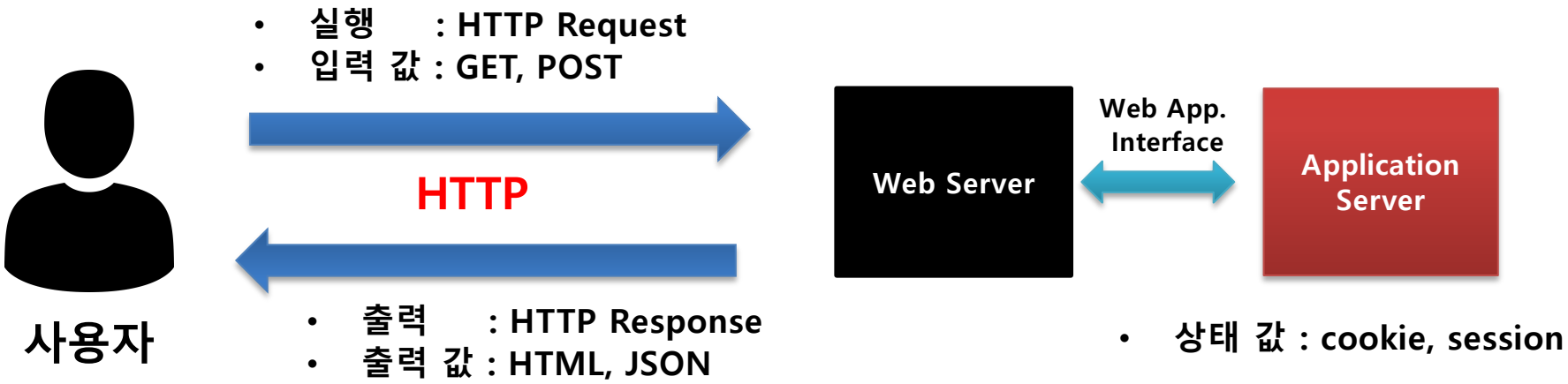
출력 값

Application

상태 값

구분	일반 애플리케이션 (Desktop 등)
실행	- 사용자가 프로그램을 직접 실행 (.exe, .jar, .app) - 실행 이후 프로세스가 지속적으로 유지됨
입력	- 마우스 클릭, 키보드 입력 등 이벤트가 즉시 프로그램 내부로 전달됨 - 예: onClick(), onKeyDown() 등의 이벤트 핸들러 호출
상태	- 상태(예: 변수, 오브젝트)는 메모리에 지속적으로 유지됨 - 프로그램 실행 중 항상 접근 가능
출력	- 그래픽 라이브러리를 통해 즉시 화면에 그려짐 - 또는 파일 등으로 직접 출력됨 (log, txt, json 등)

Web Application의 특징 : Web 어플리케이션 관점



구분	웹 애플리케이션
실행	- 사용자가 브라우저에서 URL 요청 - 요청이 들어올 때마다 서버 측 애플리케이션 코드가 일시적으로 실행됨
입력	- HTTP 요청(GET, POST)을 통해 입력 데이터 전달 - 웹 서버가 요청 수신 및 전처리 후 → 애플리케이션 서버로 전달 - 애플리케이션 서버가 라우터를 통해 적절한 핸들러 함수에 연결
상태	- 웹은 기본적으로 무상태(stateless) - **상태 관리 기능이 추가적으로 필요** - 예) 세션, 쿠키 등
출력	- 서버는 HTML, JSON 등 포맷의 응답 데이터를 생성하여 클라이언트로 전송 - 브라우저가 이를 해석해 화면에 렌더링

웹 어플리케이션 개발을 위한 Backend 학습 순서

단계	내용	비고
1. 해당 언어의 기본 문법	변수, 조건문, 반복문, 함수, 모듈 등	→ 백엔드 언어의 **기초 체력**
2. GET, POST 입력값 획득 방법	라우팅, 파라미터 처리	→ **사용자 요청 처리의 핵심**
3. 쿠키, 세션 사용 방법	상태 유지 방식	→ **HTTP는 무상태(stateless)** → 상태 유지를 위해 필수
4. DBMS 연동 방법	SQL 작성, 연결, 조회 및 삽입 처리	

Back-end Framework

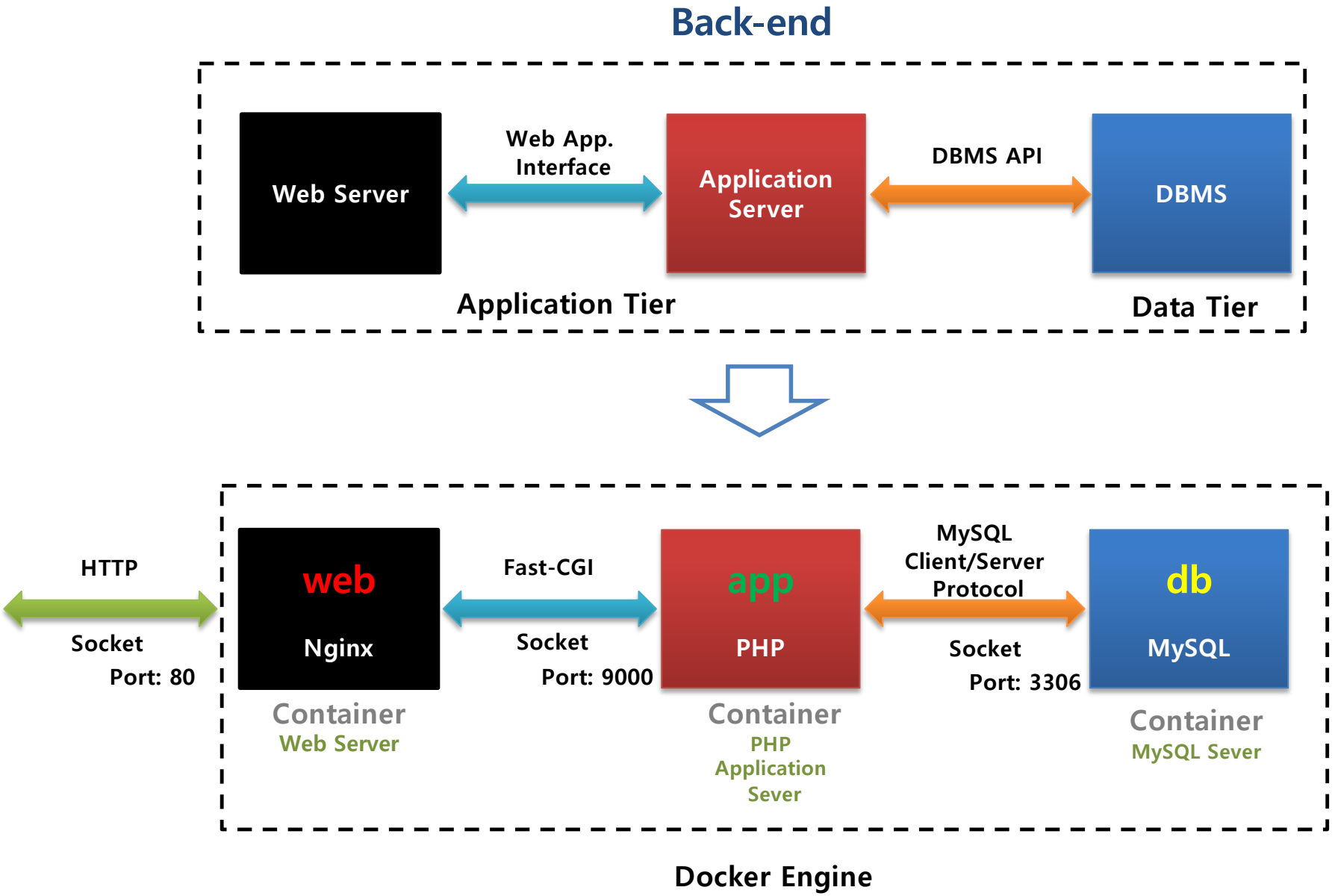
- 라우팅 (자동화된 방식)
- 미들웨어 개념
- 템플릿 렌더링
- 요청/응답 객체
- URL 매핑 및 컨트롤러 구성

Backend 개발 환경 구성 : Docker compose file

```yaml
services:
  ▷ Run Service
  web:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - ./src:/usr/share/nginx/html
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - app

  ▷ Run Service
  app:
    image: php:8.2-fpm
    volumes:
      - ./src:/var/www/html
      - ./docker/php/custom.ini:/usr/local/etc/php/conf.d/custom.ini

  ▷ Run Service
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_PASSWORD}
    volumes:
      - db_data:/var/lib/mysql
      - ./docker/mysql/init.sql:/docker-entrypoint-initdb.d/init.sql

volumes:
  db_data:
```

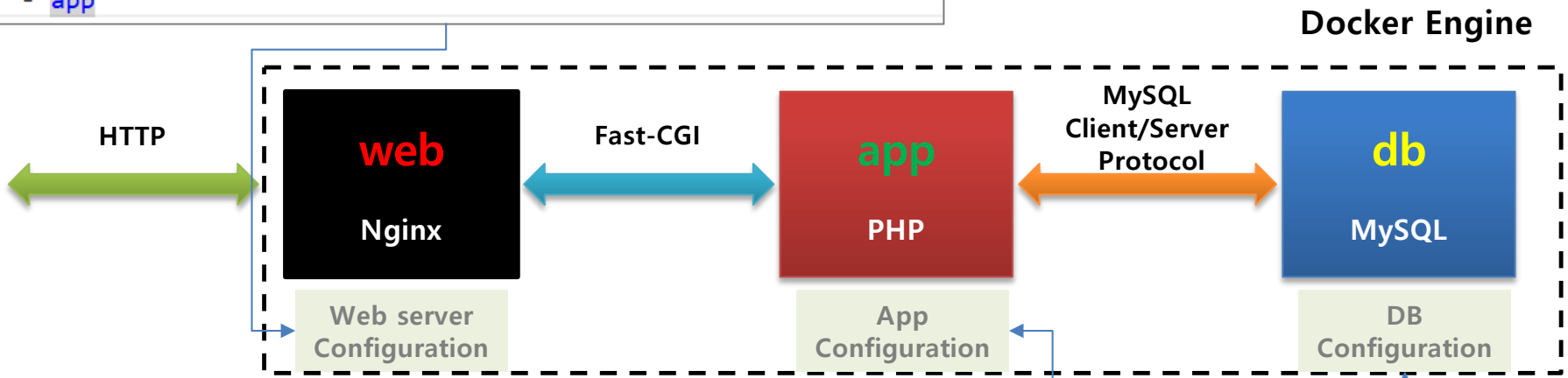
Source codes : https://github.com/gsc-lab/backend

Backend 개발 환경 구성 : Sever & Client 관점



Backend 개발 환경 구성 : 설정 관점

```yaml
web:
  image: nginx
  ports:
    - "8080:80"
  volumes:
    - ./src:/usr/share/nginx/html
    - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
  depends_on:
    - app
```



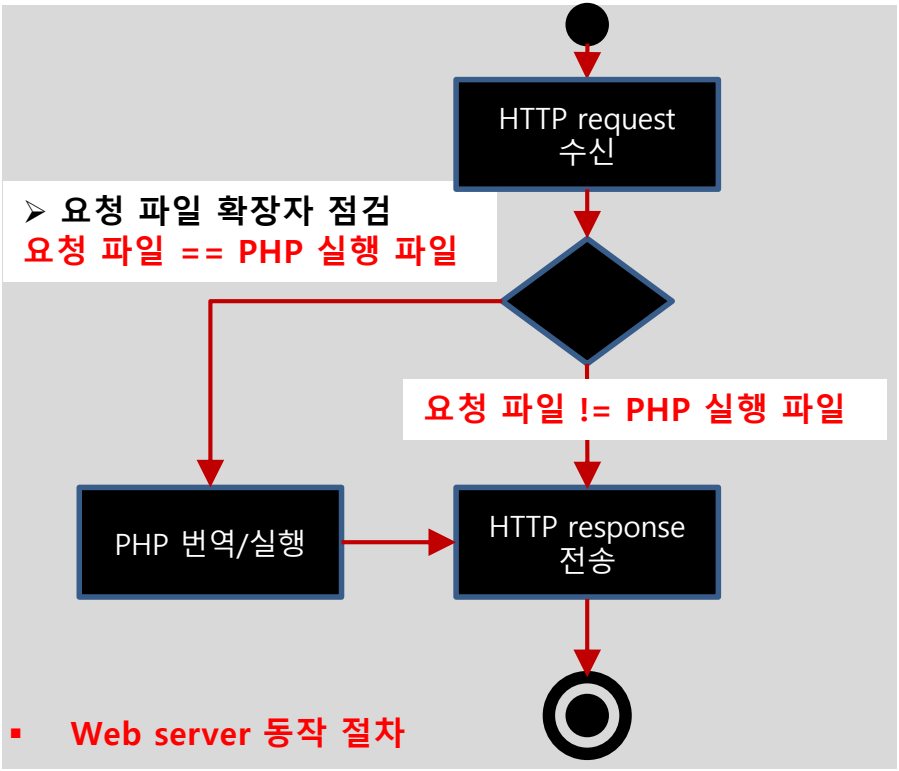

```yaml
app:
  image: php:8.2-fpm
  volumes:
    - ./src:/var/www/html
    - ./docker/php/custom.ini:/usr/local/etc/php/conf.d/custom.ini
```

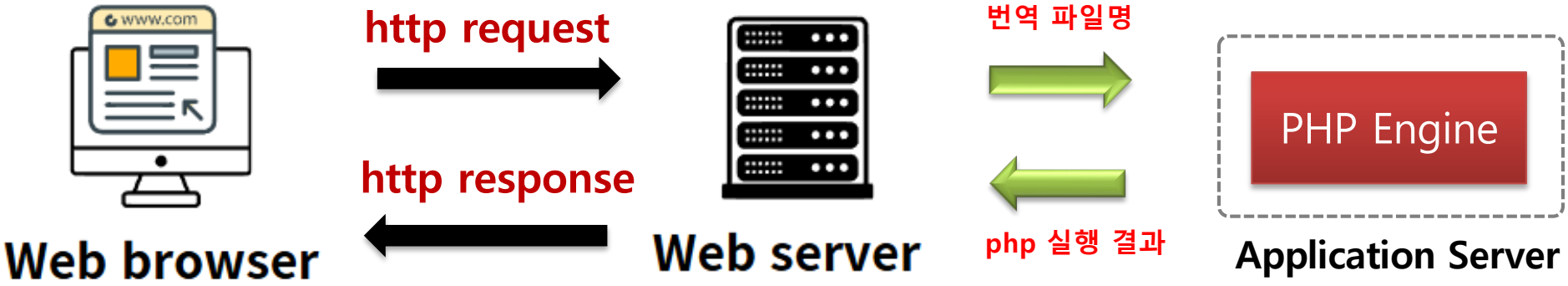
```yaml
volumes:
  db_data:
```

```yaml
db:
  image: mysql:8.0
  environment:
    MYSQL_ROOT_PASSWORD: ${DB_PASSWORD}
  volumes:
    - db_data:/var/lib/mysql
    - ./docker/mysql/init.sql:/docker-entrypoint-initdb.d/init.sql
```

PHP 동작 절차 (1)

- **PHP (Hyper Text Preprocessor)**
  - 웹 어플리케이션을 개발하기 위한 언어
  - Server-side script language
  - Interpreter language





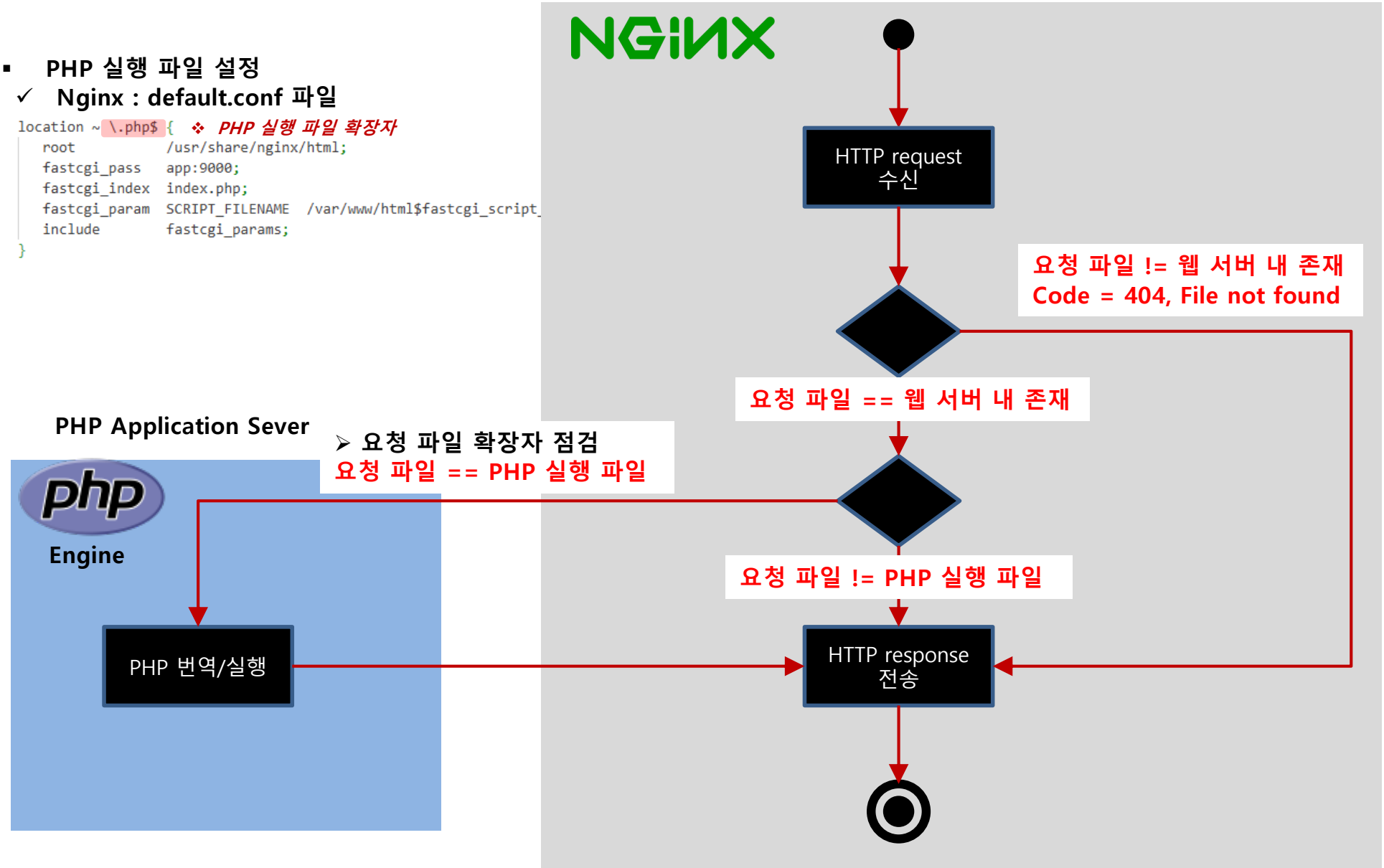
PHP 동작 절차 (2)

- PHP 실행 파일 설정
  - ✓ Nginx : default.conf 파일

```
location ~ \.php$ {
    root           /usr/share/nginx/html;
    fastcgi_pass   app:9000;
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME  /var/www/html$fastcgi_script_
    include        fastcgi_params;
}
```

❖ *PHP 실행 파일 확장자*

- Web server 동작 절차



Extra Slides

대표적인 Web Application Interface 종류

✓ **웹앱 인터페이스(Web Application Interface)**

➢ **웹 서버(Web Server)**와 **웹 애플리케이션(Application)** 사이의 요청(Request)과 응답(Response)을 주고받는 **통신 방식** 또는 연결 구조

방식	설명	주요 사용 환경
• **CGI**	요청마다 외부 프로그램 실행 (매우 비효율적)	초창기 웹
• **FastCGI**	CGI의 성능 개선판 – 프로세스를 재사용함	PHP, 초기 Python
• **WSGI**	Python 웹 앱과 서버 연동 표준 (동기)	Flask, Django
• **ASGI**	WSGI의 비동기 버전	FastAPI, Django Channels
• **Rack**	Ruby 기반 인터페이스	Ruby on Rails
• **Servlet API**	Java EE 기반 표준 인터페이스	Tomcat, Jetty

Q/A

감사합니다

주문식교육의 산실
영진전문대학교