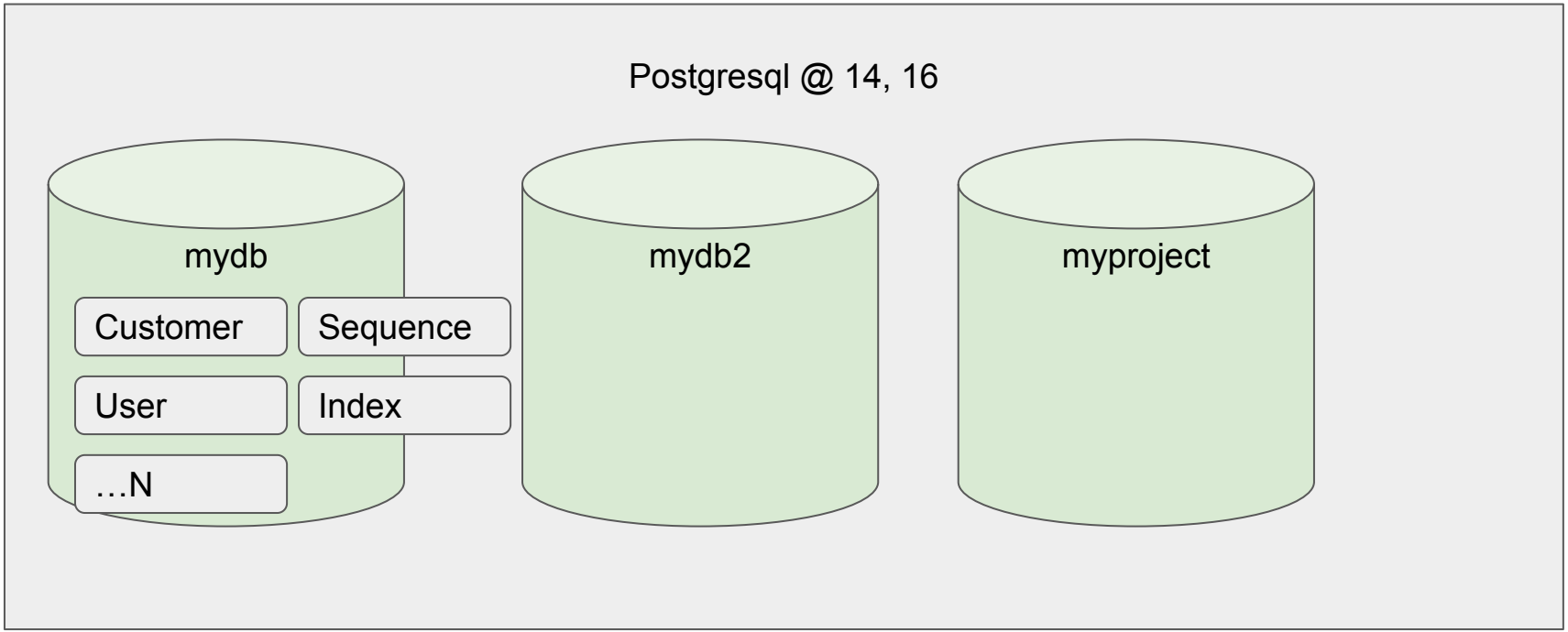
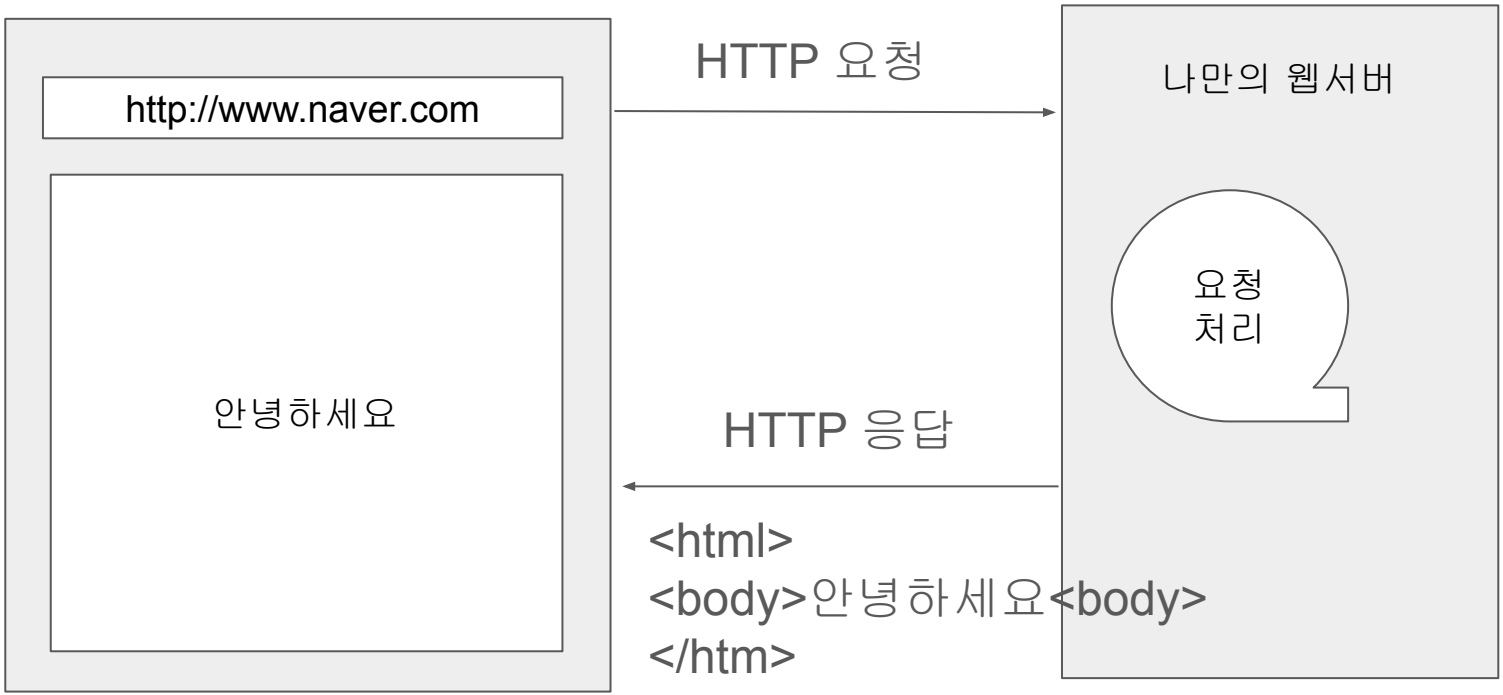
설명 PPT

남부_웹개발실무

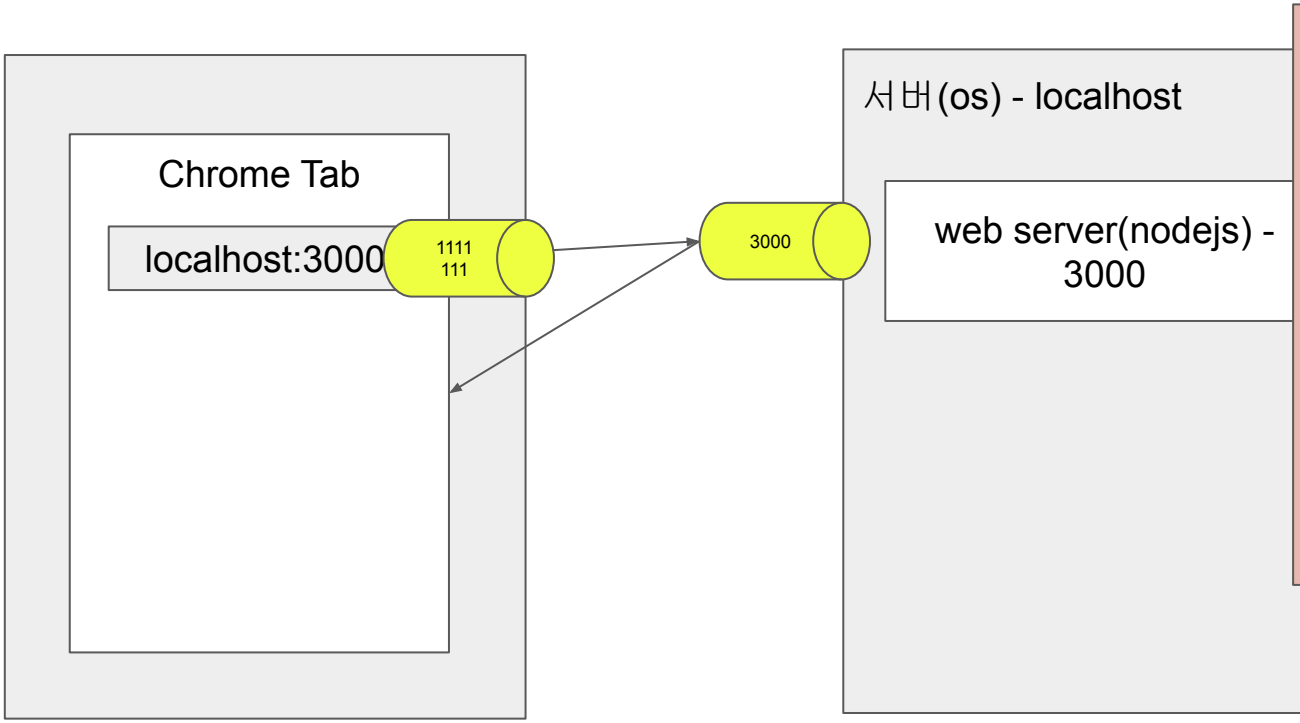
Posgresql


Postgresql @ 14, 16
mydb
Customer
Sequence
User
Index
…N
mydb2
myproject

Web Server


http://www.naver.com
안녕하세요
HTTP 요청
나만의 웹서버
요청
처리
HTTP 응답
<html>
<body>안녕하세요<body>
</htm>

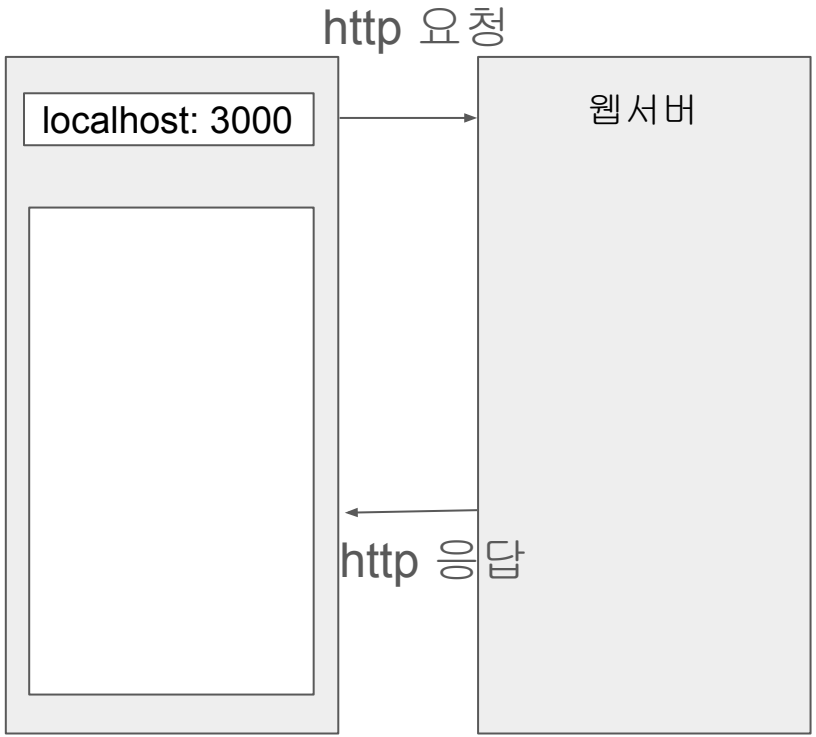
포트 번호



80: 웹서버 기본 포트 http://
443: 보안 웹서버 포트 https://
22: ssh 포트 (원격 터미널 연결용)
21: ftp 포트 (파일 전송 전문)
25: smtp 메일
5432: postgres
3306: mysql

0 ~ 1023 : 시스템 사용포트
1024 ~ 49151: 개발자 사용포트
49152 ~ 655535 : 클라이언트포트

HTTP 요청과 응답



http 요청

항목	의미	예시
주소	무엇을 요청할지	주소?bn=1
헤더	부가정보	인증정보
바디	데이터를 실어보내는 곳	POST, PUT

http 응답

항목	의미	예시
상태코드	성공 실패	
응답헤더	부가정보	ContentType
응답본문	결과데잍터	html , json, text

응답 코드

- 200
  - 200 : 요청이 성공적으로 처리되었어요
  - 201 : 요청이 성공적으로 처리되고, 리소스가 생성됬어요
  - 204 : 요청은 성공했는데 너한테 줄 본문이 없어요.
- 300
  - 301 : 요청한 리소스가 영구적으로 다른데로 갔어요
- 400
  - 400 : 요청이 잘못되어 서버가 요청을 알아들수 없어요
  - 401 : 인증이 필요해요 (로그인 같은게 필요해요)
  - 404 : 요청한 리소스가 없어용
- 500
  - 500 : 인터널 서버에러 서버에거 처리중에 오류가 났어요 (try catch 안해서 발생)
-

URL 설명

https://www.82cook.com:8080/entiz/enti.php?bn=15&bt=23&

https → 프로토콜을 의미

www.82cook.com → 도메인

8080 → 포트번호

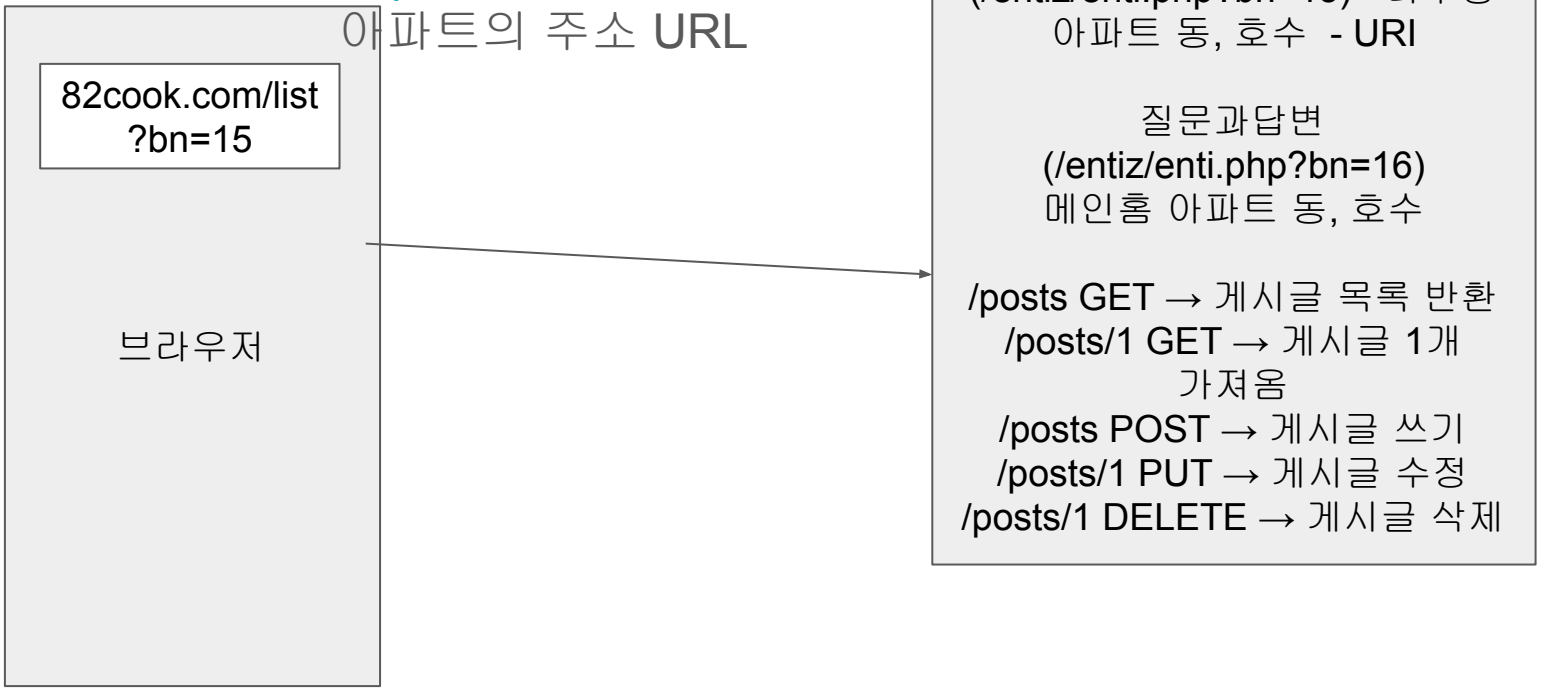
/entiz/enti.php → path , uri

? → 쿼리 파라미터를 시작한다는 의미

bn=15 → bn 키의 값은 15야 의미

URL 과 URI

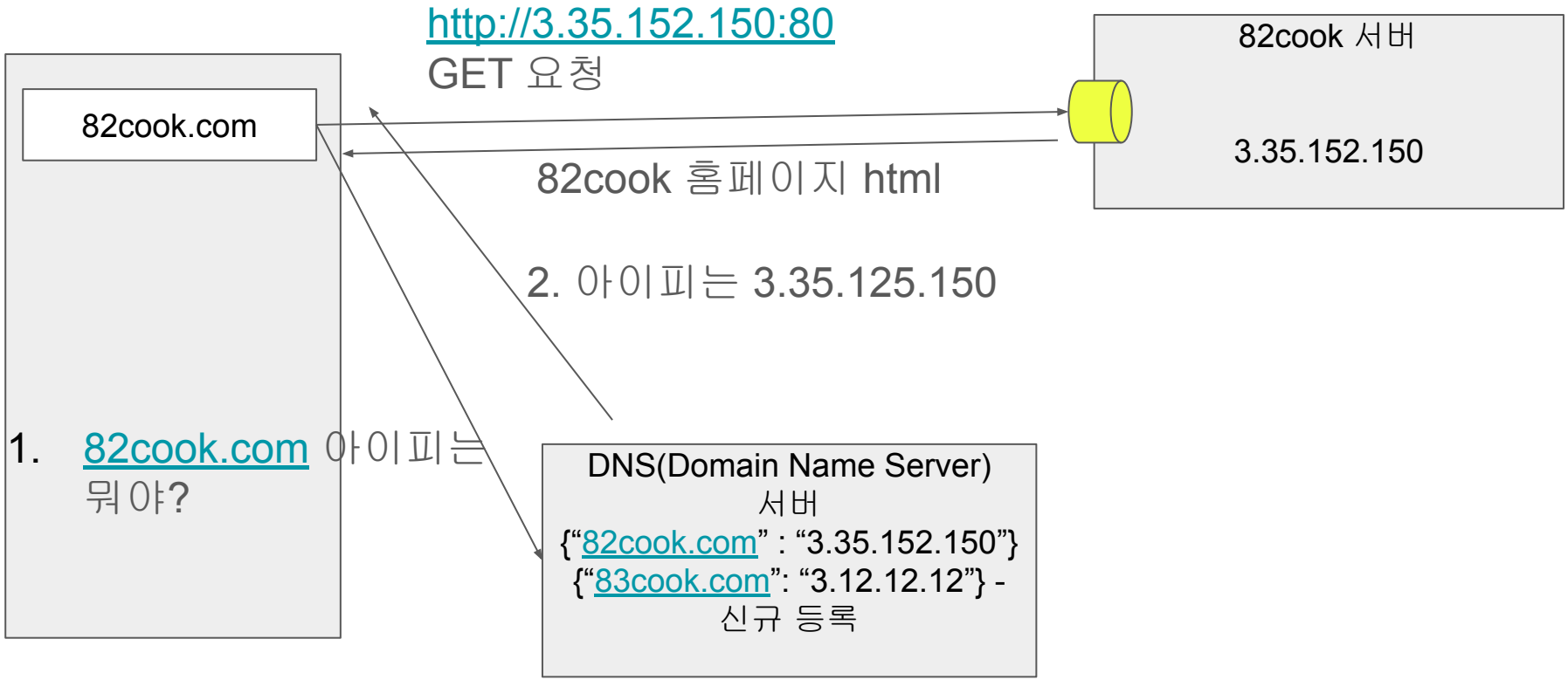
https://www.82cook.com:8000/
아파트의 주소 URL



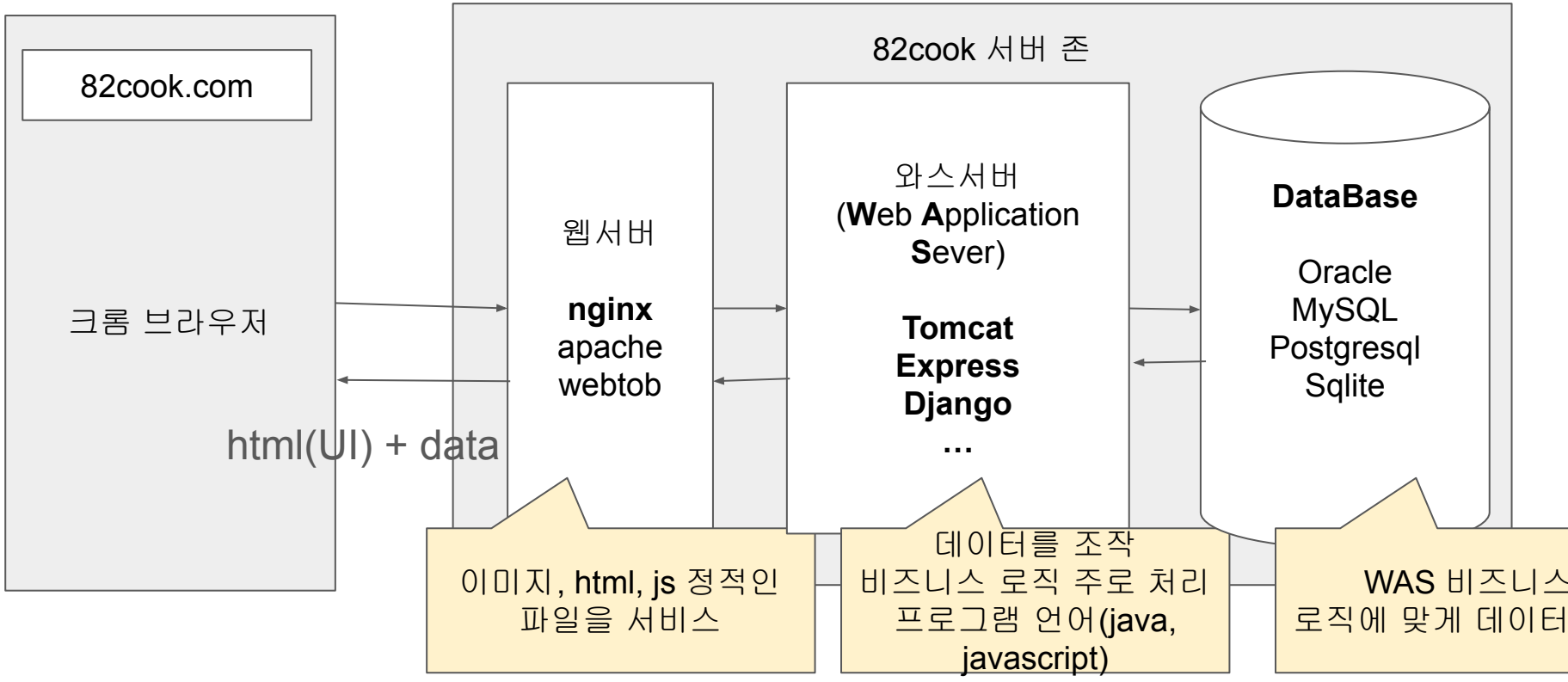
HTTP 메소드

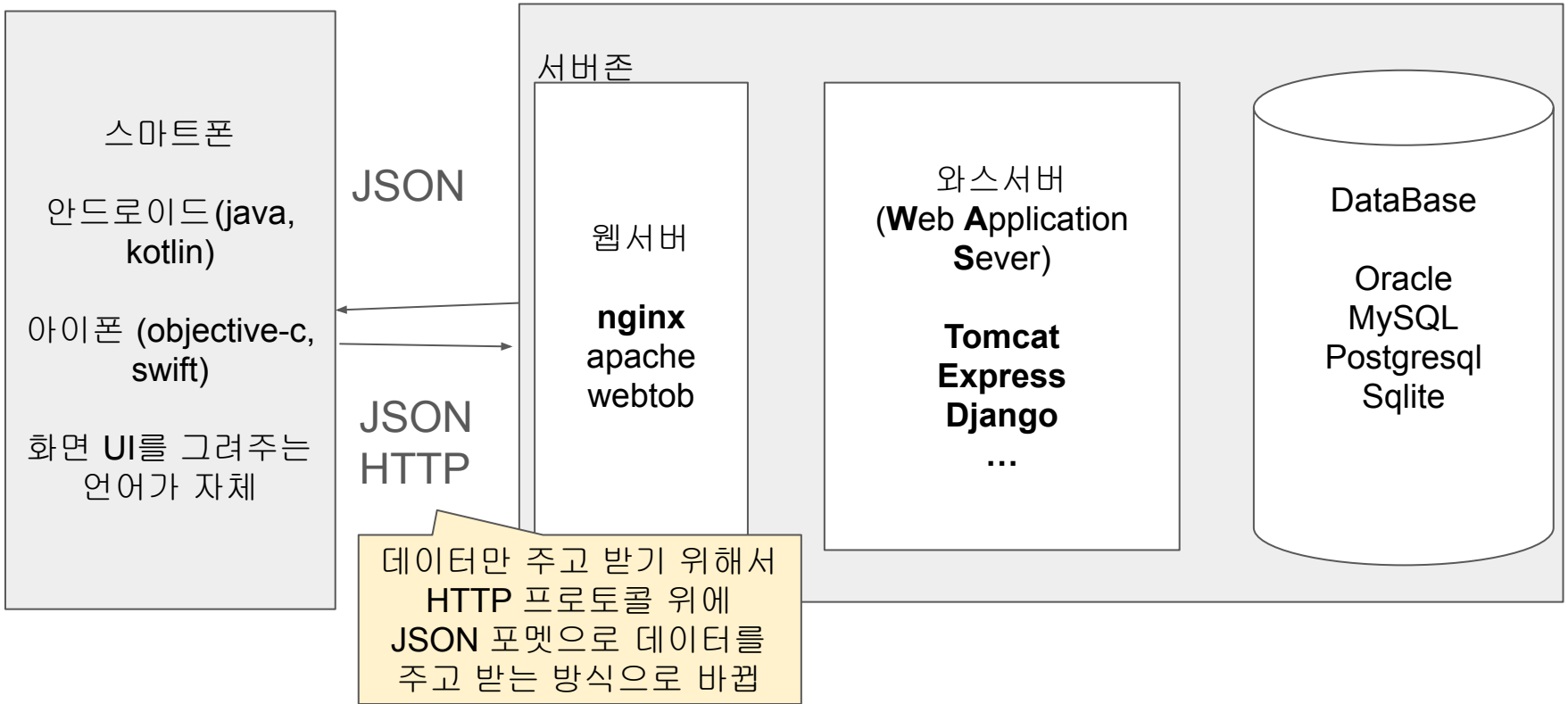
1. GET : 서버에 리소스를 요청합니다.
2. POST : 서버에 데이터를 전송할 때 사용합니다. 클라이언트 → 서버 , 게시글 쓰기
3. PUT : 기존 리소스를 수정할 때 사용합니다.
4. PATCH : 기존 리소스를 부분적으로 수정할 때 사용합니다.
5. DELETE : 기존 리소스를 삭제할 때 사용합니다.

DNS(Domain Name Server) 동작 방식



3티어 서버 구조


82cook.com
크롬 브라우저
html(UI) + data
82cook 서버 존
웹서버
nginx
apache
webtob
와스서버
(Web Application Sever)
Tomcat
Express
Django
...
DataBase
Oracle
MySQL
Postgresql
Sqlite
이미지, html, js 정적인 파일을 서비스
데이터를 조작
비즈니스 로직 주로 처리
프로그램 언어(java, javascript)
WAS 비즈니스
로직에 맞게 데이터

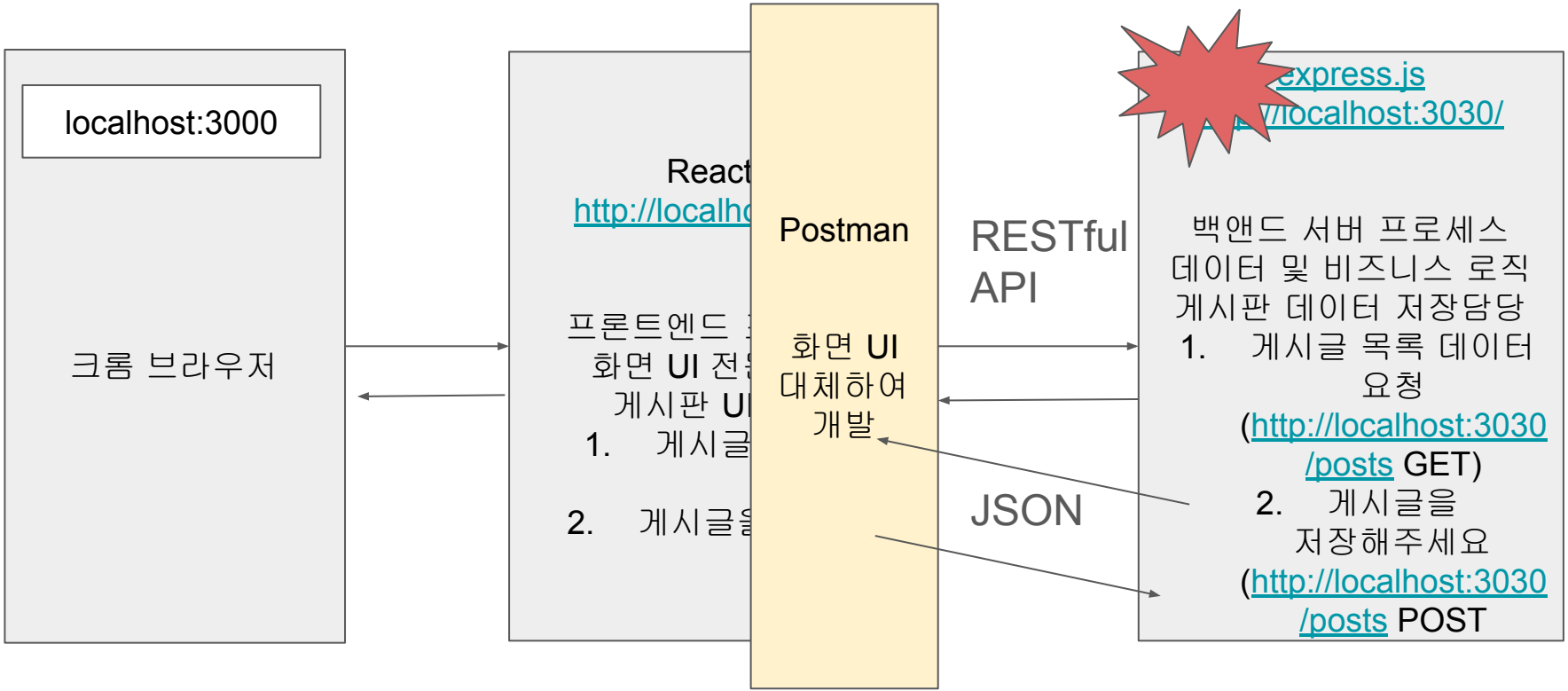

스마트폰
안드로이드(java, kotlin)
아이폰 (objective-c, swift)
화면 UI를 그려주는 언어가 자체
JSON
JSON
HTTP
서버존
웹서버
nginx
apache
webtob
와스서버
(Web Application Sever)
Tomcat
Express
Django
...
DataBase
Oracle
MySQL
Postgresql
Sqlite
데이터만 주고 받기 위해서
HTTP 프로토콜 위에
JSON 포멧으로 데이터를
주고 받는 방식으로 바뀜

RESTful API

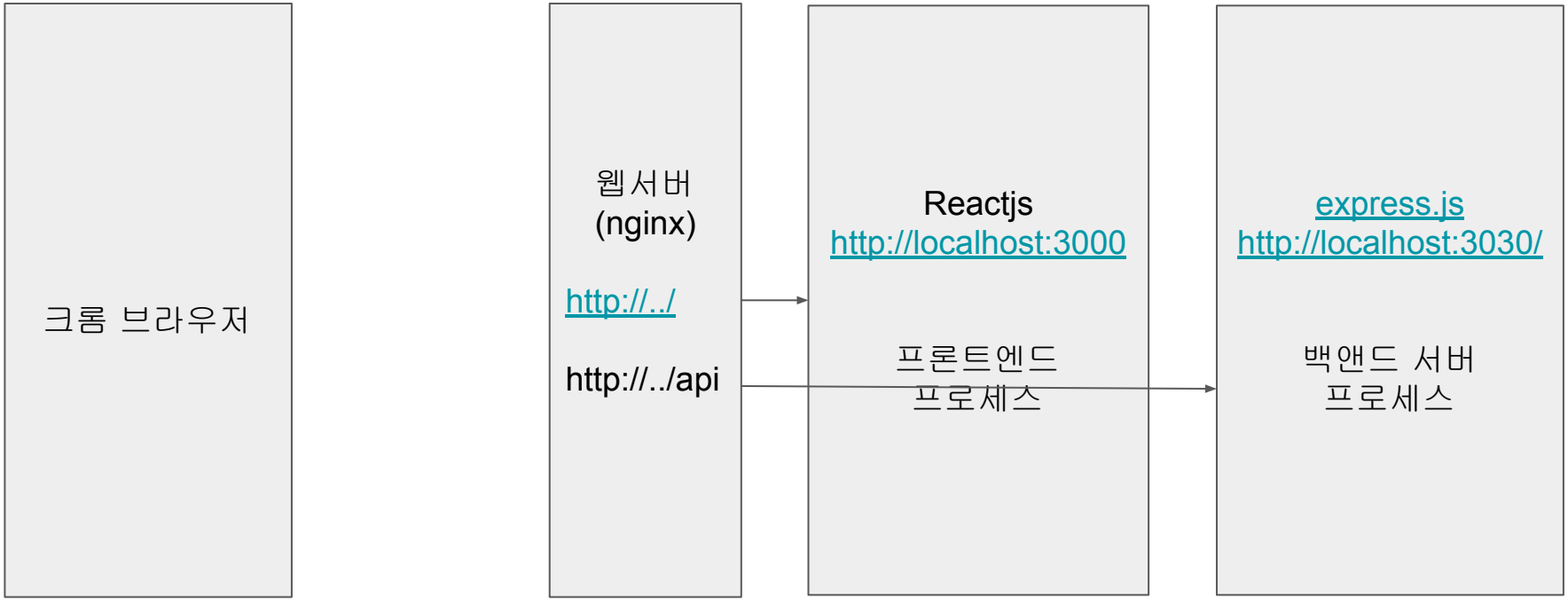
- 포멧 : JSON
- 전송방식 : HTTP
  - GET : 데이터를 읽을때 사용
  - POST : 데이터를 서버에 보낼 때 (클라이언트가, postman/react 소스)
  - PUT : 데이터를 수정할 때 사용
  - DELETE : 데이터를 지울 때 사용

- 게시글 목록 : GET /posts
- 게시글 상세 : GET /posts/12
- 게시글 쓰기 : POST /posts
- 게시글 삭제 : DELETE /posts/12

frontend + backend



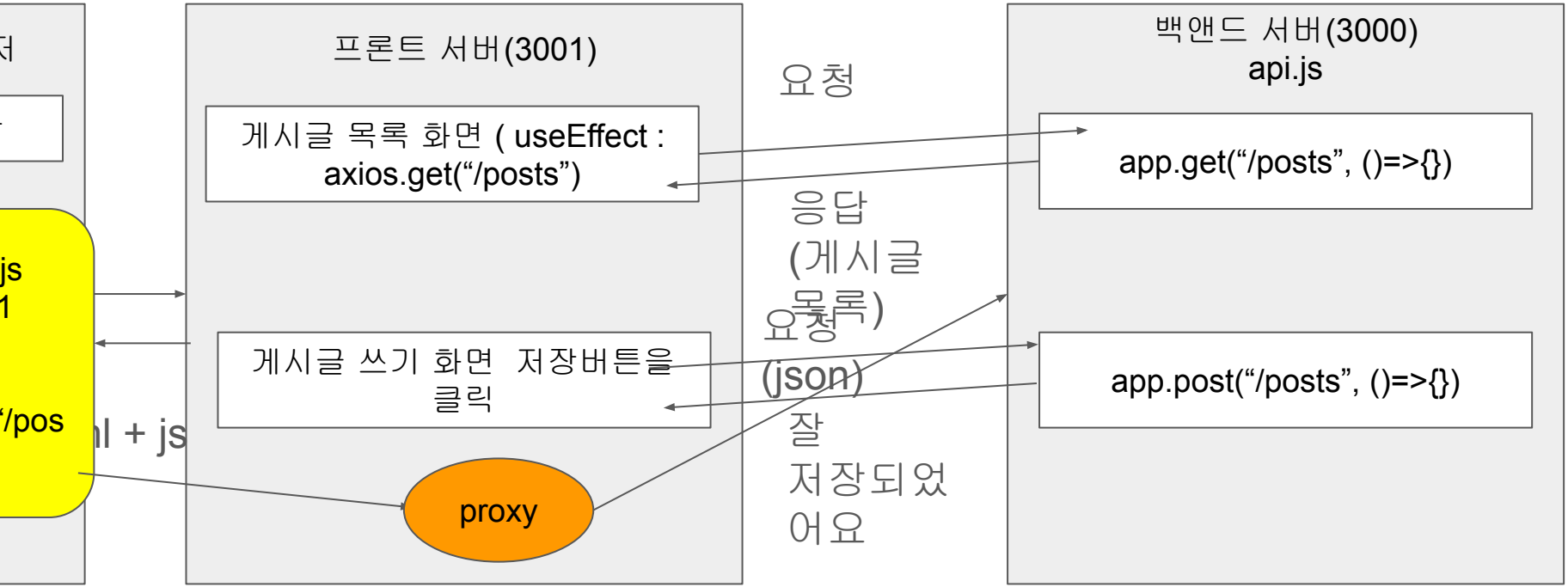
배포했을때(VM 직접배포)



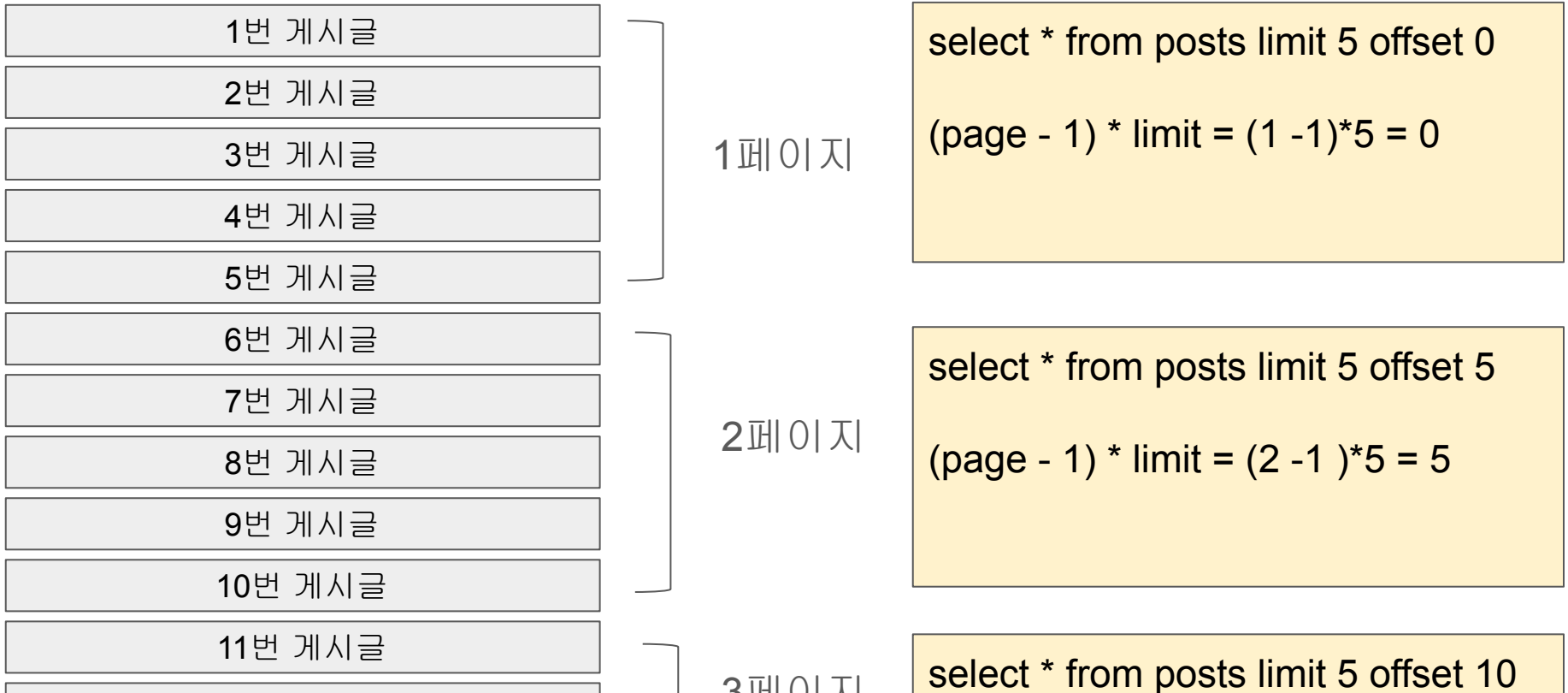
파라미터 설명

https://www.82cook.com:80/entiz/enti.php?bn=10

- https → 프로토콜
- www.82cook.com → 도메인 주소 (IP)
- 80 → 82cook 의 서버 포트 번호
- /entiz/enti.php → path, uri
- ? 뒤에는 query → 서버에 간략한 데이터를 전송하기 위해서 사용 → 크기에 제한(255)
  - ?key1=value1&key2=value2&key3=value3 GET, POST, PUT, DELETE
- POST 요청(Request) : Body 곳에다 넣어서 보냅니다 → 게시글 본문 처럼 큰 사이즈, 첨부파일, 이미지파일


프론트 서버(3001)
게시글 목록 화면 (useEffect :
axios.get(“/posts”)
게시글 쓰기 화면 저장버튼을
클릭
proxy
요청
응답
(게시글
목록)
요청
(json)
잘
저장되었
어요
백앤드 서버(3000)
api.js
app.get(“/posts”, ()=>{})
app.post(“/posts”, ()=>{})

페이지 네이션


1번 게시글
2번 게시글
3번 게시글
4번 게시글
5번 게시글
6번 게시글
7번 게시글
8번 게시글
9번 게시글
10번 게시글
11번 게시글
1페이지
2페이지
select * from posts limit 5 offset 0
(page - 1) * limit = (1 -1)*5 = 0
select * from posts limit 5 offset 5
(page - 1) * limit = (2 -1)*5 = 5
select * from posts limit 5 offset 10

ORM

- Object 를 사용하듯이(자바스크립트 프로그래밍 만으로) SQL을 대체하여 데이터베이스 데이터를 조작하게하는 도구(라이브러리)
- 자바스크립트 : Sequelize, TypeORM, Prisma
- 자바 : JPA, 하이버네이트, MyBatis - (select 문을 전부 별도의 xml 로 모아서 관리)
- 루비 : 액티브 레코드
- ORM도구를 사용해보면 내부적으로는 전부 SQL 변환되어 실행된다 .
- 프로그래머 입장에서 SQL 을 볼 수가 없다

Sequleize

```
db.prepare(`select * from posts`).all()
```

```
const posts = Post.findAll();
```

```
db.prepare(`select * from posts`).get()
```

```
const post = Post.findOne();
```

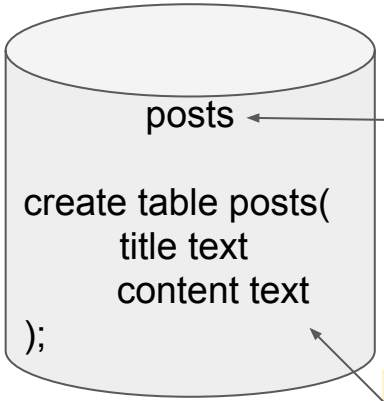
```
db.prepare(`update posts set title = ?
`).get(title)
```

```
const post = Post.update({title: title})
```

Sequelzie

```
db.prepare(`select * from posts`).all()
```

```
const posts = Post.findAll();
```




```
Post → sequelize 사용되는 모델명
const Post = sequelize.define('Post', {
    title : text,
    content : text
})
sequeleize.sync();
```

```
create table posts(
        title text
        content text
);
```

레이어 아키텍처

app.js

POST /posts -> app.post("/posts,()={})

GET /posts -> app.get("/posts,()=>{})

POST /todos -> app.post("/todos,()=>{})

GET /todos -> app.get("/todos,()=>{})

GET /notes -> app.get("/notes,()=>{})

POST /notes -> app.post("/notes,()=>{})

3000줄

app.js

routes

라우터

(/posts)

라우터

(/todos)

라우터

(/notes)

controllers

컨트롤러

GET

POST

PUT

DELETE

(/posts)

컨트롤러

GET

POST

PUT

DELETE

(/todos)

컨트롤러

GET

POST

PUT

DELETE

(/notes)

mode

/pos

/todo

note