

백앤드의 이해

iOS Network Lecture

서버 어플리케이션의 본질

Request를 받아서 Response를 준다.

2017.07.17

최동훈

당신의 커리어 전환점 패스트캠퍼스

IOS PROGRAMMING SCHOOL

프로세스

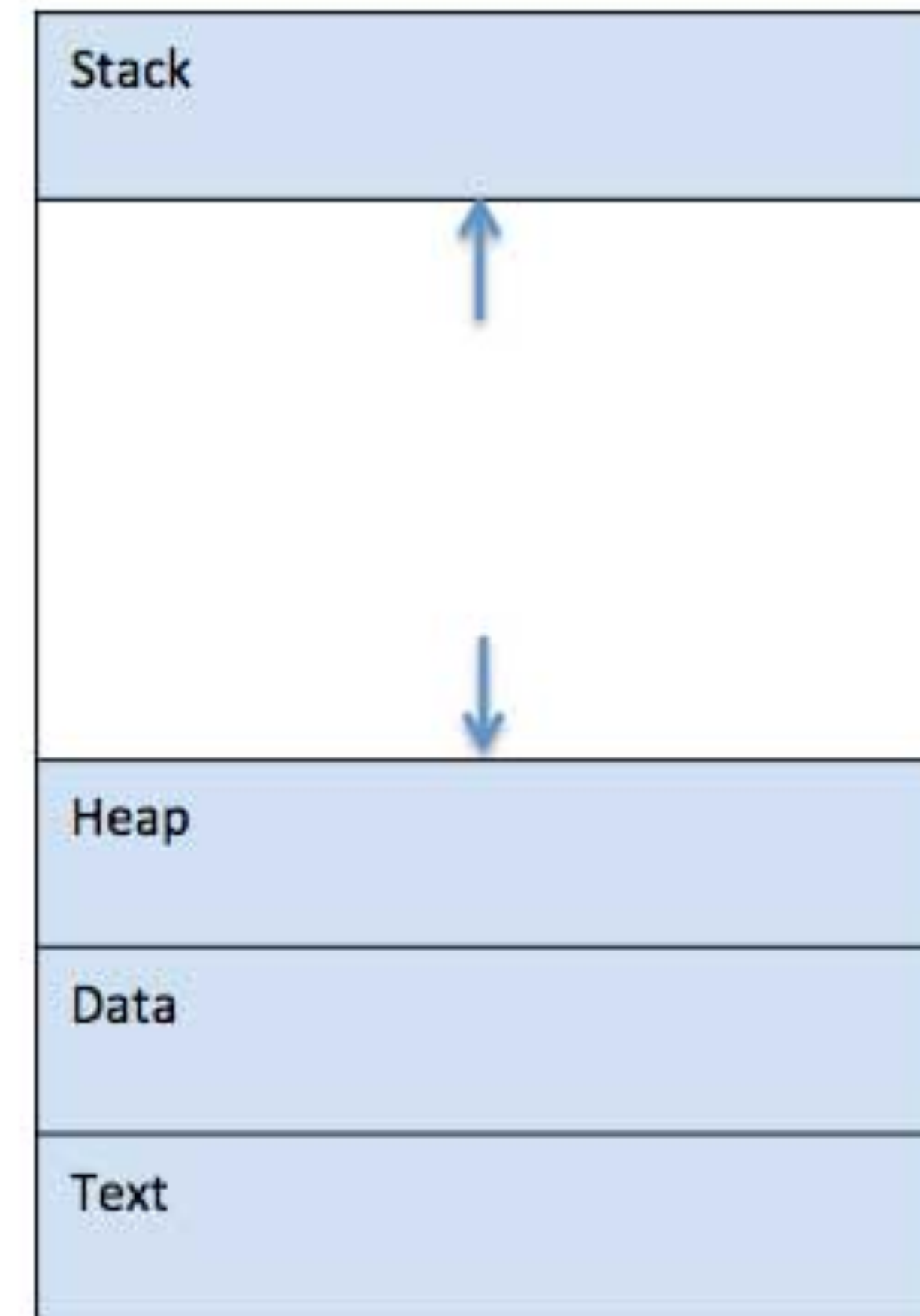
프로세스(process)

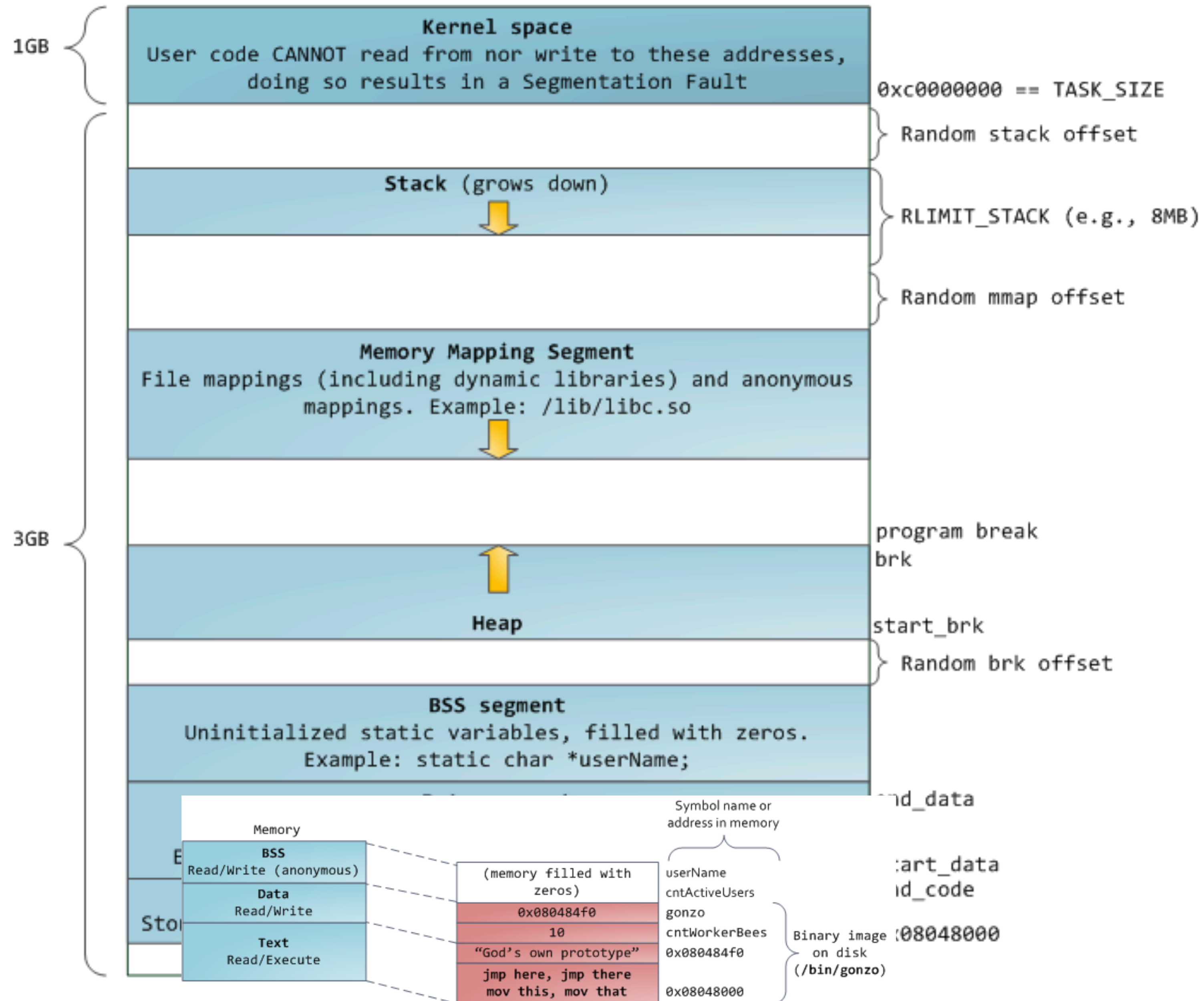
- 실행중인 프로그램
- CPU에 의해 실행되는 컴퓨터 프로그램의 인스턴스
- 운영체제가 관리하는 자원의 대상

(참고: 메모리 및 I/O도 운영체제가 관리하는 관리의 대상)

프로세스 메모리 구조

- Stack Memory (함수에 쓰임)
- Heap Memory (malloc)
- Data, Text (static variable, const variable, code, etc)

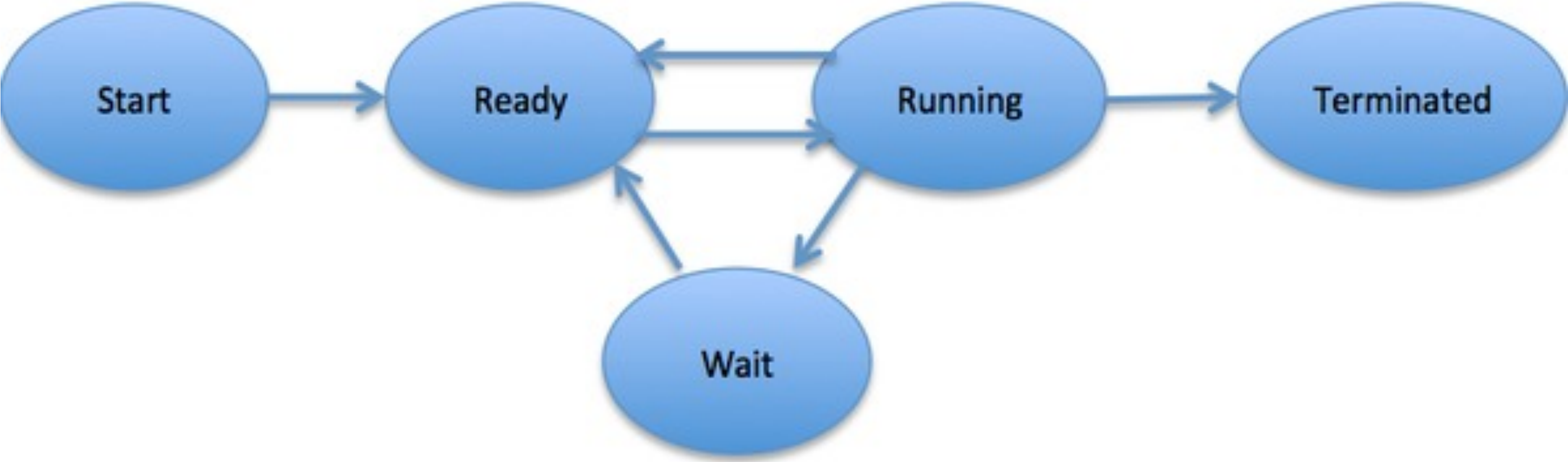




프로세스 제어 블록(PCB)

- 프로세스 상태
- 프로세스 권한
- 프로세스 ID
- 부모 프로세스 카운터
- CPU 레지스터
- CPU 스케줄 관련 정보
- 메모리 관리 정보
- 상태 회계 정보 (CPU 점유율, 실행 시간 등)
- I/O 상태 관련 정보 (어느 Device를 사용중인 지, 어떤 File을 사용중인 지 등)

Pointer	Process state
Process number	
Program counter	
Registers	
Memory limits	
List of open files	
...	



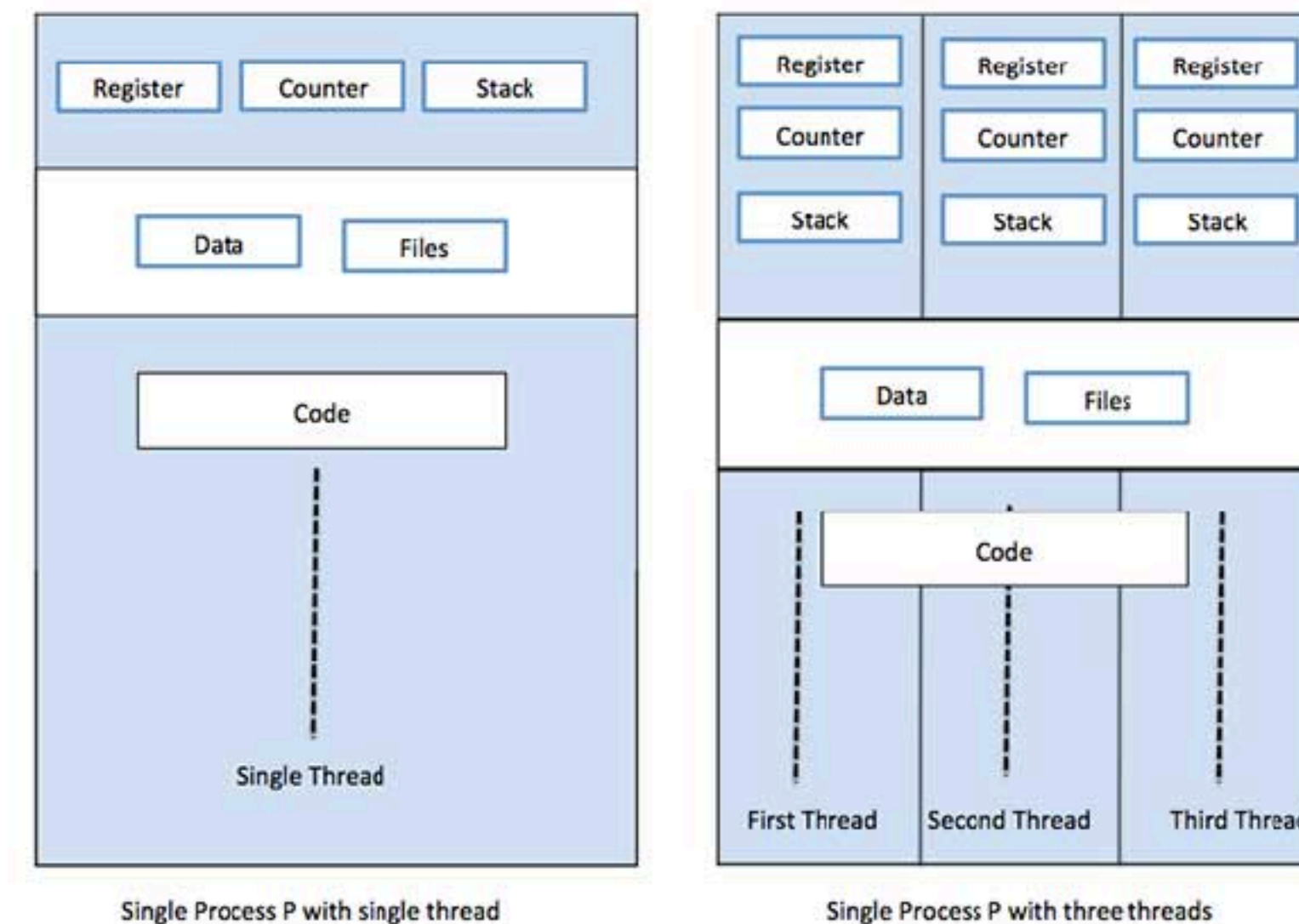
당신의 커리어 전환점 패스트캠퍼스

IOS PROGRAMMING SCHOOL

쓰레드

쓰레드(thread)

- 경량화된 프로세스(a light weight process)라고도 함
- 쓰레드는 프로세스에게 병렬 처리 수단을 제공
- 프로세스는 하나 이상의 쓰레드를 소유하고 있음



Process	Thread
Process is considered heavy weight	Thread is considered light weight
Unit of Resource Allocation and of protection	Unit of CPU utilization
Process creation is very costly in terms of resources	Thread creation is very economical
Program executing as process are relatively slow	Programs executing using thread are comparatively faster
Process cannot access the memory area belonging to another process	Thread can access the memory area belonging to another thread within the same process
Process switching is time consuming	Thread switching is faster
One Process can contain several threads	One thread can belong to exactly one process

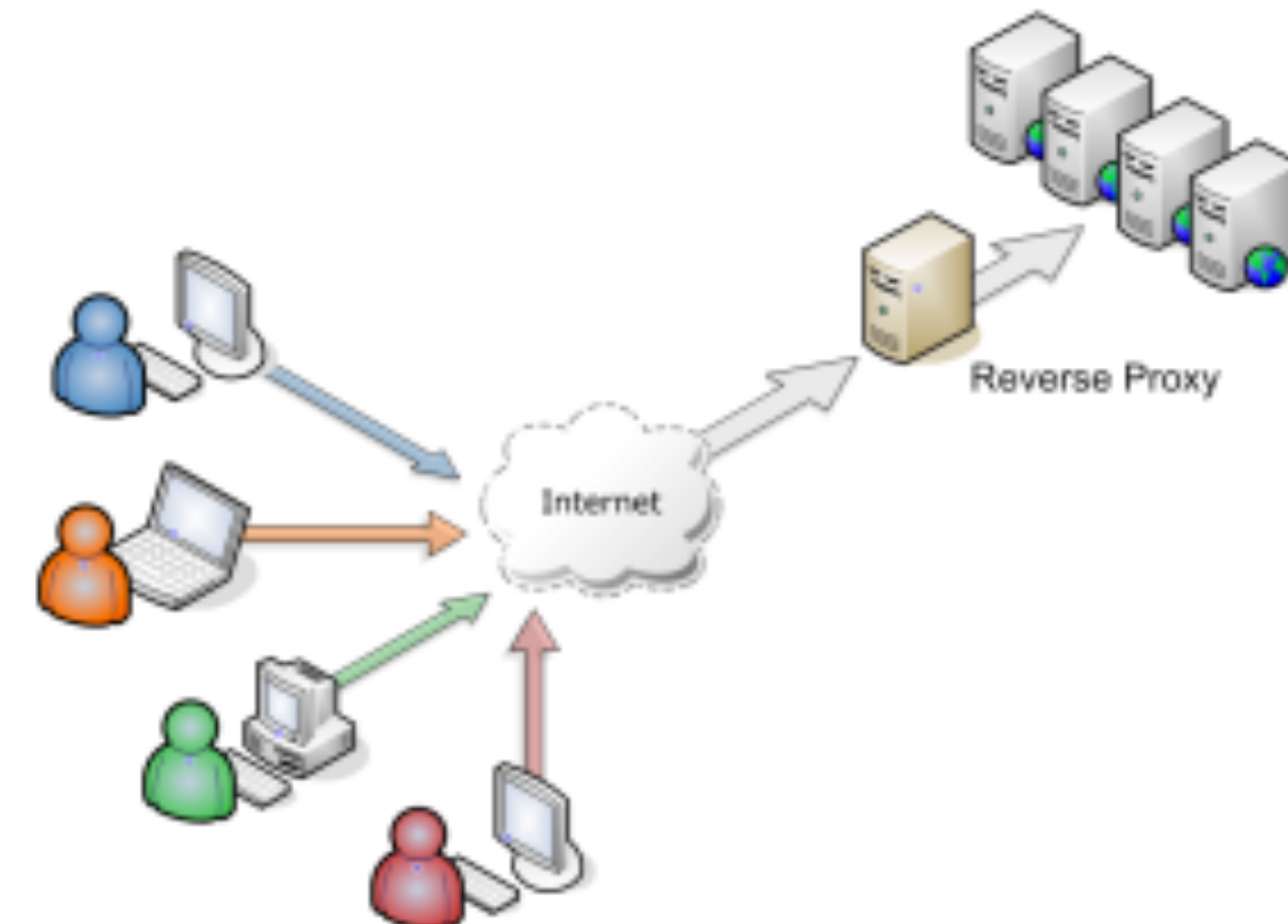
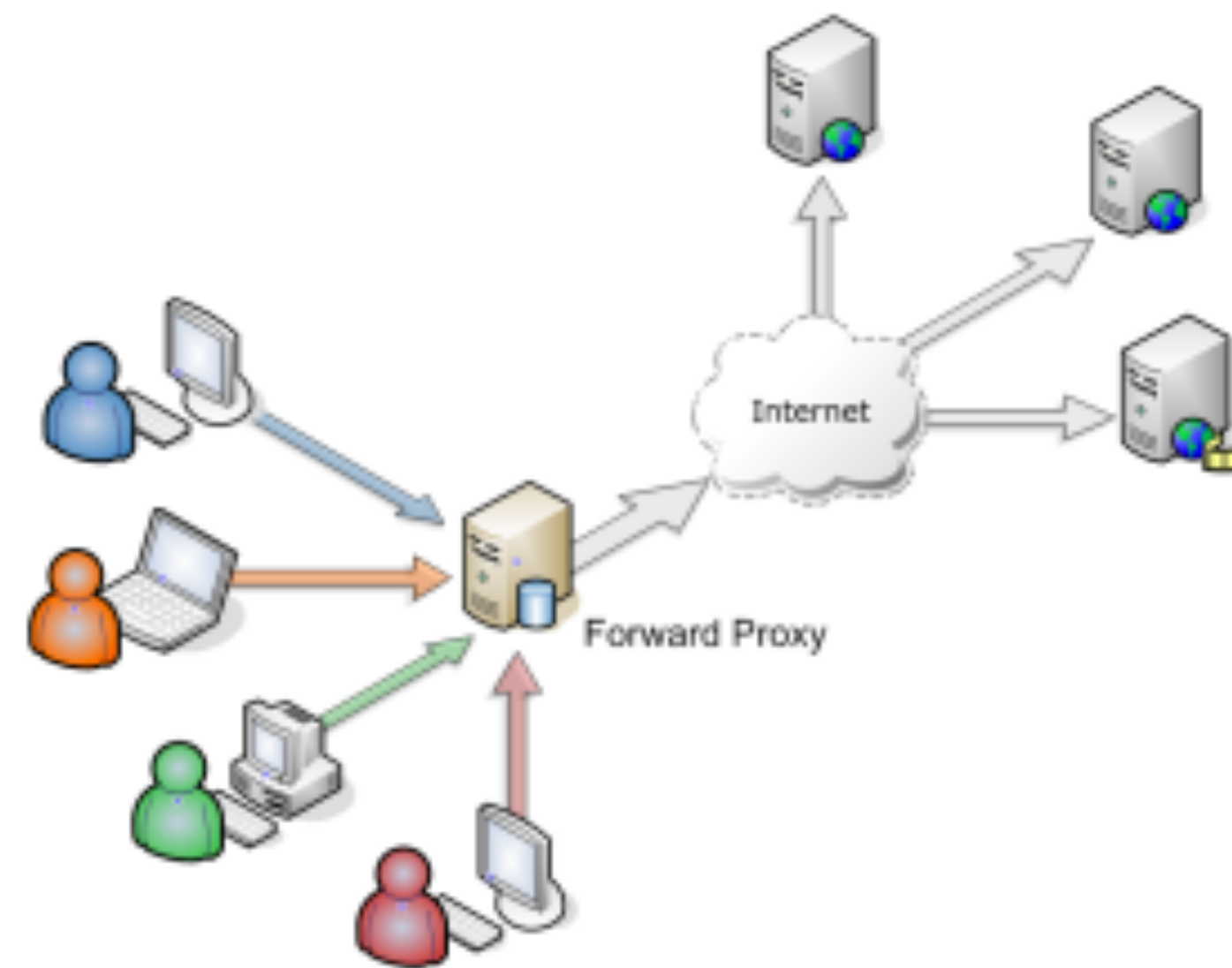
당신의 커리어 전환점 패스트캠퍼스

IOS PROGRAMMING SCHOOL

Web Server vs App Server

Web server

- HTTP 프로토콜을 다룸
- 주로 정적 콘텐츠(static content) 처리
- Forward Proxy와 Reverse Proxy 지원
- nginx, apache



연습문제

- 연습문제 : 주소값을 가지고 서버를 분산시키는 발란서는 L4일까 L7일까?

예) www.fastcompus.co.kr/os -> A서버
www.fastcompus.co.kr/front -> B서버

Application Server

- WAS(Web Application Server)라고도 함
- 비즈니스 로직을 처리
- 주로 동적 콘텐츠 담당
- HTTP 메시지 통신을 주로 하지만 CGI를 사용하기도 함

CGI

(Common Gateway Interface)

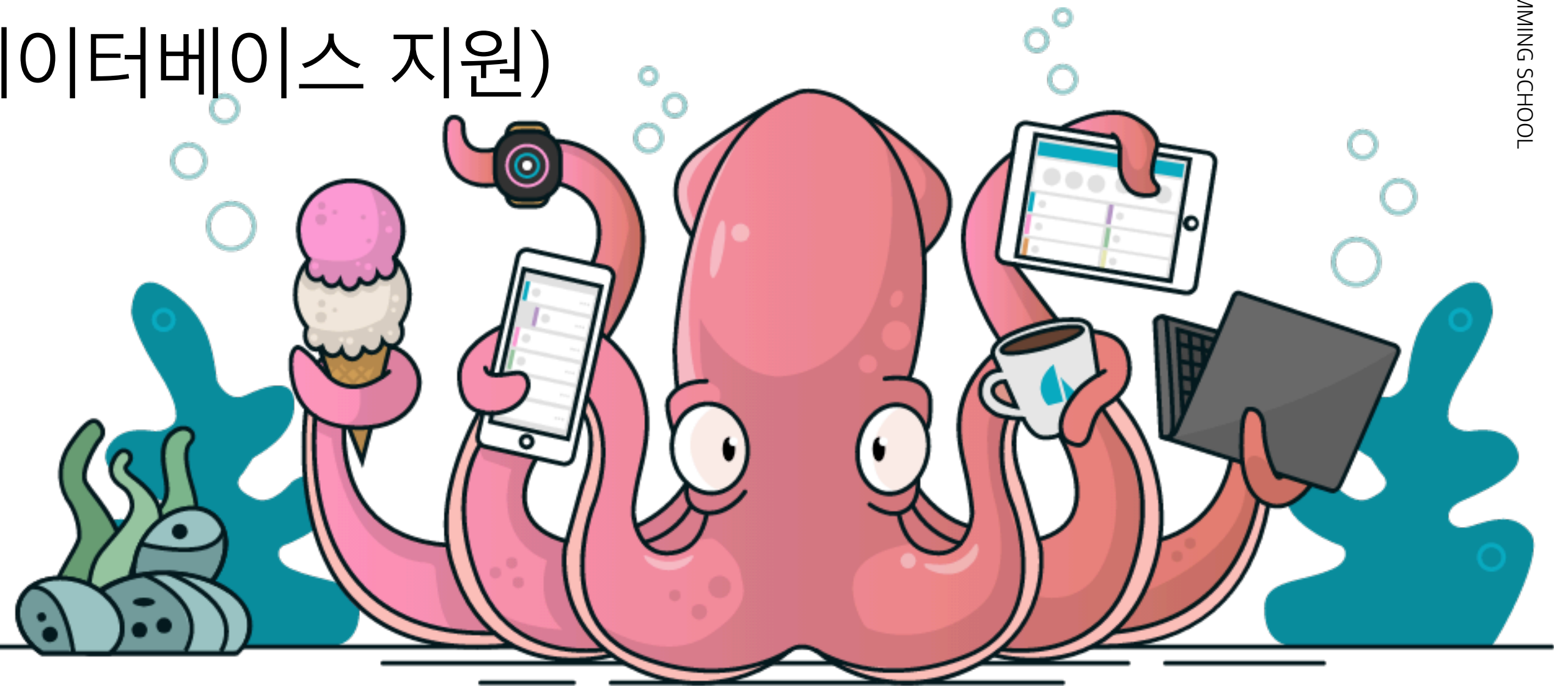
- 웹서버를 위한 표준 프로토콜
- **웹서버**는 CGI를 통해서 **어플리케이션(예: 콘솔 어플리케이션)**을 실행 -> 어플리케이션은 request를 가지고 response를 생성후 response를 웹서버로 전송 -> 이러한 어플리케이션을 CGI script 혹은 CGI라고 함
- perl, php가 대표적인 CGI Language
- Application의 stdout이 웹서버로 전송됨
- python은 wsgi(Web Server Gateway Interface)
- ruby는 rack(Ruby web server interface)
- node.js는 one thread application으로 gateway interface가 필요 없음 (하지만 굳이 cgi 방식으로 돌리겠다는 node.js cgi 라이브러리가 존재함)

하나의 request packet을 입력으로
하나의 response packet을 출력으로 하는
process or thread or function으로 매칭

실습 : Restful API 제작

Sails.js

- <http://sailsjs.com/>
- 100% 자바스크립트 full stack web framework
- Waterline ORM (모든 데이터베이스 지원)
- Express 기반
- REST API 자동 생성
- Socket.IO 웹소켓 내장



RESTful API

정의 : request 메서드의 동사를 사용하여 데이터를 CRUD 하는것

GET /boat -> boat 테이블의 목록을 제공해줌 /boat/find 와 동일

GET /boat/:id -> boat 테이블의 특정 id를 갖는 데이터를 제공 /boat/find/:id 와 동일

POST /boat -> boat 테이블에 데이터를 추가 /boat/create와 동일

PATCH /boat/:id -> boat 테이블에 특정 id를 갖는 데이터를 수정 /boat/update/:id

DELETE /boat/:id -> boat 테이블에 특정 id를 갖는 데이터를 삭제 /boat/destory/:id

장점 : 편하다 (규약은 설정보다 편하다)

단점 : 보안에 취약 할 수 있음

실습 - sails js 및 프로젝트 설치

sails js 설치

\$ npm install sails@beta -g

sails project 생성

\$ sails new hello

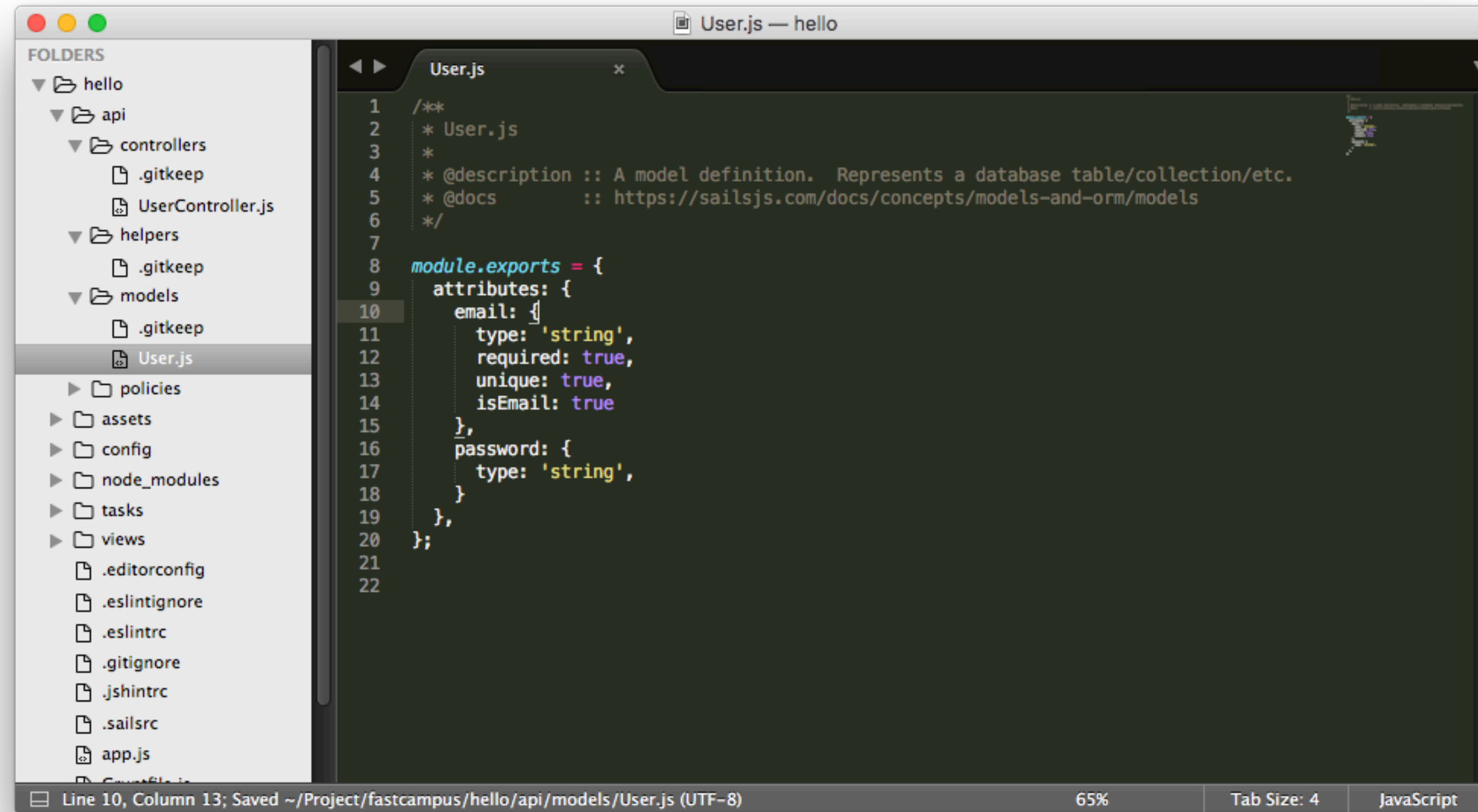
sails api 생성

\$ cd hello

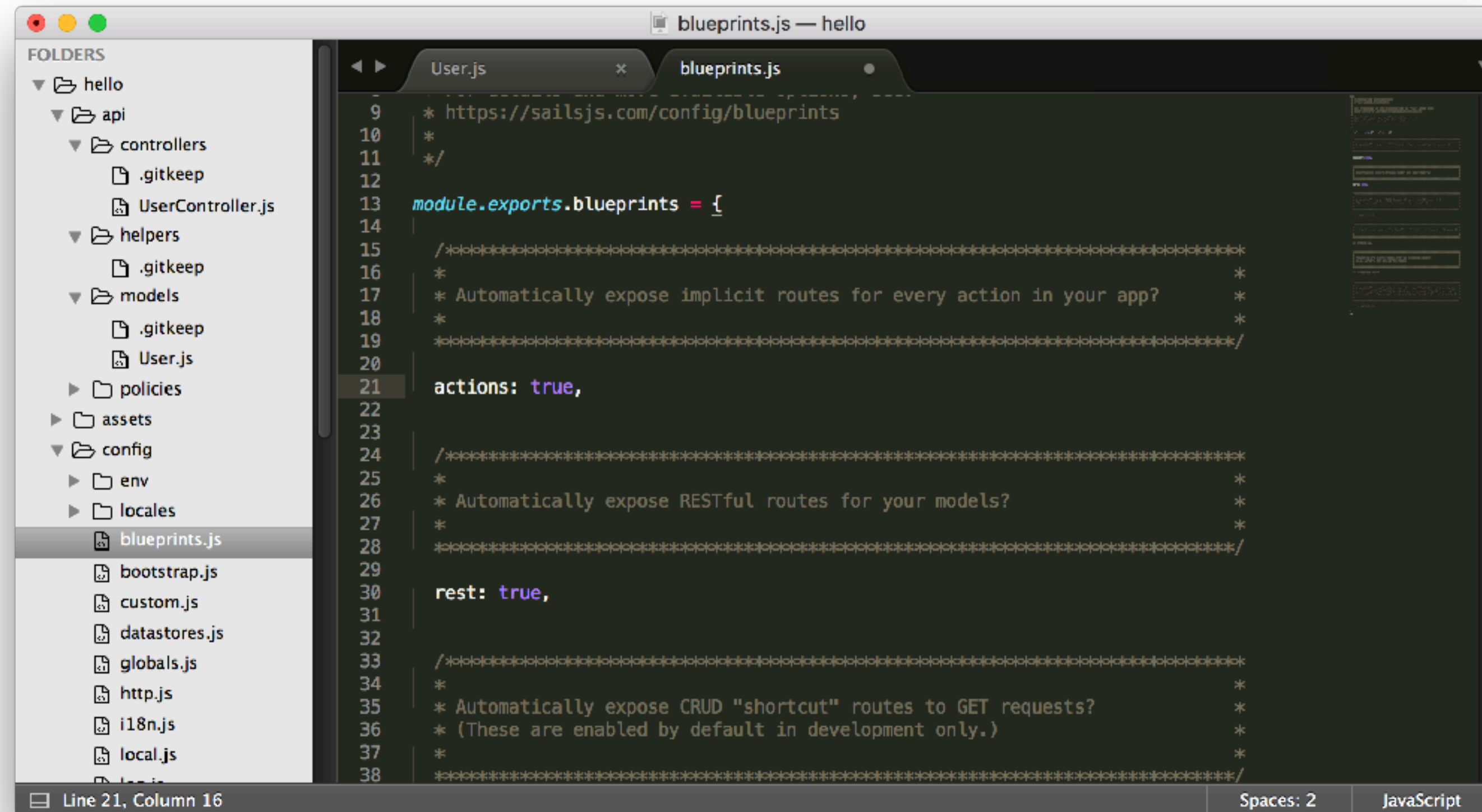
\$ sails generate api User

```
1. ubuntu@ip-10-10-1-79: ~/easyscan (bash)
sapsaldogs-MacBook-Pro:easyscan_old sapsaldog$ cd ~/Project/
sapsaldogs-MacBook-Pro:Project sapsaldog$ ls
IFA2016          cloudgate          stackserver        leeuam_architecture
sailsdoc         easyscan           test              leeuam_se-mil-ga-gui
sendic           gmsref             unitycookbook_translate mombrush
algorithm        smilebuy           i-synapse         moveit
arduino          smsmodule          xcodetest
sapsaldogs-MacBook-Pro:Project sapsaldog$ mkdir fastcampus
sapsaldogs-MacBook-Pro:Project sapsaldog$ cd fastcampus/
sapsaldogs-MacBook-Pro:fastcampus sapsaldog$ ls
sapsaldogs-MacBook-Pro:fastcampus sapsaldog$ sails new hello
info: Installing dependencies...
Press CTRL+C to cancel.
(to skip this step in the future, use --fast)
info: Created a new Sails app `hello`!
sapsaldogs-MacBook-Pro:fastcampus sapsaldog$ cd hello/
sapsaldogs-MacBook-Pro:hello sapsaldog$ ls
Gruntfile.js README.md  api      app.js    assets    config    node_modul
es package.json tasks    views
sapsaldogs-MacBook-Pro:hello sapsaldog$ sails generate api User
info: Created a new api!
sapsaldogs-MacBook-Pro:hello sapsaldog$
```


실습 - 모델 수정



실습 - REST API 활성화



실습 - 서버 실행

```
2. node
info: ✓ Auto-migration complete.
info:
info:      .-...-
info:
info:  Sails      <|      .-...-
info:  v1.0.0-36  | \
info:            /  \
info:          /    \
info:        /      \
info:      /        \
info:    /          \
info:  /            \
info: /              \
info:/            \
info: \            /
info:  \          /
info:   \        /
info:    \      /
info:     \    /
info:      \  /
info:       \/
info:      ---
info:  ---
info:
info: Server lifted in `~/Users/sapsalldog/Project/fastcampus/hello`
info: To shut down Sails, press <CTRL> + C at any time.

debug: -----
debug: :: Fri Jul 07 2017 19:40:46 GMT+0900 (KST)

debug: Environment : development
debug: Port        : 1337
debug: -----
```

\$ sails lift

실습 - 결과 확인

웹 브라우저 접속

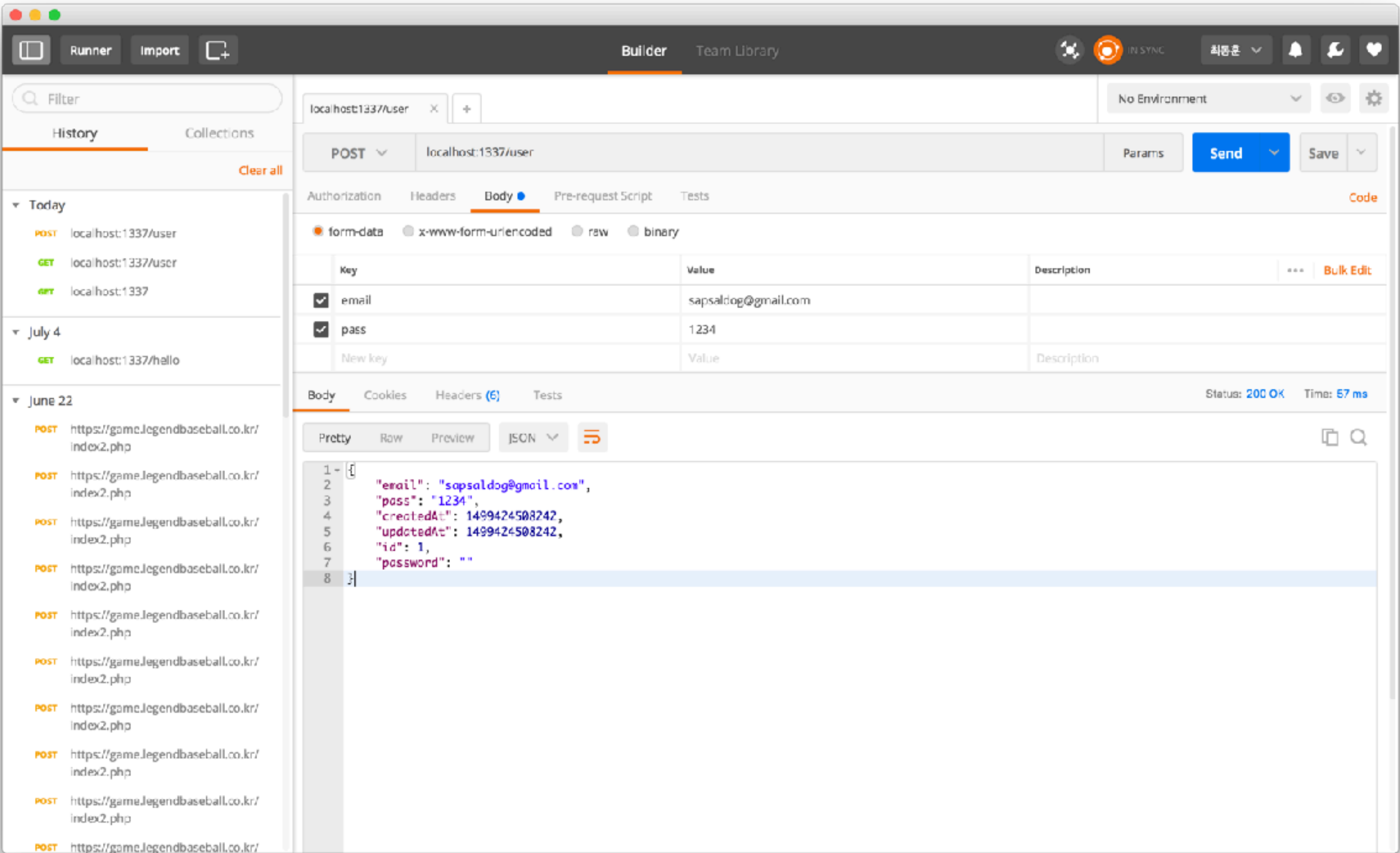
(<http://localhost:1337>)

(<http://localhost:1337/user>)

POSTMAN & Postman Interceptor 설치

(<https://chrome.google.com/webstore>)

POSTMAN을 통해서 GET, POST, PATCH, DELETE 사용



실습 - 과정 설명

blueprints를 활성화하고 sails lift를 실행 하면

프레임워크가 모델에 따른 특정 경로를 묵시적으로 바인드 한다.

묵시적(implicit) vs 명시적(explicit)

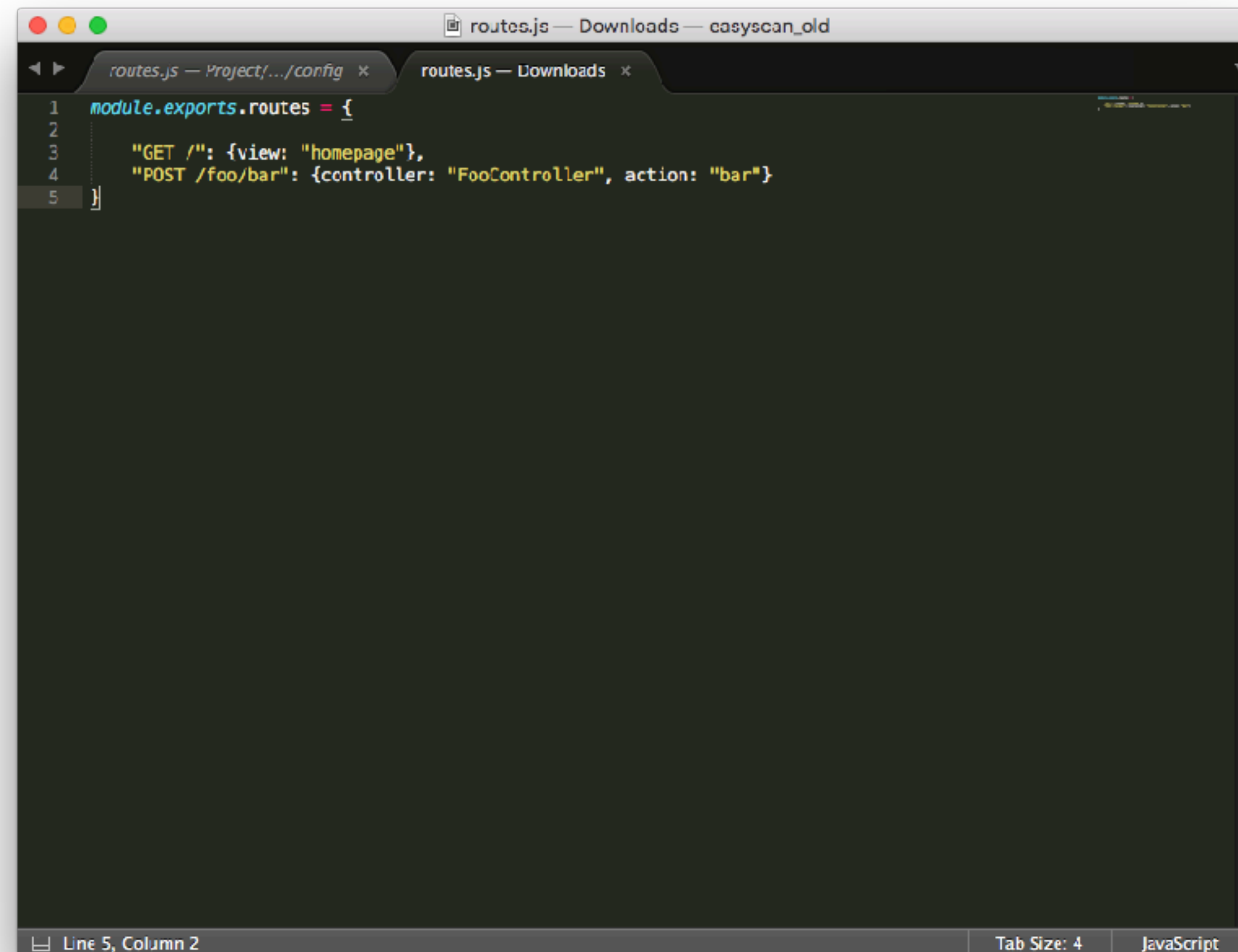
묵시적 -> 말하지 않아도 자동으로

레일즈의 철학

- DRY : “Don’t Repeat Yourself”

- 설정 보다 관습(Convention Over Configuration)

Route



```
1 module.exports.routes = {  
2  
3   "GET /": {view: "homepage"},  
4   "POST /foo/bar": {controller: "FooController", action: "bar"}  
5 }
```

request 패킷을 관찰 후
이 패킷을 대상으로 하는 “무언가”로 보내는 과정

실습

- 묵시적 라우팅 (action 사용)
- 명시적 라우팅 (routes.js 이용)

ORM

(Object Relation Mapping)

ORM이 없었던 시절

백엔드 프로그래머가 직접 쿼리를 제작하고, 결과 값을 파싱하여 변수에 바인드 함

```
String sql = "SELECT ... FROM persons WHERE id = 10";
```

```
DbCommand cmd = new DbCommand(connection, sql);
```

```
Result res = cmd.Execute();
```

```
String name = res[0]["FIRST_NAME"];
```

ORM이 생긴 이후

백엔드 프로그래머는 쿼리를 고안할 필요가 없어짐 / 결과 값 파싱 및 바인드도 신경 쓸 필요가 없어짐

```
Person p = repository.GetPerson(10);
```

```
String name = p.getFirstName();
```

자동으로 쿼리를 생성해줌 -> 일부분 효율성의 문제 발생 -> 여전히 수동으로도 사용

실습 - sails.js의 ORM 확인

- \$ sails console 명령어로 sails console을 띄워서 확인
- <http://next.sailsjs.com/documentation/reference/waterline-orm/models/create>

도커(docker)

도커(docker)

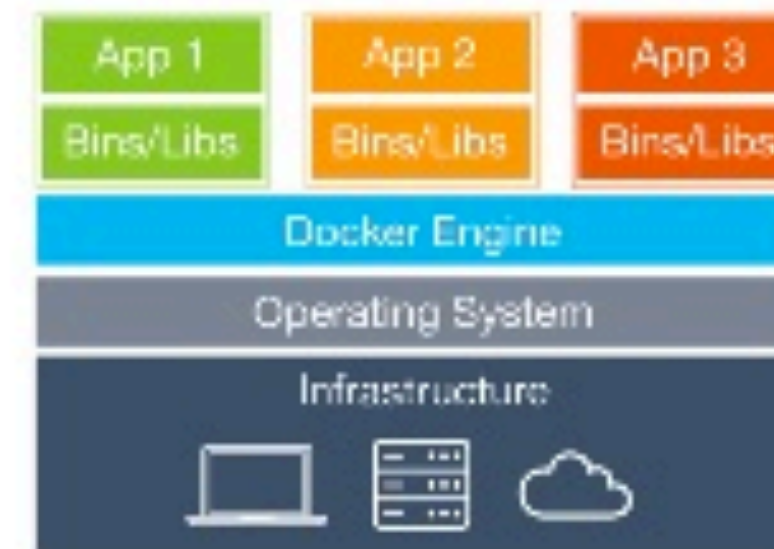
- 도커는 컨테이너를 제공하는 소프트웨어 기술이다.
- 도커는 추가적인 추상 레이어 및 운영체제 레벨의 가상화의 자동화를 제공한다.
- 가볍다 = 빠르다 (일반 가상 머신과 비교하여)
- 쉽다. 각종 이미지들을 손쉽게 구할 수 있다.
- 도커 핵심 명령어 정리 : <https://gist.github.com/nacyot/8366310>

Virtual Machines vs Docker



Virtual Machines

Each virtual machine includes the application, the necessary binaries and libraries and an entire guest operating system - all of which may be tens of GBs in size.



Containers

Containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system. They're also not tied to any specific infrastructure - Docker containers run on any computer, on any infrastructure and in any cloud.

Source: <https://www.docker.com/what-docker>

DBMS

(Database Management System)

Docker로 MariaDB 설치

MariaDB Docker로 이미지 만들기

```
$ docker run --name [이미지 이름] -d \
```

```
-v [컴퓨터에 db파일을 저장할 경로]:/data/db \
```

```
-p 3306:3306 \
```

```
-e MYSQL_ROOT_PASSWORD=1234 \
```

```
mariadb
```

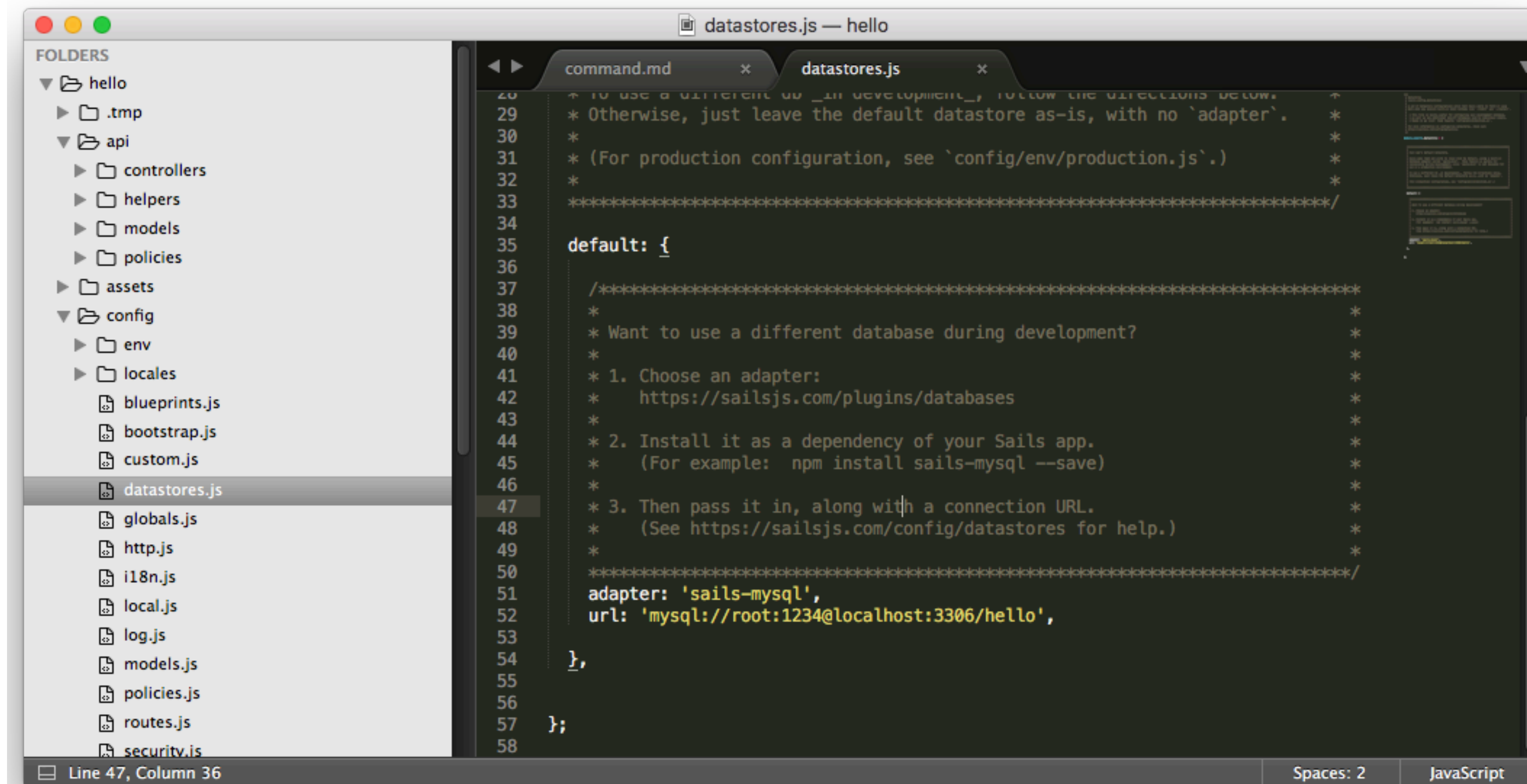
만든 이미지 실행하기

```
$ docker start [이미지 이름]
```



실습 : DBMS 활용

MariaDB와 서버 연결



config/datastores.js 파일에서 현재 띄운 DB 연결

```
1. bash
error: • Check that you're using the latest stable version of Sails.
error: • Have a question or need help? (http://sailsjs.com/support)
sapsaldogs-MacBook-Pro:hello sapsalldog$ sails lift

info: Starting app...

error: A hook (`orm`) failed to load!
error:
error: Trying to use an unrecognized adapter, `sails-mysql`, in datastore `default`.
This may or may not be a real adapter available on NPM, but in any case it looks
like `sails-mysql` is not installed in this app
(at least it is not in the expected path within the local `node_modules/` directory).

To attempt to install this adapter, run:
npm install sails-mysql --save

error: Could not load Sails app.
error:
error: Tips:
error: • First, take a look at the error message above.
error: • Check that you're using the latest stable version of Sails.
error: • Have a question or need help? (http://sailsjs.com/support)
sapsaldogs-MacBook-Pro:hello sapsalldog$
```

1


```
1. bash
error:
error: Error: `drop` called its `badConnection` exit with:
{ error:
  { Error: ER_BAD_DB_ERROR: Unknown database 'hello'
    at Handshake.Sequence._packetToError (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/protocol/sequences/Sequence.js:48:14)
    at Handshake.ErrorPacket (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/protocol/sequences/Handshake.js:101:18)
    at Protocol._parsePacket (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/protocol/Protocol.js:280:23)
    at Parser.write (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/protocol/Parser.js:73:12)
    at Protocol.write (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/protocol/Protocol.js:39:16)
    at Socket.<anonymous> (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/Connection.js:96:28)
    at emitOne (events.js:96:13)
    at Socket.emit (events.js:188:7)
    at readableAddChunk (_stream_readable.js:176:18)
    at Socket.Readable.push (_stream_readable.js:134:10)
    at TCP.onread (net.js:547:20)
    -----
    at Protocol._enqueue (/Users/sapsalldog/Project/fastcampus/hello/node_modules/mysql/lib/protocol/Protocol.js:141:48)
    at Protocol.handshake (/Users/sapsalldog/Project/fastcampus/hello/node_mod
```




DBeaver 설치

- <http://dbeaver.jkiss.org/>
- Enterprise 버전으로 다운로드
- 각종 데이터 베이스의 쿼리를 날려볼 수 있다.

실습 - DB 직접 확인

- DBEAVER로 데이터베이스 생성
- POSTMAN으로 User CRUD 해보기
- DBEAVER로 데이터베이스 갱신 확인 해보기

RDBMS

(Relational Database Management System)

Relational databases

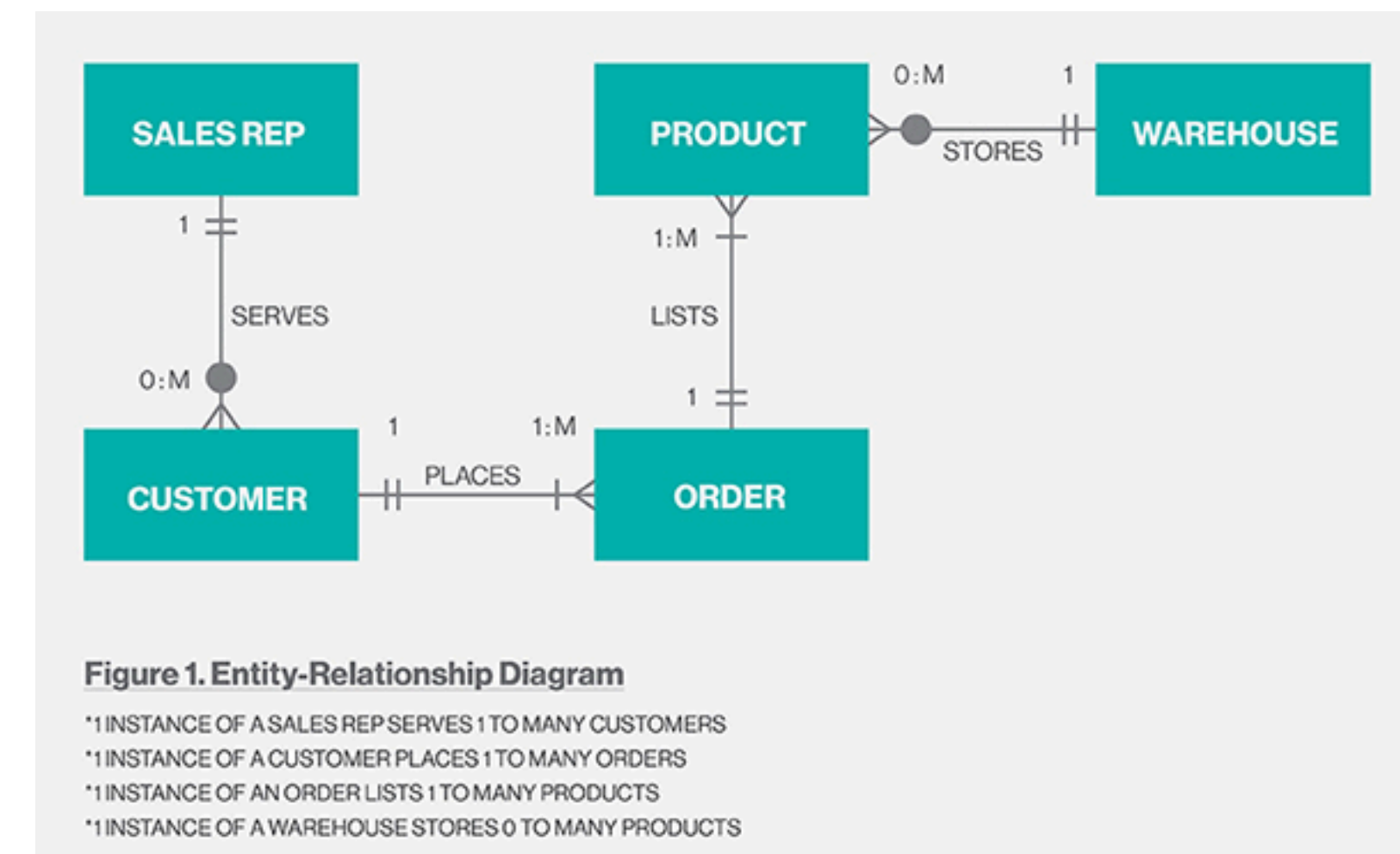
- 관계형 모델(Relational Model)

- 구조체(테이블 스키마 지칭)와 1차 술어논리로 구성된 언어(DB에서는 SQL을 지칭)를 사용하여 데이터를 다루는 접근법

Symbol	Meaning
\vee	or
\wedge	and
\neg	not
\Rightarrow	logically implies
\Leftrightarrow	logically equivalent
\forall	for all
\exists	there exists

- 관계 종류

1. 일대일
2. 일대다 / 다대일
3. 다대다



- 대표적인 제품 : Oracle, MySQL, MariaDB, Sybase, Microsoft SQL Server, Access, etc.

DBMS 기본 용어

- 스키마(schema)
- 테이블(table)
- 쿼리(query)
- 뷰(view)
- 행(row)
- 열(column)
- 기본키(primary key)
- 외래키(foreign key)
- 데이터형(data type)

SQL

(Structered Query Language)

#SQL(Structered Query Language)

- SQL구문은 몇 가지 주요 키워드로 구성돼 있어 쉽게 배울 수 있음
- 특정 RDBMS에서만 쓸수 있는 SQL 구문도 존재 하지만 대부분은 공통적으로 쓸 수 있음
- 대소문자 구분 안함

테이블 생성 - CREATE TABLE [테이블 이름] ([컬럼 이름] [자료형], ...);

테이블 삭제 - DROP TABLE [테이블 이름]

입력 - INSERT INTO [테이블 이름]([컬럼 이름], ...) VALUES([입력값], ...);

조회 - SELECT [컬럼이름], ... FROM [테이블 이름] WHERE [조건식]

수정 - UPDATE [테이블 이름] SET [컬럼 이름] = [수정할 값] WHERE [조건식]

삭제 - DELETE [테이블 이름] WHERE [조건]

실습 - 쿼리 실습

- DBEAVER를 활용하여 QUERY 문을 생성, 실행 해보자.
- 먼저 쿼리로 데이터를 생성, 결과를 sails.js에서 확인 해보자.

트랜잭션(Transaction)

- 데이터베이스 관리 시스템의 상호작용의 단위
- 논리적 작업 단위(LUW, Logical Units of Work)
- 목적 : 데이터베이스 완전성(integrity) 유지

NoSQL

RDBMS와 다르게 최초 생성시 데이터간 관계를 정의하지 않음 (설계시 유연) - 명시적인 스키마가 존재하지 않음

대표적인 NoSQL

- MongoDB, Redis, HBase 등

RDBMS vs NoSQL 장단점

RDBMS의 장점

- 범용성, 고성능, 안정적, 일관성
- 정규화를 전제로 하고 있기 때문에 업데이트시 비용이 적다
- 데이터베이스 설계시 불필요한 중복을 방지
- 복잡한 형태의 쿼리가 가능하여 원하는 데이터를 얼마든지 볼 수 있다. (Join, Group by 등)
- 이미 성숙한 기술

RDBMS의 단점

- 대량의 데이터 입력처리
- 테이블의 인덱스 생성, 스키마 변경 시 성능
- 개발, 운영할 때 데이터 컬럼을 확정 짓기 어려운 경우

NoSQL의 장점

- 데이터 분산에 용이
- 복제 및 장애대응에 용이
- 데이터를 고속으로 처리할 필요가 있는 경우
- 로그 등 계속 적재되어야 하고 대량의 데이터를 저장하고 싶은 경우 용이

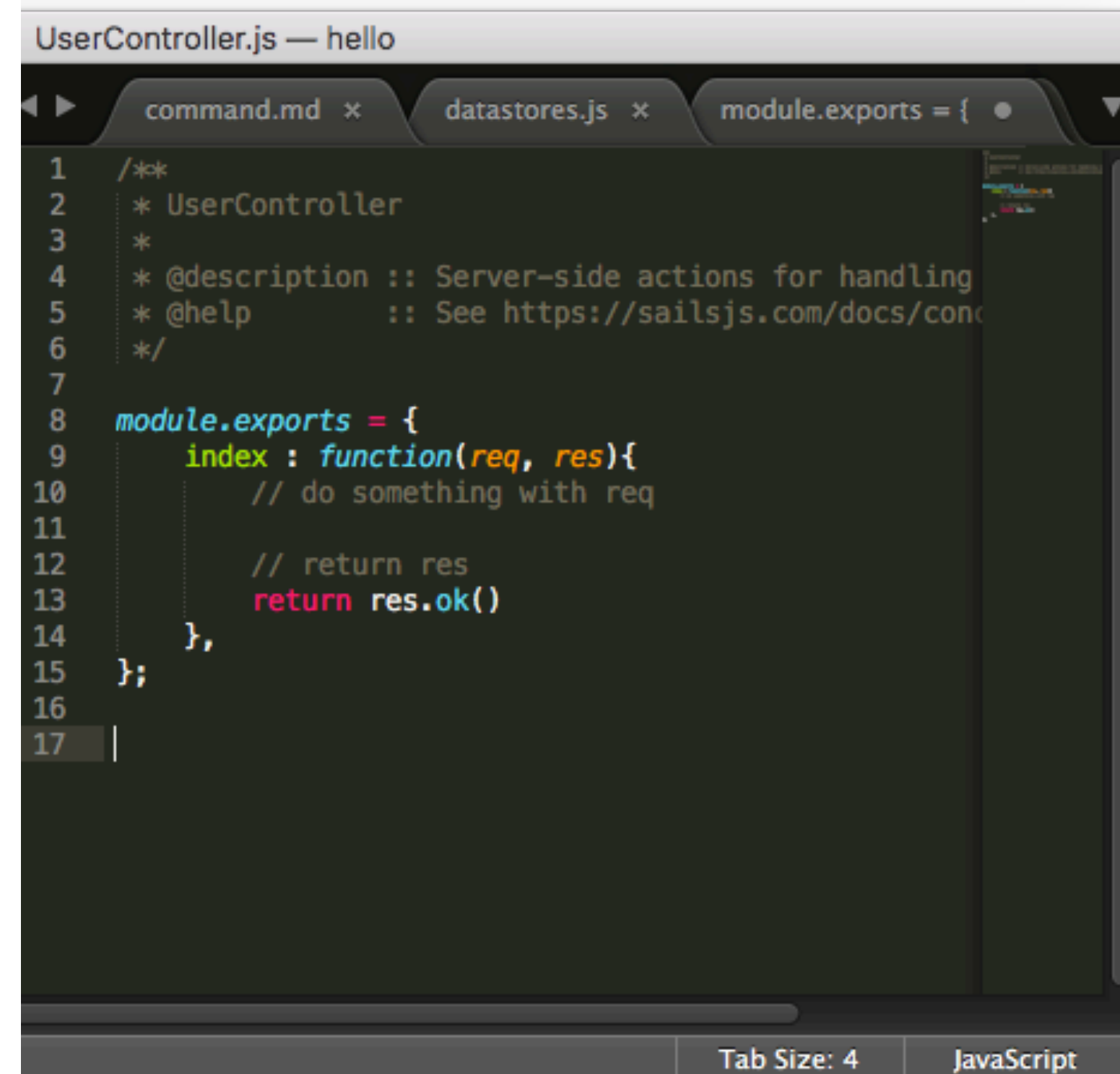
NoSQL의 단점

- 각 솔루션의 특징을 이해 해야할 필요가 있음. 러닝커브가 가파름
- 새로운 기술로 분류되기 때문에 운영에 대한 노하우가 부족 (아는 사람 적음)
- RDBMS와 비교하여 아직 안정성 부족
- 보안에 취약하기 때문에 별도의 보안 체계를 마련
- 데이터 모델링의 어려움

REQUEST, RESPONSE

sails.js 컨트롤 함수

- 익스프레스 4.0 형식과 동일
- <https://expressjs.com/en/4x/api.html#req>
- <http://next.sailsjs.com/documentation/reference/request-req>



```
UserController.js — hello
command.md × datastores.js × module.exports = {
1  /**
2   * UserController
3   *
4   * @description :: Server-side actions for handling
5   * @help       :: See https://sailsjs.com/docs/conc
6   */
7
8  module.exports = {
9    index : function(req, res){
10      // do something with req
11
12      // return res
13      return res.ok()
14    },
15  };
16
17
Tab Size: 4 JavaScript
```


req 주요 파라미터

- req.body : request의 바디
- req.header : request의 헤더
- req.cookies : request의 쿠키
- req.method : request의 메서드 (GET, POST 등)
- req.params : request의 URL 파라미터

(라우팅 설정에서 /user/:name 을 설정 한 경우

req.params.name 으로 :name을 접근 할 수 있음)

res 주 사용법

```
res.set('Content-Type', 'text/html');
```

```
res.send(new Buffer('some html'));
```

```
res.send(new Buffer('whoop'));
```

```
res.send({ some: 'json' });
```

```
res.send('some html');
```

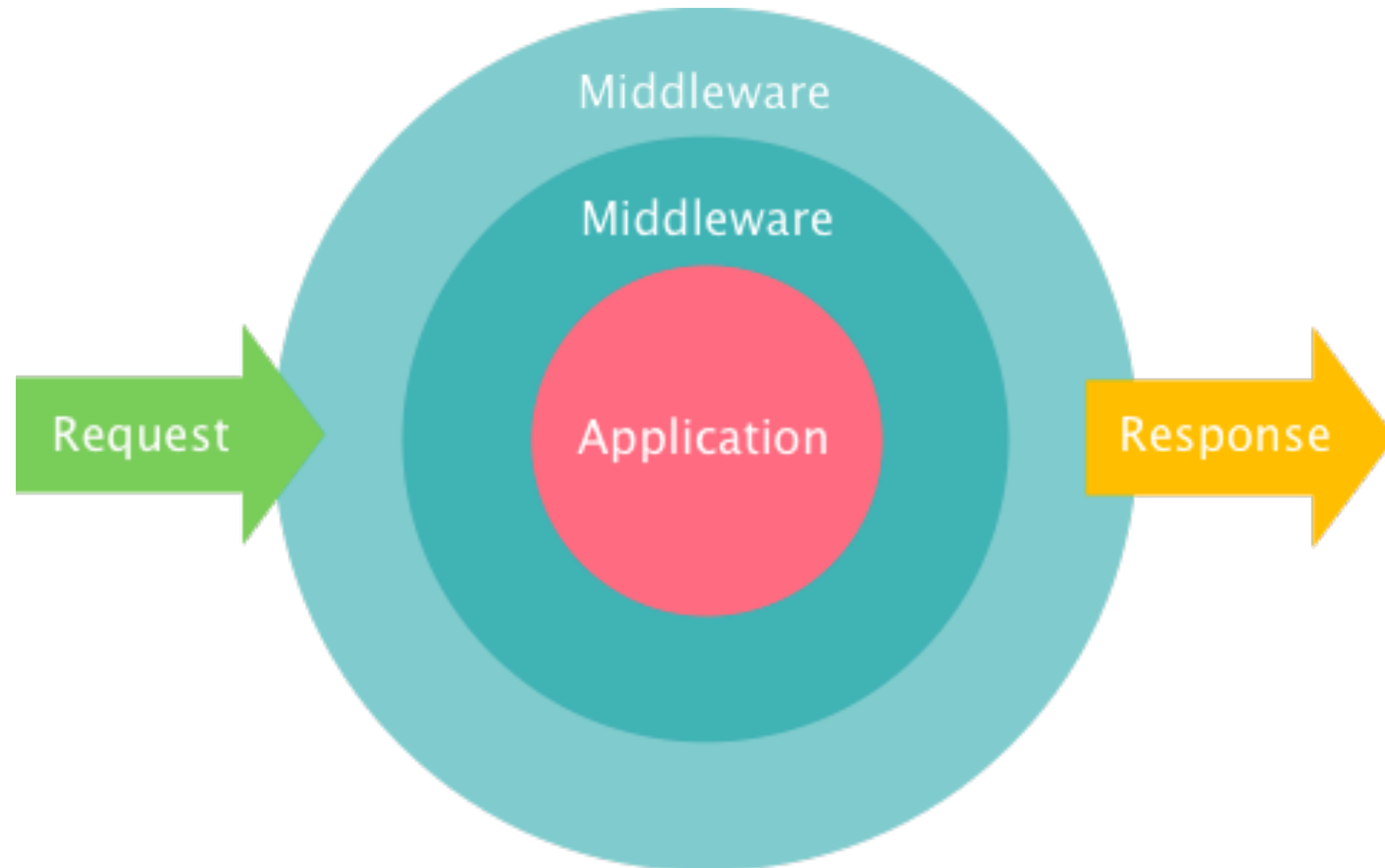
```
res.send(404, 'Sorry, we cannot find that!');
```

```
res.send(500, { error: 'something blew up' });
```

```
res.send(200);
```

```
return res.redirect('/checkout');
```

Middleware



당신의 커리어 전환점 페스트캠퍼스

IOS PROGRAMMING SCHOOL

미들웨어

- <https://expressjs.com/en/4x/api.html#app.use>
- <http://next.sailsjs.com/documentation/concepts/middleware>
- request/response stack의 ‘가운데’에 있어서 붙여진 이름
- Sails에서는 4가지 방법으로 middleware를 지원함(HTTP, Policy, Hook의 routes, Custom response)

실습 : 로그인 구현

레디스(redis)

<https://redis.io/>

더 빠른 결과를 제공하기 위해 데이터를 메모리에 저장함

(메인 메모리의 처리 속도는 HDD의 800배, SSD의 40배 정도
빠름)

인메모리 데이터 저장소 : 레디스 vs memcached

K-V 방식의 access



실습: 레디스로 세션 관리

실습 - 레디스로 세션관리 I

도커로 레디스를 설치 한다.

```
$ docker run -p 6379:6379 --name test-session-redis -d redis
```

connect-redis 를 설치한다.

```
$ npm install connect-redis --save
```

config/session.js에 다음 구문을 추가한다.

```
adapter: 'connect-redis',
```

```
url: 'redis://localhost:6379/0',
```

서버 어플리케이션을 실행한다

```
$ sails lift
```

실습 - 레디스로 세션관리 II

브라우저를 통해 어플리케이션 서버로 접속한다

주소 : <http://localhost:1337>

디버깅 목적을 위해 redis-cli 실행한다.

`docker run -it --link test-session-redis:redis --rm redis redis-cli -h redis -p 6379`

redis command : <https://redis.io/commands>

redis-cli 에서 저장된 키값들을 확인한다.

`redis-cli keys '*'`

위 키 리스트에서 임의의 값을 확인해본다.

`redis-cli keys '{key value}'`

당신의 커리어 전환점 패스트캠퍼스

IOS PROGRAMMING SCHOOL

View

EJS

- EJS(Effective JavaScript Templating) : <http://ejs.co/>
- 마지막으로 response 패킷에 넣을 HTML 데이터를 조립
- sailsjs는 ejs를 기본 view engine으로 채용

실습: 서버사이드 렌더링

GRUNT 소개

- 자바스크립트의 테스크 러너 - 자동화(Automation)
- 반복적인 일을 줄여주는데 사용
- 수많은 플러그인이 존재 (<https://gruntjs.com/plugins>)
- <https://gruntjs.com/sample-gruntfile>



GRUNT 분석

TASK

- 작업을 정의함
- 대표 TASK: minification, compilation, unit testing, linting

TARGET

- TASK는 여러개의 TARGET을 소유 가능
- 각 target은 다른 option 적용 가능
- 대표 target : development

GRUNT의 특징 (feature)

- 작업 옵션 설정
- 파일 유틸리티
- 템플릿 기능
- 외부 데이터 가져오기

실습 : less 자동화

THANK YOU :-)

당신의 커리어 전환점 패스트캠퍼스

IOS PROGRAMMING SCHOOL