



Omni-channel
Personalization
Platform
www.dable.io

시간당 수백만 요청을 처리하는 node.js 서버 운영기

김군우 (CPO, co-founder at Dable)

2015-11-12



김군우 (miya.pe.kr)



lemonpen



Game SNS(for AION)



Front-end 유틸들
(N-MET, N-WAX, OpenWAX)

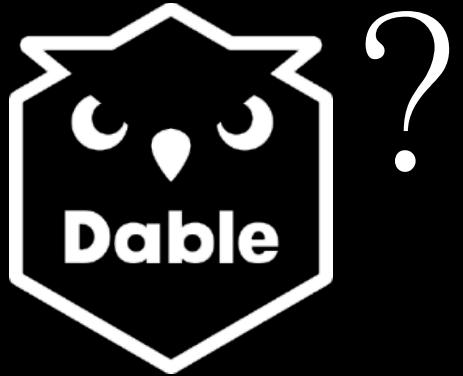


댓글 플러그인 Affogato



RecoPick

1. 수백만 요청?
2. 어떻게 운영?
3. 어떻게 개발?
4. 삽질 좀 했지?



1. 데이블이 뭐하는 회산데
수백만 요청씩이나?

겨우 6개월 밖에 안된 회사지만...



거대한 사이트들의 로그를 수집하고 있는 B2B 기업



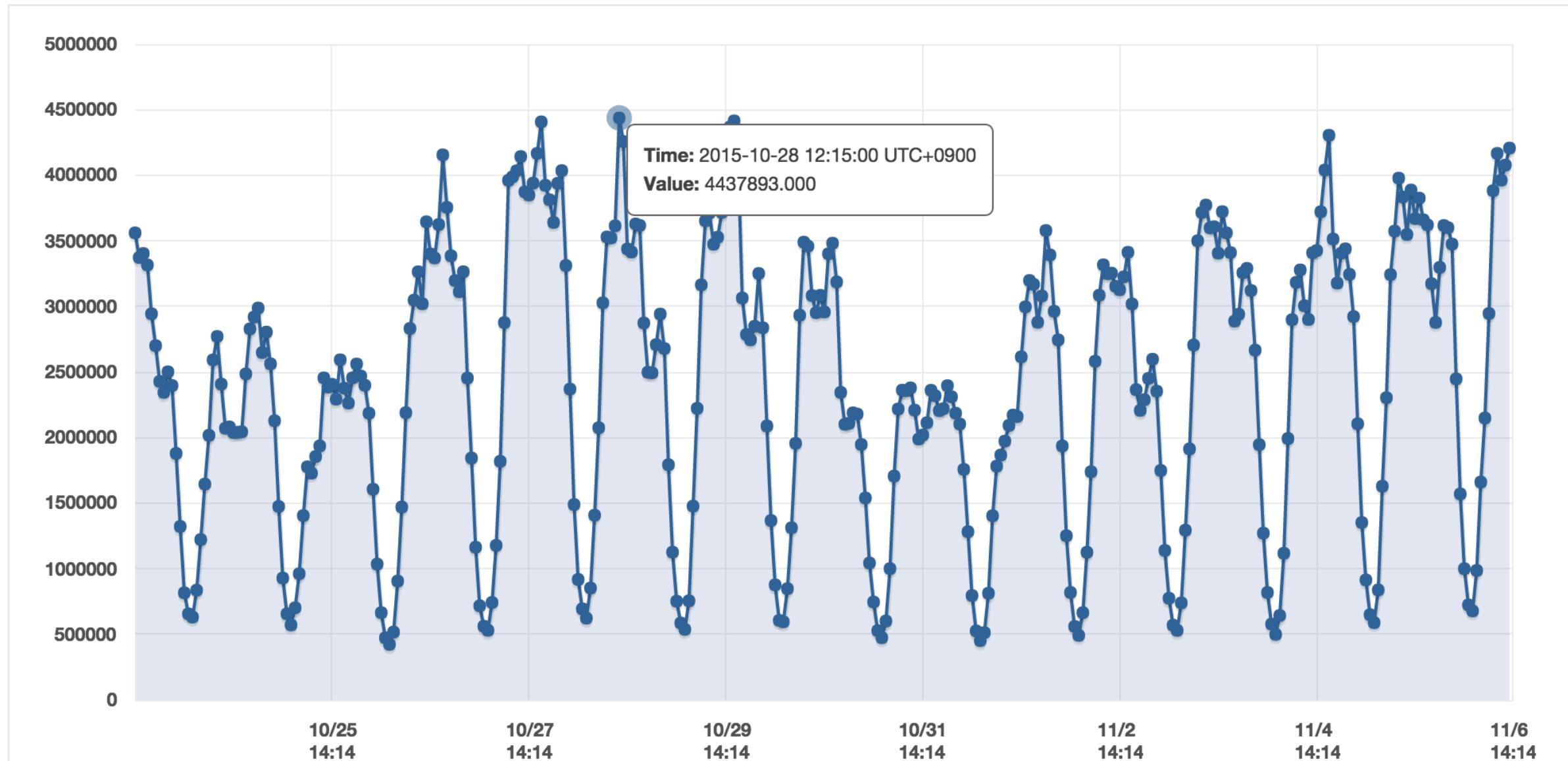
그래서 서비스 시작할 때부터 요청 수가 많았어요.

Sum ▾

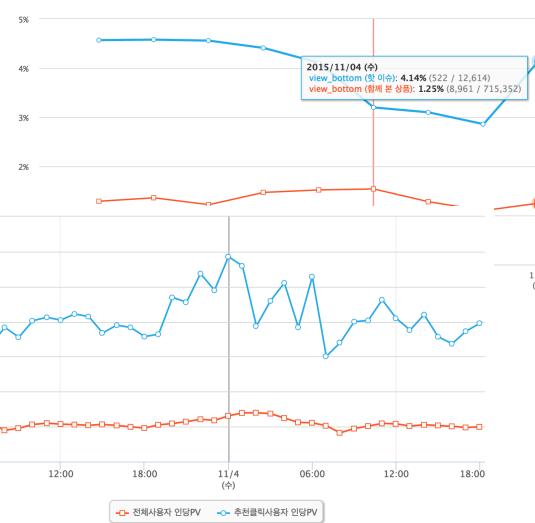
RequestCount by count

Period

1 hour ▾

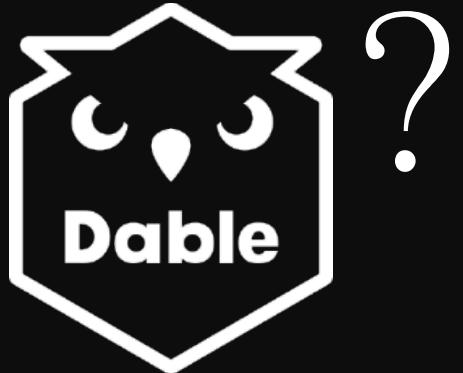


이런 형태로 서비스하고 있습니다.



이런 형태로 서비스 하고 있습니다.

- Node.js Packages in Dable
 - Web Servers
 - API: 로그 수집, 추천 제공(위젯 렌더링)
 - Recommendation(s): 추천 제공(알고리즘 별)
 - Dashboard: 통계 시각화 (고객들에게 성과 제공)
 - HQ: 내부용 관리자
- scripts
 - admin scripts (각종 관리용 스크립트 모음)
 - meta crawler
 - (API, Recommendation 등에서 모듈 형태로 포함)



2. 그래서 운영은 어떻게?

기술전...





Good

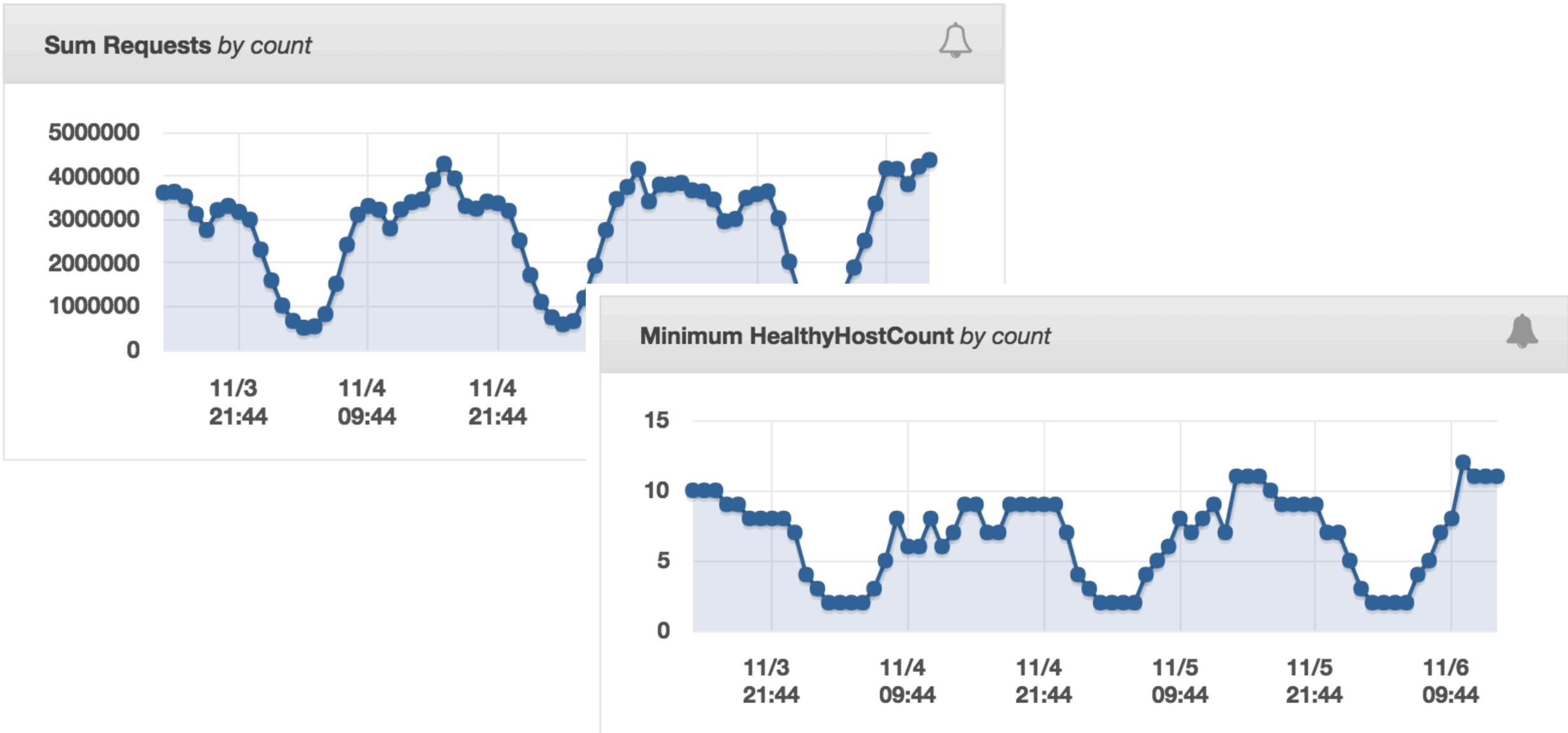
- Auto Scailing, Load Balancing 등 알아서 처리
- 서버 모니터링도 쉽게 가능하도록 지원된다.

Bad

- 플랫폼 대응이 조금 느리긴 하다.
- 1주일 전에 node v4.2.1 버전업
- 그 전까진 v0.12.6을 쓰다가;;
- 서버 환경 자유도가 좀 떨어진다.

Google App Engine, Heroku 같은 Cloud Paas Platform

Instance 수는 요청 수에 비례하여 새벽에 2대, 피크 때 10대 남짓



CPU 사용률에 따라 Instance를 넣고 빼고 합니다.

Trigger measurement:

CPUUtilization



The measure name assi

Upper threshold:

50

The upper limit for the metric. If the data points exceed th

Upper breach scale

1

increment:

The incremental amount to use when performing scaling a
optionally followed by a % sign.

Lower threshold:

30

The lower limit for the metric. If the data points are below
activated.

Lower breach scale

-1

increment:

The incremental amount to use when performing scaling a
optionally followed by a % sign.

v0.12.6 → v4.2.1

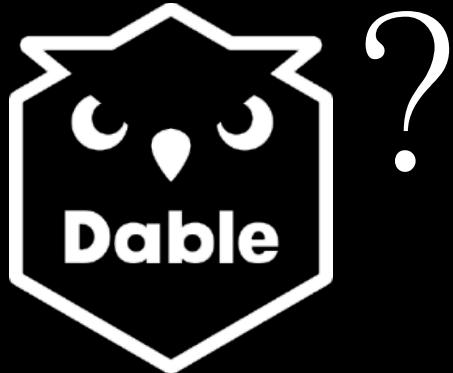
- 메모리 사용량이 반으로 줄었다!
(3.75G 중 30% -> 15%)
- CPU 사용량, 응답 속도는 체감상 크게 달라지지는 않았다.
- ~~근자감이 생겼다.~~

그 밖에 대량 트래픽 관리를 위해...

- 캐시로서의 redis 서버 활용
 - 시간당 최대 백만건의 로그를 저장해야 하는 API 서버
 - 현재 MySQL(legacy), AWS kinesis에 동시에 저장
 - 매번 INSERT 하면 부담
 - 그래서 각 서버 Instance에 local redis 서버를 띄워놓고
 - local redis로 로그를 Queue처럼 쌓은 후
 - 20건 정도씩 Dequeue 해다가 두 군데에 몰아넣는 구조

그 밖에 대량 트래픽 관리를 위해...

- **로그 저장 HTTP 요청은 최대한 빠르게 응답**
 - 로그 저장 요청은 특성 상 응답 시 성공/실패 여부 필요없음
 - 일단 HTTP 요청은 먼저 응답하고 (200 ok)
 - 저장에 필요한 일련의 동작들은 async하게 처리
 - HTTP 요청의 동작과 무관한 응답을 하는 경우라면
이 방법으로 응답 시간을 줄일 수 있을 것 같다.



3. 그래서 개발은 어떻게?

node.js 웹 서버 디렉토리 구조: express(v4.13.x) 기반

app.js

route.js

controller/

lib/

front/

test/

views/

public/

app.js: 서버 실행 파일. cluster, web server 실행

app.js

route.js

controller/

lib/

front/

test/

views/

public/

Cluster count == CPU count

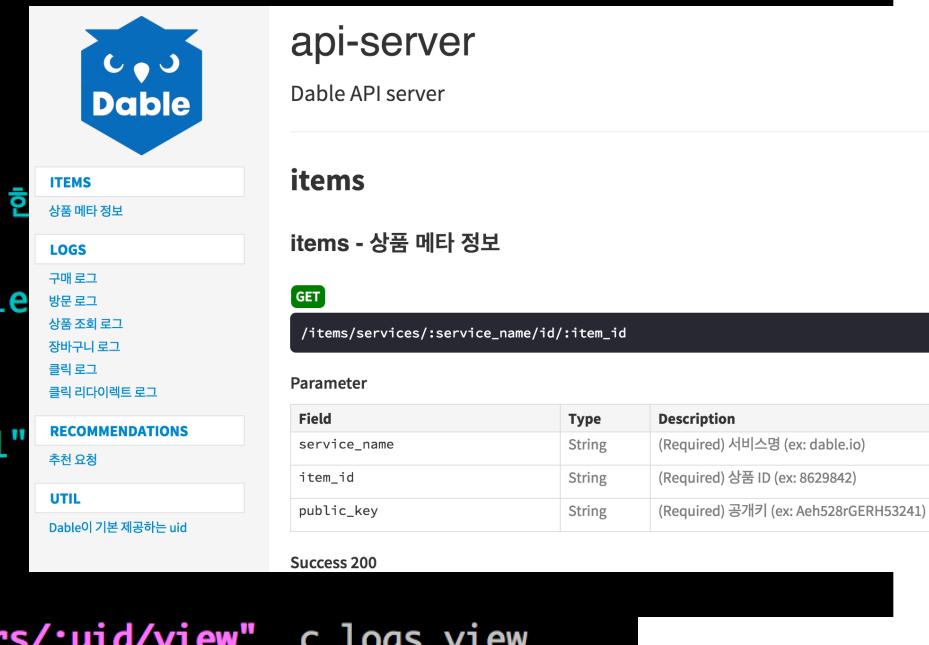
route.js - 라우팅 맵, Documentation (with apidoc)

app.js
route.js
controller/
lib/
front/
test/
views/
public/

```
route = (app) =>
    app.get "/", (req, res) => res.send "DABLE API SERVER WORKS"

    # log APIs
    ###
    @api {get, post} /logs/services/:service_name/users/:uid/visit 방문 로그
    @apiName visit_log
    @apiGroup logs
    @apiUse common_log_api
    ###
    app.get "/logs/services/:service_name/users/:uid/visit", c.logs.visit
    ###
    @api {get} /logs/services/:service_name/users/:uid/view 상품 조회 로그
    @apiName view_log
    @apiGroup logs
    @apiUse common_log_api

    @apiParam {Object[]} items (Required) 조회 한 상품 목록
    @...
    @apiParamExample {json} items param example
        {
            "items": [
                {"id": "ITEM_ID", "c1": "CATEGORY1"}
            ]
        }
    ###
    app.get "/logs/services/:service_name/users/:uid/view", c.logs.view
```



The screenshot shows the Dable API documentation interface. On the left, there's a sidebar with navigation links: ITEMS (상품 메타 정보), LOGS (구매로그, 방문로그, 상품조회로그, 장바구니로그, 클릭로그, 클릭리다이렉트로그), RECOMMENDATIONS (추천요청), UTIL (Dable이 기본 제공하는 uid). The main content area has a title 'api-server' and a subtitle 'Dable API server'. Below that, it says 'items' and 'items - 상품 메타 정보'. It shows a 'GET' method with the URL '/items/services/:service_name/id/:item_id'. A 'Parameter' table is shown with three rows: 'service_name' (String, Required), 'item_id' (String, Required), and 'public_key' (String, Required). At the bottom, it says 'Success 200'.

express middleware는 controller로, 복잡한 로직, 공통 로직은 lib으로

app.js

route.js

controller/

lib/

front/

test/

views/

public/

Front end 코드들. public/dist/로 concat & uglify

app.js

route.js

controller/

lib/

front/

style/

script/

test/

views/

public/

- CSS: stylus, JS: coffeescript
- 모듈별로 쪼갠 여러개의 파일로 개발
- JS의 경우 Browserify (CommonJS Pattern) 사용하여 서버 로직을 함께 쓰기도...
- 배포 시 concat & uglify 하여 하나의 파일로
- 기타
 - 개발시에는 grunt watch 이용
 - source map을 활용하여 디버깅

Unit tests. mocha+should+rewire(for Server), qunit/karma(for Front-end)

app.js
route.js
controller/
lib/
front/
test/
test-front/
views/
public/

Jenkins

Jenkins test-api-server #448

프로젝트로 돌아가기

상태

바뀐점

Console Output

View as plain text

빌드 정보 수정

이 빌드를 삭제

Git Build Data

No Tags

이전 빌드

다음 빌드

콘솔 출력

widget lib

✓ select abtest group by abtest active value

✓ fetch_widget_recommendations method

85 passing (3s)

```
[ 4mRunning "qunit:all" (qunit) task[24m
Testing test-qunit/collector.test.html ....[ 32mOK[ 39m
Testing test-qunit/debug.test.html ..[ 32mOK[ 39m
Testing test-qunit/invalid_log.test.html .[ 32mOK[ 39m
Testing test-qunit/view.dable=4.is_hidden.test.html .[ 32mOK[ 39m
Testing test-qunit/view.meta.block.test.html .[ 32mOK[ 39m
Testing test-qunit/view.meta.category.test.html .[ 32mOK[ 39m
Testing test-qunit/view.meta.test.html .[ 32mOK[ 39m
Testing test-qunit/view.meta_invalid_enc.test.html .[ 32mOK[ 39m
Testing test-qunit/widget.item_id.test.html .[ 32mOK[ 39m
[ 32m>> [ 39m32 assertions passed (5224ms)
```

Unit tests. rewire 사용 예제

```
app.js
var fsMock = {
  readFile: function (path, encoding, cb) {
    expect(path).to.equal("/somewhere/on/the/disk");
    cb(null, "Success!");
  }
};

myModule.__set__("fs", fsMock);

myModule.readSomethingFromFileSystem(function (err, data) {
  console.log(data); // = Success!
});
```

기타 즐겨쓰는 라이브러리

- moment: 날짜, 시간 다루기
- async -> Q: Promise!
- lodash(underscore): Super Utility
- sanitizer: 문자열에서 안전하지 않은 HTML 태그들을 걸러주는 라이브러리.

기타 즐겨쓰는 라이브러리

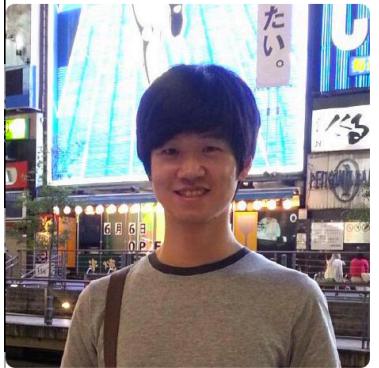
- lz-string: Front/Server 양쪽에서 사용 가능한 문자열 압축 라이브러리. Front에서 만든 URL의 긴 파라미터를 줄이기 위해 사용. (2000자 -> 800자)

```
// compress (보내는 쪽)
reco_list = [{id:"121", method:'1.1.1'}, {id:"A42", method:'1.2.1'}, ... ];
reco_list_lz = lz.compressToEncodedURIComponent(reco_list);

url = "http://api.dable.io/logs/blahblah?reco_list_lz=NobwRAlgJmBcYCYAsBWAHAZgSsAaMAtgKYAuAF
gPYzwCMADAHQ1NgC%2Bu40ciSAbCnRoB2PIVKVqYekxbt0k5Cl4BOJE1HFyVbt0Y02HSAr5ohQuhvHbajPQfndkvGs-3
5NEnbdmGu8ZEIY6MqWwpK6Pg7%2BSJhICG5iYV4y%2BnJGjjE0CAhooZ42Kfbp0YFoKDjuVuHeqb7GKOa8vHnWUjVFFj
xoNMqBLdWFaZ0BKJn9yXZDxvzlFYn5bYN1GbwyvH2VSQWTALpAA";

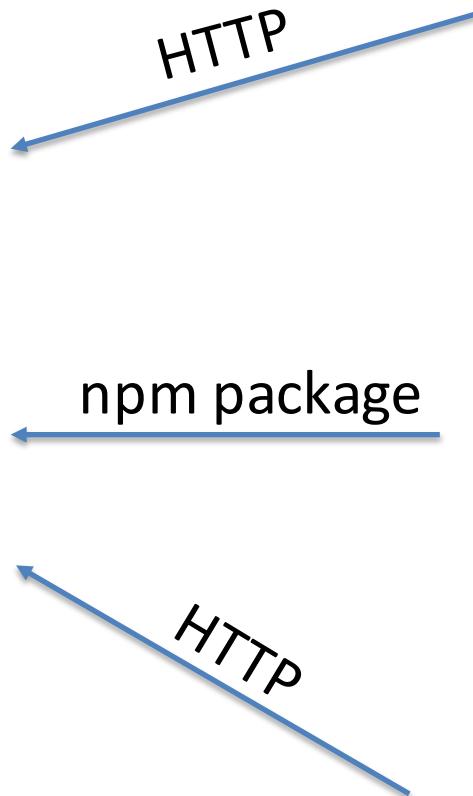
// decompress (받는 쪽)
reco_list = JSON.parse(lz.decompressFromEncodedURIComponent(reco_list_lz));
```

협업?



Goonoo Kim
mctenshi

Module A (Node.js)



Module B (Node.js)



Module C (Node.js)



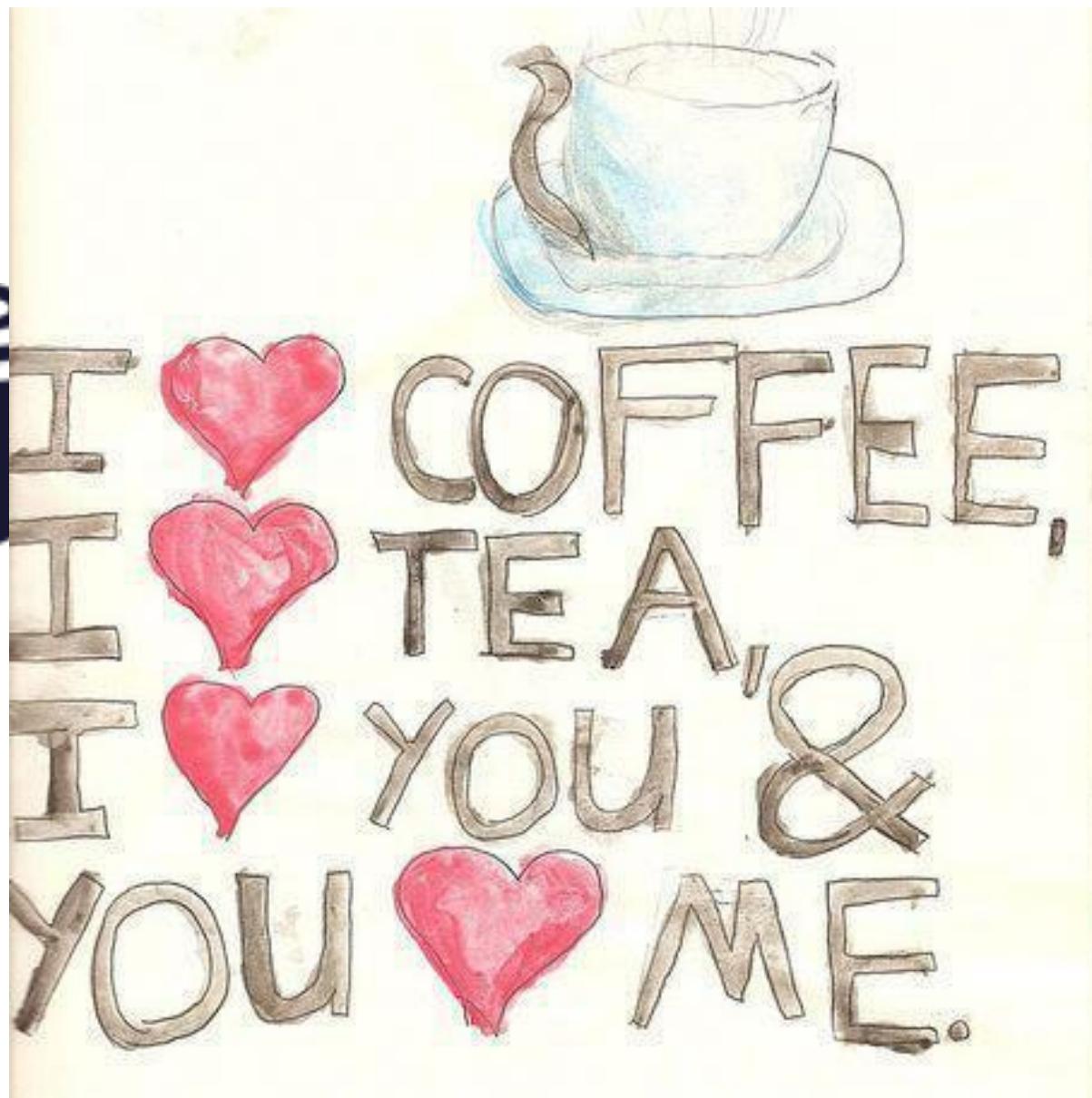
Module D (Python)

- 한 repo를 같이 작성하는 경우는 드물다
- 각자 project를 생성하여
 - npm package 형태로 가져다 쓰거나
 - HTTP로 통신한다.
 - HTTP 통신
 - 성능은 별로지만
 - B 수정 시 A 재배포가 필요없어 관리에 좋다.

CoffeeScript를 즐겨쓰는데요...



왜냐하면 커피를 좋아해서...



ipt



CoffeeScript

- 들여쓰기, 띄어쓰기, brace 어디다 열지, jslint/jshint... 피곤
- 눕으니 타이핑 속도가 느려짐. IDE 취향 아님 -_-;;
 - 물론 이건 es6에 익숙해지는 해결책도...
 - 근데 front end JavaScript 코딩도 같은 언어로 하다보니 es6 쓰면 context swiching이 잘 안됨 ㅠ

```
var that = this;  
function callback(){  
    that.something();  
}
```

```
# in coffeescript  
callback => @something()
```



CoffeeScript

“ less typing, bad readability
Why CoffeeScript Isn't the Answer
Replace CoffeeScript with ES6

네... CoffeeScript를 추천할 심산은 아니에요.
ES6, Babeljs. 조금 더 나은 해답들이 많죠.



CoffeeScript

그런데 IE7, IE8를
버릴 수 없는
상황이라면? (...)

거기다 저같은 불
편함을 느끼신다
면 추천!

 Goonoo Kim
10월 12일 오후 5:57 · 서울 · 수정됨 · ▶

Dable 서비스의 대충 최근 1만건 로그로 브라우저 점유율을 계산해보았는데 IE7이 얼마나 될까요? 참고로 Dable의 고객은 대략 이런 사이트들: <https://dable.io/Partners>
정답은 댓글에

 좋아요  댓글 달기  공유하기



4. 몇 가지 삽질기

삽질 1: 자기 서버에 HTTP로 요청을 날리는 로직을 만들었다가...

일단 설명 전에... 이것들, 익숙하시죠?

502 Bad Gateway

nginx/1.8.0

The screenshot shows a GitHub repository page for the project 'request / request'. At the top, there's a navigation bar with a GitHub icon, 'This repository', a search bar, and links for 'Pull requests' and 'Issues'. Below the header, the repository name 'request / request' is displayed with a blue icon. A description follows: 'Simplified HTTP request client.' Below this, there are statistics: '1,889 commits', '16 branches', and '98 releases', each accompanied by a small icon. A green button labeled 'Branch: master' with a dropdown arrow is present. At the bottom, a recent commit by 'simov' is shown: 'Merge pull request #1893 from request/greenkeeper-eslint-1.9.0 ...'.

삽질 1: 자기 서버에 HTTP로 요청을 날리는 로직을 만들었다가...

그럼 request를 날린 서버가 502 Bad Gateway라면?

502 Bad Gateway

nginx/1.8.0

```
request.get("http://some.where/in/the/world",
  function (err, response, body) {
  // err가 있을까 없을까?
});
```

없더라고요.....

삽질 1: 자기 서버에 HTTP로 요청을 날리는 로직을 만들었다가...

```
var fetch_widget_data = function (options) {
  request({
    url: 'http://api.dable.io/recommendations/playnode.io/',
    form: options
  }, function (err, resp, body) {
    if (err) return res.json({result: []});
    // BLAHBLAH
    return res.json(
      JSON.parse(body)
    );
  });
};
```

1. 무언가 다른 원인으로 서버 뺌음
2. 서버 리스트트 (리스타트 도중 일부 nginx의 502 Bad Gateway 에러)
3. 여기서 날린 Request가 그 에러에 해당
4. err는 null이고 body는 502 Bad Gateway에 해당하는 HTML 코드라 JSON.parse 시도 중 서버 뺌음
5. 무한루프

삽질 1: 자기 서버에 HTTP로 요청을 날리는 로직을 만들었다가...

```
var fetch_widget_data = function (options) {
  request({
    url: 'http://api.dable.io/recommendations/playnode.io/...',
    form: options
  }, function (err, resp, body) {
    if (err) return res.json({result: []});
    // BLAHBLAH
    try { return res.json(JSON.parse(body)); }
    catch (e) { return res.json({result: []}); }
  });
};
```

JSON.parse 로직을 try, catch 처리.

후에 HTTP 요청을 function call 형태로 변경. (진작 이랬어야...)

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

2015-08-15 16:20:52 UTC+0900	INFO	Adding instance 'i-be95cc4b' to your environment.
2015-08-15 16:20:49 UTC+0900	INFO	Added EC2 instance 'i-be95cc4b' to Auto Scaling Group 'awseb-e-efbczggwxa-stack-AWSEBAutoScalingGroup-GTE9RB9AZ49Q'.
2015-08-15 16:18:11 UTC+0900	INFO	Removed instance 'i-71396784' from your environment. (Reason: Instance is in 'shutting-down' state)
2015-08-15 16:09:58 UTC+0900	INFO	restartAppServer is starting.
2015-08-15 10:34:11 UTC+0900	WARN	Environment health has transitioned from YELLOW to RED
2015-08-15 10:32:11 UTC+0900	WARN	Environment health has transitioned from GREEN to YELLOW
2015-08-15 10:32:11 UTC+0900	WARN	Elastic Load Balancer awseb-e-e-AWSEBLba-1NXX5H3MA73NK has zero healthy instances.
2015-08-14 15:29:25 UTC+0900	INFO	Removed instance 'i-a64f1053' from your environment. (Reason: Instance is in 'terminated' state)
2015-08-14 15:21:55 UTC+0900	INFO	Adding instance 'i-71396784' to your environment.

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

원인은 Memory Leak

서버 구동 후 Memory는 야금야금 상승

휴일의 한가운데에서 뺏어버렸다.

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

메모리가 왜 올라가는지 몰라서...

서버 로그도 보고

코드도 거듭 정독하고

디버그 코드도 잔뜩 심어보고

그래도 몰라서...



삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

Heap Snapshot을 찍어봤다. (with <https://github.com/bnoordhuis/node-heapdump>)

크롬 개발자도구의 Profiles 탭에서 두 snapshot을 비교해봤다.

이제 어쩌지 싶었다.

The screenshot shows the Chrome DevTools interface with the 'Profiles' tab selected. On the left, there's a sidebar with 'Elements', 'Network', 'Sources', 'Timeline', 'Profiles' (which is active), 'Resources', 'Audits', and 'Console'. Below that is a section for 'HEAP SNAPSHOTS' containing two entries: 'heapdump_0726_01' (40.8 MB) and 'heapdump_0726_02' (40.5 MB). The main area is titled 'Comparison' and shows a table for 'Constructor' objects. The table has columns for '# New', '# Del...', '# D...', 'Alloc.', 'Freed...', and 'Size ...'. The data shows significant differences between the two snapshots, particularly in the count of new objects and their sizes. A 'Class filter' dropdown is also visible above the table.

Constructor	# New	# Del...	# D...	Alloc.	Freed...	Size ...
▶(array)	49 322	48 682	+640	9 42...	8 95...	+472...
▶(string)	36 963	37 012	-49	8 58...	8 26...	+319...
▶(compiled code)	16 620	16 551	+69	5 19...	4 68...	+501...
▶(closure)	13 975	13 539	+436	1 00...	974...	+31 ...
▶(system)	15 207	15 068	+139	704...	740...	-35 ...
▶Object	15 831	15 772	+59	563...	560...	+2 712
▶Array	11 312	11 200	+112	361...	358...	+3 584
▶smalloc	35	32	+3	331...	291...	+39 ...
▶system / Context	3 513	3 344	+169	245...	233...	+12 ...
▶(concatenated string)	4 937	4 885	+52	197...	195...	+2 080
▶(reexecn)	858	856	+2	61 776	61 632	+144

Retainers

Object	Distance	Shallow Size	Retained Size

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

비교한 내용을 하나하나 읽기 시작한다.

읽다보니 패턴이 좀 보인다.

유... 유레카--;

The screenshot shows the Chrome DevTools interface with the 'Memory' tab selected. On the left, there's a sidebar with 'Profiles' and 'HEAP SNAPSHTOS' sections. Under 'HEAP SNAPSHTOS', two snapshots are listed: 'heapdump-10197972.727563' (111 MB) and 'heapdump-10485104.77953' (128 MB). The latter is highlighted with a blue selection bar. The main area is titled 'Comparison' and shows a tree view of memory differences between the two snapshots. At the top of the tree, it says '(array)' and then lists several objects under '(object elements)[] @2639297'. These objects include various arrays and objects like 'code deopt data', 'object properties', and 'transition array'. Below this, there's a section titled 'Retainers' which shows the chain of references for an object in an array. It starts with 'elements in Array @123381', which contains 'news1.kr:80:: in @123177', 'requests in Agent @57997', 'globalAgent in @113789', 'exports in NativeModule @44865', '_http_agent in @14347', '_cache in function NativeModule() @14343', 'NativeModule in system / Context @13523', 'context in function () @2797', 'clearInterval in @543', '2 in [] @44665', 'context in function () @2787', and 'context in function () @2751'. To the right of the tree view, there's a table with columns for 'Alloc. Size', 'Freed Size', 'Distance', 'Shallow Size', and 'Retain'. The first few rows of the table are visible, showing values like 19 216, 11 512, 9 264, etc.

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

아래가 문제의 코드. 뭐가 문제였을까요?

```
var crawl = function (url, callback) {
  request({
    url: url,
    headers: { 'User-Agent': 'Mozilla/5.0 (Macintosh; ...'}
  }, function (err, res, body) {
    if (err) return callback(err);

    var result = body;
    // !@!%#$AT#T@#%
    callback(null, result);
  });
};

// examples (callback은 생략)
crawl("http://news.kbs.co.kr/news/view.do?ncd=3107610");
crawl("http://news1.kr/articles/?2377986");
```

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

```
var crawl = function (url, callback) {
  request({
    url: url,
    headers: { 'User-Agent': 'Mozilla/5.0 (Macintosh; ...'}
  }, function (err, res, body) {
    if (err) return callback(err);

    var result = body;
    // !@!%#$AT#T@#%
    callback(null, result);
  });
};

// examples (callback은 생략)
crawl("http://news.kbs.co.kr/news/view.do?ncd=3107610");
crawl("http://news1.kr/articles/?2377986");
```

원인은 timeout 없는 HTTP request
고객사 중 한곳이 API Server IP를 Block
Block 방식이 응답을 주지 않는 식
응답이 없으니 linux 기본 timeout(2시간) 적용
요청만큼의 request가 메모리에 쌓여갔던 것

삽질 2: 평화로운 휴일. 서버가 뺐었다. 뺏은 서버는 살아나지 않았다.

```
var crawl = function (url, callback) {
  request({
    url: url,
    timeout: 5000,
    headers: {'User-Agent': 'Mozilla/5.0 (Macintosh; ...'}
  }, function (err, res, body) {
    if (err) return callback(err);
    var result = body;
    // !@!%#$AT#T@#%
    callback(null, result);
  });
};

// examples (callback은 생략)
crawl("http://news.kbs.co.kr/news/view.do?ncd=3107610");
crawl("http://news1.kr/articles/?2377986");
```

timeout을 추가하여 문제 해결

마치며... SPECIAL THANKS TO NODE.JS

저에게 node.js란...

서버 개발 지식이 부족한 저같은 프론트 엔드 개발자도
큰 규모의 서비스를 운영할 수 있게 해준 존재입니다.

같이 삽질하실 분... 찾습니다. +_+

WANTED: node.js 개발자 (빅데이터 흥미있으신 분 대환영!)

dable@dable.io

A photograph of a group of seven people (four men and three women) standing in front of a wooden wall. On the wall is a large red geometric logo resembling a four-pointed star or a diamond shape, and the words "CAMPUS SEOUL" in white capital letters. To the left of the photo is a blue hexagonal logo for "Dable" featuring a white owl icon. Below the photo, the word "Dable" is written in a bold, sans-serif font, followed by the text "인터넷/소프트웨어". At the bottom right are several social media-style buttons with Korean text: "문의하기", "좋아요", "메시지", and three dots.

Dable
인터넷/소프트웨어

문의하기 ▾

좋아요 ▾

메시지

...



THANK YOU