

개발 프로세스 실습 2

Hyunchan, Park

<http://oslab.jbnu.ac.kr>

Division of Computer Science and Engineering

Jeonbuk National University



참고자료

- <http://www.nextree.co.kr/p8574/>
 - Node.js
- <https://www.lesstif.com/pages/viewpage.action?pageId=14745703>
 - Curl
- <https://www.leafcats.com/215>
 - Jenkins by Docker (plugin 설치 실패)
- <https://kkensu.tistory.com/58>
 - Jenkins 수동 설치 ubuntu
- <https://yaboong.github.io/jenkins/2018/05/14/github-webhook-jenkins/>
 - GitHub-Jenkins 연동
- <https://subicura.com/2016/07/11/coding-convention.html>
 - Linter
- <https://proinlab.com/archives/1885>
 - 슬랙 챗봇
- <https://heropy.blog/2018/03/16/mocha/>
 - 모카 테스트

참고자료

- <https://github.com/slackapi/node-slack-sdk>
 - Node-slack-sdk
- <https://bcho.tistory.com/759>
 - 조대협 개발자님 블로그: 개발 프로세스
- <https://bcho.tistory.com/1237>
 - Jenkins-GitHub 연동
- <https://kutar37.tistory.com/entry/Jenkins-Github-%EC%97%B0%EB%8F%99-%EC%9E%90%EB%8F%99%EB%B0%B0%ED%8F%AC-3>
 - Jenkins-GitHub 연동

실습에서 진행해볼 개발 프로세스

1. Development
2. Code Convention Check & Unit test
3. Commit
4. Pull Request
5. Code review
6. Merge
7. Integration and Build (on the Build environment, deployed by git clone)
8. Deploy the build results to the Test environment
9. Integration test
10. Q/A: Quality Assurance on the Test environment
11. Deploy to the Service environment

Test

SW Test

- Static Test (정적 테스트 혹은 분석)
 - 프로그램 실행 전에 코드, 바이너리 등을 대상으로 수행
 - 코드 컨벤션, 버그, 보안 등
 - 장점: 빠르게 수행이 가능함
- Dynamic Test (동적 테스트)
 - 프로그램을 실제 수행하여 결과 및 시스템에 미치는 영향 평가
 - 결과값, 로직의 정상 동작, 자원 (CPU 및 메모리) 사용량 등 평가
 - 단점
 - 실제 수행 환경과 동일한 환경 필요
 - 수행 시간이 오래 걸림
 - 자동화가 어려울 수 있음 (사람의 interaction 이 필요한 경우)

Code Convention

feat. Git-Hooks

MARVEL

CIVIL WAR

CAPTAIN AMERICA



Code convention

- 여러 사람이 함께 작업하는 코드에 대해,
- 가독성을 높이고 유지관리를 보다 용이하게 하기 위해
- 코드의 스타일을 동일한 형태로 맞추는 작업
- 간단하지만 서로의 작업을 불편하게 만드는 예
 - Tab vs. Spaces : 편집기에 따라 매우 불편할 수 있음
 - Naming : 변수의 역할을 파악하는 중요한 역할인데... `int a;`
 - Underbars : C 표준인데, 입력하기 불편함...
- 많은 OSS 프로젝트에서 코딩 스타일을 명시하고 있음
- 어떻게 관리? 강요? 할 것인가?

예) C Coding Standard

C Coding Standard

Adapted from <http://www.possibility.com/Cpp/CppCodingStandard.html> and NetBSD's style guidelines

For the C++ coding standards click [here](#)

Variable Names on the Stack

- use all lower case letters
- use '_' as the word separator.

Justification

- With this approach the scope of the variable is clear in the code.
- Now all variables look different and are identifiable in the code.

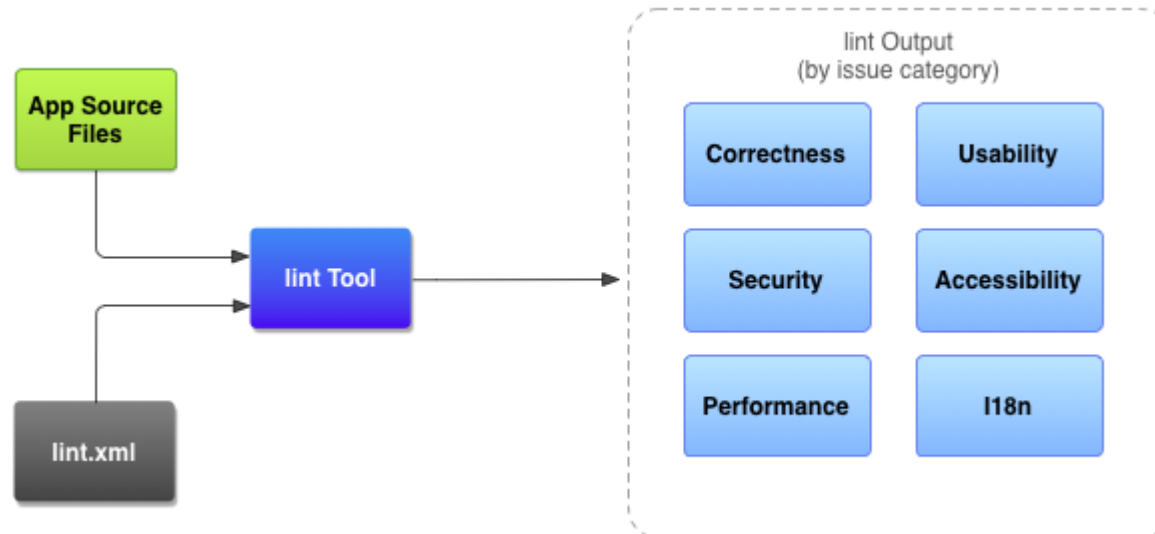
Example

```
int handle_error (int error_number) {  
    int          error= OsErr();  
    Time         time_of_error;  
    ErrorProcessor error_processor;  
}
```

<https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>

Lint

- 코드 스캔 도구, 정적 분석 도구
- 코드 컨벤션의 테스트 용으로 많이 사용
 - 다양한 도구: ESLint and JSLint for JavaScript, Android Lint 등 (참고: <https://atomlinter.github.io/>)
- 그 외 구조 분석을 통해 문법에 맞더라도 잠재적인 버그를 발생시킬 수 있는 코드나, 보안 문제 등을 체크할 수 있음
 - 예) malloc() 했는데 free() 안함
- 사용방법: 체크할 요소를 정의한 환경 파일을 준비하고, 소스 코드를 체크



ESLint

- `sudo npm install -g eslint eslint-config-airbnb-base eslint-plugin-import`
- `eslint --init`

```
root@hcpark:~/chatbot# eslint --init
? How would you like to use ESLint? To check syntax and find problems
? What type of modules does your project use? JavaScript modules (import/export)
? Which framework does your project use? None of these
? Does your project use TypeScript? No
? Where does your code run? Node
? What format do you want your config file to be in? JavaScript
Successfully created .eslintrc.js file in /home/ubuntu/chatbot
root@hcpark:~/chatbot# █
```

- (Trouble shootings)
 - `sudo ln -s /usr/bin/nodejs /usr/local/bin/node`
 - `sudo npm cache clean -f`
 - `npm install -g n`
 - `sudo n stable`
 - `npm init`

초기 설정 완료, 테스트

```
root@hcpark:~/chatbot# vi .eslintrc.js
module.exports = {
  "env": {
    "es6": true,
    "node": true
  },
  "extends": "eslint:recommended",
  "globals": {
    "Atomics": "readonly",
    "SharedArrayBuffer": "readonly"
  },
  "parserOptions": {
    "ecmaVersion": 2018,
    "sourceType": "module"
  },
  "rules": {
  }
};
```

```
root@hcpark:~/chatbot# eslint v1/index.js
root@hcpark:~/chatbot#
```

- 아무 결과 나오지 않음
- 현재 정의된 Rule 이 없기 때문

예) Indentation Check

```
root@hcpark:~/chatbot# vi .eslintrc.js
module.exports = {
  "env": {
    "es6": true,
    "node": true
  },
  "extends": "eslint:recommended",
  "globals": {
    "Atomics": "readonly",
    "SharedArrayBuffer": "readonly"
  },
  "parserOptions": {
    "ecmaVersion": 2018,
    "sourceType": "module"
  },
  "rules": {
    indent: ['error', 2]
  }
};
```

```
ubuntu@hcpark:~/chatbot$ eslint movie.js
/home/ubuntu/chatbot/movie.js
  2:1  error  Expected indentation of 2 spaces but found 1 tab  indent
  3:1  error  Expected indentation of 2 spaces but found 1 tab  indent

* 2 problems (2 errors, 0 warnings)
  2 errors and 0 warnings potentially fixable with the `--fix` option.

ubuntu@hcpark:~/chatbot$
```

Options

This rule has a mixed option:

For example, for 2-space indentation:

```
{
  "indent": ["error", 2]
}
```

Or for tabbed indentation:

```
{
  "indent": ["error", "tab"]
}
```

Airbnb 규칙

<https://github.com/airbnb/javascript>

- Google의 Java 규칙과 더불어 JavaScript 에서 많이 사용하는 규칙
- Naver D2 등 많은 조직에서 도입하여 사용

Airbnb JavaScript Style Guide() {

A mostly reasonable approach to JavaScript

Note: this guide assumes you are using [Babel](#), and requires that you use [babel-preset-airbnb](#) or the equivalent. It also assumes you are installing shims/polyfills in your app, with [airbnb-browser-shims](#) or the equivalent.

downloads **6M/month** downloads **8.7M/month** [gitter](#) [join chat](#)

This guide is available in other languages too. See [Translation](#)

Other Style Guides

- [ES5 \(Deprecated\)](#)
- [React](#)
- [CSS-in-JavaScript](#)
- [CSS & Sass](#)
- [Ruby](#)

Airbnb-base 룰 기반으로 체크

```
root@hpcpark:~/chatbot# vi .eslintrc.js
module.exports = {
  "env": {
    "es6": true,
    "node": true
  },
  "extends": "airbnb-base",
  "globals": {
```

- Extends 에 추가
- Rules에는 아무 내용 없도록 수정 (Airbnb 룰만 사용)

```
ubuntu@hpcpark:~/chatbot$ eslint food.js
```

```
/home/ubuntu/chatbot/food.js
```

1:1	error	Unexpected var, use let or const instead	no-var
1:12	warning	Unexpected unnamed function	func-names
1:20	error	Missing space before function parentheses	space-before-function-paren
1:24	error	A space is required after ','	comma-spacing
2:1	error	Unexpected tab character	no-tabs
2:1	error	Expected indentation of 2 spaces but found 1 tab	indent
2:2	warning	Unexpected console statement	no-console
3:1	error	Unexpected tab character	no-tabs
3:1	error	Expected indentation of 2 spaces but found 1 tab	indent
4:2	error	Missing semicolon	semi

* 10 problems (8 errors, 2 warnings)

6 errors and 0 warnings potentially fixable with the `--fix` option.

자동 수정

```
ubuntu@hcpark:~/chatbot$ eslint food.js --fix
```

```
/home/ubuntu/chatbot/food.js
```

```
1:14  warning  Unexpected unnamed function  func-names
2:3    warning  Unexpected console statement  no-console
```

```
* 2 problems (0 errors, 2 warnings)
```

```
var food = function(rtm,channel) {
  console.log('밥 집을 추천합니다. ');
  rtm.sendMessage('주변 맛집을 추천해드릴게요.', channel);
}

module.exports = food;
```



```
const food = function (rtm, channel) {
  console.log('밥 집을 추천합니다. ');
  rtm.sendMessage('주변 맛집을 추천해드릴게요.', channel);
};

module.exports = food;
```

Lint 도구의 적용 이슈

- 언제 Lint Test를 수행해야 하는가?
 - PR 보내면 리뷰어가?
 - 리뷰어: “야 그 정도는 개발자가 해서 줘야지!”
 - PR 안보내고 Push 해버리면? <- 이걸 나중에 다시 해결하자
 - Test 과정에서?
 - Lint 오류 때문에 테스트까지 갔다가 다시 되돌아오라고?
 - Code convention 오류로 생기는 수많은 커밋은 또 어찌고?
 - Code convention 이 안 맞으면 아예 Commit 이 안되게 하면 어떨까?



Git Hooks: Pre-commit

- Git 은 자체적으로 다양한 hooks 을 제공함
 - Hook: 특정 이벤트가 발생할 때 지정된 코드를 수행시키는 기법
 - Git 기본 포함 hooks: `/.git/hooks` 에 정의. 매우 다양한 상황에 적용 가능
 - `.sample` 을 제거하면 바로 동작함
 - <https://git-scm.com/book/ko/v1/Git%EB%A7%9E%EC%B6%A4-Git-%ED%9B%85>
 - http://woowabros.github.io/tools/2017/07/12/git_hook.html
- Pre-commit Hook
 - 커밋이 수행될 때 코드를 수행하고, 결과에 따라 commit 수행 여부를 제어
 - Lint 테스트를 삽입하고, 통과하지 못하면 Commit 이 안되게 하자

Git Hooks Directory 및 예제

```
root@hcpark:~/chatbot/.git/hooks# ls -al
```

```
total 56
```

```
drwxrwxr-x 2 ubuntu ubuntu 4096 Nov 10 13:57 .
```

```
drwxrwxr-x 8 ubuntu ubuntu 4096 Nov 10 13:57 ..
```

```
-rwxrwxr-x 1 ubuntu ubuntu 478 Nov 10 13:57 applypatch-msg.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 commit-msg.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 fsmonitor-watchman.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 post-update.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 pre-applypatch.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 pre-commit.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 pre-push.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 pre-rebase.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 pre-receive.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 prepare-commit-msg.sample
```

```
-rwxrwxr-x 1 ubuntu ubuntu 806 Nov 10 13:57 update.sample
```

```
root@hcpark:~/chatbot/.git/hooks# vi pre-commit.sample
```

```
#!/bin/sh
```

```
#
```

```
# An example hook script to verify what is about to be committed.
```

```
# Called by "git commit" with no arguments. The hook should
```

```
# exit with non-zero status after issuing an appropriate message if
```

```
# it wants to stop the commit.
```

```
#
```

```
# To enable this hook, rename this file to "pre-commit".
```

```
if git rev-parse --verify HEAD >/dev/null 2>&1
```

```
then
```

```
    against=HEAD
```

```
else
```

```
    # Initial commit: diff against an empty tree object
```

```
    against=4b825dc642cb6eb9a060e54bf8d69288fbee4904
```

```
fi
```

```
# If you want to allow non-ASCII filenames set this variable to true.
```

```
allownonascii=$(git config --bool hooks.allownonascii)
```

```
# Redirect output to stderr.
```

Pre-commit 설정 (예제)

- 디렉토리 내 모든 .js 파일에 대해 eslint 수행
- 프로그램의 수행 결과값이 0 이 아니라면 (비정상 종료)
 - Exit 1 로 즉각 종료하여, commit 이 수행되지 않도록 함

```
#!/bin/sh

eslint . --ext .js
if [ $? -eq 1 ];
then
    echo "코드 컨벤션 테스트 실패! 커밋이 취소되었습니다."
    exit 1
else
    echo "코드 컨벤션 테스트 통과!"
fi
```

```
ubuntu@hcpark:~/chatbot$ git add .
ubuntu@hcpark:~/chatbot$ git commit -m "일부러 Tab 문자를 넣어봤습니다."

/home/ubuntu/chatbot/v3/index.js
18:1  error  Expected indentation of 4 spaces but found 1 tab  indent

* 1 problem (1 error, 0 warnings)
1 error and 0 warnings potentially fixable with the `--fix` option.

코드 컨벤션 테스트 실패! 커밋이 취소되었습니다.
ubuntu@hcpark:~/chatbot$
```

Husky: Git Hook management manger

- Bash shell script 가 익숙하지 않은 경우
- 사용법
 - Sudo npm install husky
 - package.json 에 오른쪽과 같이 설정
 - 끝!
- 다양한 hooks 들을 쉽게 관리

```
ubuntu@hpcpark:~/chatbot$ git commit -m "일부러 Tab 문자를 넣어봤습니다."
husky > pre-commit (node v8.10.0)

/home/ubuntu/chatbot/v3/index.js
18:1  error  Expected indentation of 4 spaces but found 1 tab  indent

* 1 problem (1 error, 0 warnings)
1 error and 0 warnings potentially fixable with the `--fix` option.

husky > pre-commit hook failed (add --no-verify to bypass)
```

```
ubuntu@hpcpark:~/chatbot$ vi package.json
```

```
{
  "name": "chatbot",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "eslint": "^6.6.0",
    "eslint-plugin-import": "^2.18.2",
    "eslint-plugin-jsx-ally": "^6.2.3",
    "eslint-plugin-react": "^7.16.0",
    "mocha": "^6.2.2"
  },
  "husky": {
    "hooks": {
      "pre-commit": "eslint . --ext .js"
    }
  },
  "dependencies": {
    "@slack/rtm-api": "^5.0.3",
    "@slack/web-api": "^5.4.0",
    "assert": "^2.0.0",
    "chai": "^4.2.0",
    "dotenv": "^8.2.0",
    "eslint-config-airbnb": "^18.0.1",
    "slack-client": "^2.0.6"
  }
}
```

(하는 김에 하나 더...) Push 금지

- 항상 코드 리뷰와 테스트 등 관리 절차를 거친 후에,
- Merge 가 이루어질 수 있도록 Push를 강제로 금지
 - 비록 권한이 있더라도! PR을 사용하자
- Husky pre-push hook 이용, master branch 에 대한 push 를 금지


```
"husky": {
  "hooks": {
    "pre-commit": "eslint . --ext .js",
    "pre-push": "git rev-parse --abbrev-ref HEAD | grep -v master"
  }
},
```

```
ubuntu@hcpark:~/chatbot$ vi package.json
ubuntu@hcpark:~/chatbot$ git push
Username for 'https://github.com': hyunchan-park
Password for 'https://hyunchan-park@github.com':
husky > pre-push (node v8.10.0)
husky > pre-push hook failed (add --no-verify to bypass)
error: failed to push some refs to 'https://github.com/hyunchan-park/chatbot.git'
```

Development Process



So far...

1. Development
 2. Code Convention Check & Unit test
 3. Commit
 4. Pull Request
 5. Code review
 6. Merge
 7. Integration and Build (on the Build environment, deployed by git clone)
 8. Deploy the build results to the Test environment
 9. Integration test
 10. Q/A: Quality Assurance on the Test environment
 11. Deploy to the Service environment
- 
- 현재까지 진행된 부분

Build & Test environment 구성

- 본래는 빌드 및 테스트용 환경을 따로 구성하는게 좋음
 - 보통 Unit test 는 개발 환경에서 진행
 - 점진적인 빌드 과정에서 integration test를 병행해서 진행할 수 있음
 - 혹은 빌드 서버를 따로 구성하고, 테스트 환경과 연동해서 진행 가능
- (우리는 생략)
 - 빌드 필요없음
 - 테스트 환경: 그냥 개발 환경에서 수행하자



Dynamic Test

1. Unit Test
2. Integration Test



Dynamic Test: Unit test and Integration Test

- Unit test

* Wikipages: [Unit test](#) and [Integration test](#)

- 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차
- 모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하고, 개발 이후 이를 검증하도록 함
- 언제라도 코드 변경으로 인해 문제가 발생할 경우, 단시간 내에 문제 모듈을 파악하고 바로 잡을 수 있음

- Integration test

- Unit test 를 통과한 모듈들을 결합하는 단계에서, 상호간의 동작이 정상적으로 수행되는지 검증함
- 모듈들을 점진적으로 통합해가며 테스트할 수 있도록 테스트 설계

* 일반적으로 입력값에 대한 출력값을 검증하는 Black-box 형식으로 작성

(참고) TDD: Test-driven development

- 테스트 주도 개발: “테스트를 먼저 만들고, 이를 통과하는 코드를 작성”
 - 각 모듈의 spec 을 정의하고, (Input, output, behavior)
 - Spec 에 따른 테스트 케이스를 먼저 작성한 후,
 - 개발 결과물이 테스트를 통과하도록 강제하는 개발 방식
- “테스트 코드 == Spec”
 - 잘 동작하는 TDD 에서는 테스트 코드를 보면 spec을 파악할 수 있음
- 효과 및 필요성
 - 테스트에 대한 강조. 결함을 줄이기 위함
 - 개발과정에서 요구사항이 계속해서 변경되는 경우, 명확한 명세를 정의하고 맞추기 위해 테스트 코드를 계속 관리

Unit Test example

```
namespace NetCoreTest
{
    public class Tests
    {
        [Fact]
        public void Test1()
        {
            var calculator = new Calculator();
            var result = calculator.Add(20, 22);
            Assert.Equal(42, result);
        }
    }
}
```

Mocha

<https://heropy.blog/2018/03/16/mocha/>

- Node.js 테스트 프레임워크 (framework)
 - 기본적으로 테스트도 일반 언어로 작성 가능함
 - If-else 구문을 통해 입력값에 대한 결과값을 검사하면 됨
 - 테스트 프레임워크는 테스트에 필요한 구문의 형태를 좀더 편리하게 사용할 수 있는 기능들을 제공함
 - 따라서 테스트 동작을 간결하게 표현하여 보다 직관적인 테스트 코드를 구현하도록 도움
 - 결과도 테스트 수행 형태로 쉽고 깔끔하게 출력해줌
 - 그 외 테스트 코드에 필요한 다양한 기능을 플러그인 등을 통해 제공
- 설치
 - `sudo npm install mocha -g`

Mocha 사용 예제: Without Mocha

호출하면 “hello” 를 출력해야 하는 함수 sayHello() 를 테스트

```
const sayHello = require('../app').sayHello;

if (sayHello) {
  console.log('sayHello should return "hello"');
  if (sayHello() === 'hello') {
    console.log('Success');
  } else {
    console.log('Fail');
  }
}
```


Mocha 사용 예제: With Mocha

* 호출하면 “hello” 를 출력해야 하는 함수 sayHello() 를 테스트

```
ubuntu@hcpark:~/chatbot$ vi app.spec.js
```

```
const sayHello = require('./app').sayHello;
const assert = require('assert');

describe('App test!', function () {
  it('sayHello should return hello', function (done) {
    assert.equal(sayHello(), 'hello');
    done();
  });
});
```

```
ubuntu@hcpark:~/chatbot$ vi app.js
```

```
module.exports = {
  sayHello: function () {
    return 'hello';
  }
}
```

Mocha 사용 예제: With Mocha

* 결과 화면: 성공 및 실패

```
ubuntu@hcpark:~/chatbot$ mocha app.spec.js

App test!
  ✓ sayHello should return hello

1 passing (9ms)

ubuntu@hcpark:~/chatbot$ mocha app.spec.js

App test!
  1) sayHello should return hello

0 passing (9ms)
1 failing

1) App test!
   sayHello should return hello:

AssertionError [ERR_ASSERTION]: 'helloo' == 'hello'
+ expected - actual

-helloo
+hello

at Context.<anonymous> (app.spec.js:6:12)
```

Mocha: Chatbot Test

```
ubuntu@hcpark:~/chatbot$ vi test.spec.js

require('dotenv').config();

const token = process.env.SLACK_TOKEN;
const tchannel = process.env.TESTING_CHANNEL;
const { RTMClient, LogLevel } = require('@slack/rtm-api');

const rtm = new RTMClient(token, {
  // logLevel: LogLevel.DEBUG,
});

(async () => {
  await rtm.start()
    .catch(console.error);
})();

const food = require('./food');
const movie = require('./movie');
const assert = require('assert');

var res;
describe('유닛 테스트를 시작합니다.', async function () {
  before(async function() {
    return res = await food(rtm, tchannel);
  });

  it('밥 메시지 테스트', function (done) {
    assert.equal ( res, 'success');
    done();
  });
});
```

```
ubuntu@hcpark:~/chatbot$ vi food.js

const food = async function (rtm, channel) {
  try {
    const res = await rtm.sendMessage('주변 맛집')
    //console.log("호출");
    return Promise.resolve('success');
  } catch (error) {
    console.log("에러!", error.data);
    return Promise.resolve('error');
  }
};

module.exports = food;
```

- 따라하지 않아도 됨
- 유닛 테스트는 현재 구조에서 큰 의미는 없음
(실제로 사용자가 해당 메시지를 받았는지가 더 중요함)

Mocha: Chatbot Test 결과

```
ubuntu@hcpark:~/chatbot$ mocha test.spec.js
```

```
유닛 테스트를 시작합니다 .  
✓ 밥 메시지 테스트
```

```
1 passing (1s)
```

```
유닛 테스트를 시작합니다 .  
에러! undefined
```

```
1) 밥 메시지 테스트
```

```
0 passing (32ms)  
1 failing
```

```
1) 유닛 테스트를 시작합니다 .  
   밥 메시지 테스트 :
```

```
AssertionError [ERR_ASSERTION]: 'error' == 'success'  
+ expected - actual
```

```
-error  
+success
```

```
at Context.<anonymous> (test.spec.js:27:12)
```

Dynamic Test

1. Unit Test
2. Integration Test



Integration Test for Chatbot

- Input: 사용자가 입력하는 메시지
 - 영화, 밥, 놀이, 그 외 아무 메시지
- Output: 입력 메시지에 따른 출력
 - 예) 영화: 취향에 맞춘 영화를 추천해드릴게요.
- 이슈
 - 출력 방향: slack channel
 - Slack 에 정상적인 메시지가 출력된다는 것을 어떻게 확인할 것인가?
 - 테스트 환경 구성에서 가장 중요한 부분: 최종 사용자 입장에서 확인
 - End-to-end test
- (Unit test 는 생략: 현재 구조는 너무 단순하므로)
- (Mocha 적용도 어려움. 현재 구조는 메시지가 전달되면 event 로 받기 때 문에, mocha의 순차적인 테스트 구조에 적용하기 어려움)

Testbot 추가

- Slack 을 통해 Input 을 주고, Output 을 검사할 수 있는 Testbot 추가
 - 사람이 하면 자동화가 안되니까, test 를 하는 bot 을 추가로 생성
- 기존 workspace 에 testing 채널 추가 (private)
- 채널에 기존 bot 초대
 - 채널에 대해 app 을 추가하면 됨
- 기존 bot 만드는 방식을 이용해서 test용 bots 추가
 - Bots 검색해도 안 나오면 “view app directory” 버튼 이용해서 bots 선택
 - Token 가져와서 똑같이 사용
- 최종: 해당 채널에 나, Chatbot, Testbot 이 있으면 성공



2019 OSS CBNU

Hyunchan Park

Jump to... Ctrl+K

Channels

2019-oss-chatbot
 # chatbot-jenkins
 # general
 # random
testing

testing2

+ Add a channel

Direct Messages

Slackbot
 Hyunchan Park (you)

+ Invite people

Apps

noticebot
 testerbot
 + Add apps

#testing

☆ 1 | 👤 0 | Add a topic

Friday, November 8th

주변 맛집을 수선애드릴게요.

고만해.

안녕하세요. 영화,밥,놀이 중에 말씀해주세요.

취향에 맞춘 영화를 추천해드릴게요.

Saturday, November 9th

new messages



testerbot APP 10:03 PM

기본메시지

밥



noticebot APP 10:03 PM

안녕하세요. 영화,밥,놀이 중에 말씀해주세요.



testerbot APP 10:03 PM

놀이



noticebot APP 10:03 PM

주변 맛집을 추천해드릴게요.



testerbot APP 10:03 PM

영화



noticebot APP 10:03 PM

고만해.

취향에 맞춘 영화를 추천해드릴게요.



testerbot APP 10:07 PM

영화

기본메시지

Search

About this channel

Channel Details

Highlights

Pinned Items

1 Member

2 Apps

noticebot APP

testerbot APP

Add App

Shared Files

Notification Preferences

Testing 을 위한 채널 개설 및 활용

- 이슈
 - 현재 로직
 - 채널에 메시지가 도착하면 이벤트가 발생(RTM_EVENTS.MESSAGE)
 - 이 메시지에서 Channel 정보를 얻고, 해당 채널에 응답을 보낼 수 있음
 - 문제점
 - Testbot이 메시지를 보내서 Chatbot 을 테스트해야 하는데, 먼저 메시지를 받지 못하면 채널 정보를 알 수 없어 메시지를 보내지 못함
- 채널 ID 받아오기
 - 먼저 채널 ID가 변하지 않는지 확인
 - 기존 Chatbot 로직을 이용해서 Testing 채널의 ID 출력
 - .env 에 ID를 기록하고, testbot.js 에서 이를 로딩해 활용

```
SLACK_TOKEN='xoxb-824792789584-813333340835-  
SLACK_TESTER_TOKEN='xoxb-824792789584-827639  
TESTING_CHANNEL='GQ8UA3LAG'
```

```
ubuntu@hcpark:~/chatbots$ nodejs index.js  
GQ8UA3LAG
```

test.js (1/2)

```
ubuntu@hcpark:~/chatbot/test$ vi test.js
```

```
require('dotenv').config();
```

```
var status = 0;
```

```
const token = process.env.SLACK_TESTER_TOKEN;
```

```
const tchannel = process.env.TESTING_CHANNEL;
```

```
const tuser = process.env.TESTING_USER;
```

```
const { RTMClient, LogLevel } = require('@slack/rtm-api');
```

```
const rtm = new RTMClient(token, {  
  // logLevel: LogLevel.DEBUG,  
});
```

```
rtm.start()  
  .catch(console.error);
```

```
rtm.on('ready', async () => {
```

```
  const res = await rtm.sendMessage( "테스트를 시작합니다.", tchannel);
```

```
  console.log("보낸 메시지 : 테스트를 시작합니다.");
```

```
  status++;
```

```
});
```

* 슬랙과 정상 접속되면 발생하는 ready 이벤트를 이용해 먼저 말을 하도록 함

test.js (2/2)

```
rtm.on('message', function (message) {  
  var text = message.text;  
  if ( message.user == tuser) {  
    switch (status) {  
      case 1:  
        if(text != "안녕하세요 . 영화 ,밥 ,놀이 중에 말씀해주세요 .") {  
          console.log( "테스트 실패 : 기본 메시지 " );  
          process.exit(1);  
        }  
        console.log("받은 메시지 :", text);  
        rtm.sendMessage("영화 ", tchannel);  
        status++;  
        break;  
    }  
  }  
});
```

* Status 를 하나씩 전진시키면서, 각 테스트 케이스의 응답을 확인하고,
다음 테스트를 위한 메시지를 전달함

test.js (2/2) rtm.on('message') 전체

* 텍스트들을 다른 파일로 관리하고 챗봇 프로그램과 공유하면 더 좋을 듯

```
rtm.on('message', function (message) {  
  var text = message.text;  
  if ( message.user == tuser) {  
    switch (status) {  
      case 1:  
        if(text != "안녕하세요 . 영화 ,밥 ,놀이 중에 말씀해주세요 .") {  
          console.log( "테스트 실패 : 기본 메시지 " );  
          process.exit(1);  
        }  
        console.log("받은 메시지 :", text);  
        rtm.sendMessage("영화 ", tchannel);  
        status++;  
        break;  
      case 2:  
        console.log("보낸 메시지 : 영화 ");  
        if(text != "취향에 맞는 영화를 추천해드릴게요 .") {  
          console.log( "테스트 실패 : 영화 " );  
          process.exit(1);  
        }  
        console.log("받은 메시지 :", text);  
        rtm.sendMessage("밥 ", tchannel);  
        status++;  
        break;  
      case 3:  
        console.log("보낸 메시지 : 밥 ");  
        if(text != "주변 맛집을 추천해드릴게요 .") {  
          console.log( "테스트 실패 : 밥 " );  
          process.exit(1);  
        }  
        console.log("받은 메시지 :", text);  
        rtm.sendMessage("놀이 ", tchannel);  
        status++;  
        break;  
      case 4:  
        console.log("보낸 메시지 : 놀이 ");  
        if(text != "고만해 .") {  
          console.log( "테스트 실패 : 놀이 " );  
          process.exit(1);  
        }  
        console.log("받은 메시지 :", text);  
        console.log("테스트가 정상 종료되었습니다 .");  
        process.exit(0);  
        break;  
      default:  
        console.log("테스트가 이상 상태입니다 .");  
    }  
  }  
});
```

수행 결과

```
ubuntu@hcpark:~/chatbot/test$ nodejs test.js
```

보낸 메시지 : 테스트를 시작합니다 .

받은 메시지 : 안녕하세요 , 영화 ,밥 ,놀이 중에 말씀해주세요 .

보낸 메시지 : 영화

받은 메시지 : 취향에 맞춘 영화를 추천해드릴게요 .

보낸 메시지 : 밥

받은 메시지 : 주변 맛집을 추천해드릴게요 .

보낸 메시지 : 놀이

받은 메시지 : 그만해 .

테스트가 정상 종료되었습니다 .

수행 결과 (실패했을 때)

```
ubuntu@hcpark:~/chatbot$ vi movie.js
```

* 일부러 movie.js 에 메시지 하나 더 추가

```
var movie = function(rtm,channel) {  
  console.log('영화를 추천합니다.');
```

```
  rtm.sendMessage('취향에 맞춘 영화를 추천해드릴게요.', channel);  
  
  //1.추천을 위한 취향 파악 #주석1  
  rtm.sendMessage('좋아하는 영화 하나를 알려주실래요?', channel);  
}  
  
module.exports = movie;
```

```
ubuntu@hcpark:~/chatbot/test$ nodejs test.js
```

보낸 메시지 : 테스트를 시작합니다 .

받은 메시지 : 안녕하세요 . 영화 ,밥 ,놀이 중에 말씀해주세요 .









보낸 메시지 : 영화







받은 메시지 : 취향에 맞춘 영화를 추천해드릴게요 .

보낸 메시지 : 밥

테스트 실패 : 밥

수행 결과

-  **testerbot** APP 12:03 PM
테스트를 시작합니다.
-  **noticebot** APP 12:03 PM
안녕하세요. 영화,밥,놀이 중에 말씀해주세요.
-  **testerbot** APP 12:03 PM
영화
-  **noticebot** APP 12:03 PM
취향에 맞춘 영화를 추천해드릴게요.
-  **testerbot** APP 12:03 PM
밥
-  **noticebot** APP 12:03 PM
주변 맛집을 추천해드릴게요.
-  **testerbot** APP 12:03 PM
놀이
-  **noticebot** APP 12:03 PM
고만해.

-  **testerbot** APP 12:08 PM
테스트를 시작합니다.
-  **noticebot** APP 12:08 PM
안녕하세요. 영화,밥,놀이 중에 말씀해주세요.
-  **testerbot** APP 12:08 PM
영화
-  **noticebot** APP 12:08 PM
취향에 맞춘 영화를 추천해드릴게요.
좋아하는 영화 하나를 알려주실래요?
-  **testerbot** APP 12:08 PM
밥
-  **noticebot** APP 12:08 PM
주변 맛집을 추천해드릴게요.

Open!



Upload to GitHub

- 현재 진행된 소스 코드들을 올리고,
 - 설정된 Hook 등 개발에 필요한 정보들을 포함하여 문서를 작성하고,
 - 프로젝트를 오픈! (광고하러 가자)
-
- (+) 슬랙을 통해 issue, pull request 등 알람을 받을 수 있게 설정
 - Slack workspace -> Add apps
 - GitHub 검색 후 install
 - All public channels 에 메시지를 보낼 수 있는 권한 부여
 - 설치 후, GitHub 대화창에 /github signin 입력하여 GitHub 계정 연동
 - 아래와 같이 특정 리포지토리에 대한 정보를 수신하도록 설정
 - /github subscribe hyunchan-park/chatbot
 - 해당 repo 로 이동하여 GitHub에도 slack app 이 설치 및 설정되도록 수행



GitHub 설치 및 계정 연동 버튼

The screenshot shows a Slack workspace interface. On the left is a dark purple sidebar with the workspace name '2019 OSS CBNU' and a user 'Hyunchan Park'. Below this are sections for 'Channels' (including # 2019-oss-chatbot, # chatbot-jenkins, # general, # random, # testing, and # testing2), 'Direct Messages' (including Slackbot and Hyunchan Park (you)), and 'Apps' (including GitHub, noticebot, and testerbot). The main area shows a direct message conversation with the GitHub bot. The bot's profile is at the top with the GitHub logo and name. The message history shows a welcome message from the bot: 'This is the very beginning of your direct message history with @GitHub. Only the two of you are in this conversation, and no one else can join it. Learn more'. Below this is a separator line for 'Today' with a 'new messages' badge. The bot's latest message, timestamped '10:28 PM', says: 'You've successfully installed GitHub on this Slack workspace 🎉 To subscribe a channel to a repository use the following slash command: /github subscribe owner/repository Looking for additional help? Try /github help'. Below the message is a status bar indicating 'Only visible to you' and a prompt to 'Finish connecting your GitHub account' with a 'Connect GitHub account' button. At the bottom is a message input field with a placeholder 'Message channel' and various formatting icons (link, bold, italic, linkify, code, list, indent, outdent, linkify, emoji).

2019 OSS CBNU ▾
Hyunchan Park

Jump to...

Channels +

- # 2019-oss-chatbot
- # chatbot-jenkins
- # general
- # random
- # testing
- # testing2

+ Add a channel

Direct Messages +

- ♥ Slackbot
- Hyunchan Park (you)

+ Invite people

Apps +

- GitHub
- noticebot
- testerbot
- + Add apps

GitHub ☆

Messages About

GitHub ● APP

This is the very beginning of your direct message history with @GitHub. Only the two of you are in this conversation, and no one else can join it. [Learn more](#)

🔗 [How does GitHub work?](#)

Today new messages

GitHub APP 10:28 PM

You've successfully installed GitHub on this Slack workspace 🎉

To subscribe a channel to a repository use the following slash command:

`/github subscribe owner/repository`

Looking for additional help? Try `/github help`

👁 Only visible to you


Finish connecting your GitHub account

[Connect GitHub account](#)



Message channel

🔗 B I 🔗 </> ☰ ☷ ☹ 📎 Aa @ 😊

GitHub 에서 접근 권한 부여



Slack by **GitHub** would like access to:

-  **Your account** (hyunchan-park)
Verify your GitHub account
-  **Resources**
Determine what resources both you and Slack can access

Slack has been installed on 1 account you have access to: jcloud-devops.
[Learn more about Slack](#)

Authorize Slack by GitHub

Authorizing will redirect to
<https://slack.github.com>

Repo 설정 화면

Today



GitHub APP 10:28 PM

You've successfully installed GitHub on this Slack workspace 🎉

To subscribe a channel to a repository use the following slash command:

`/github subscribe owner/repository`

Looking for additional help? Try `/github help`

👁 Only visible to you

Finish connecting your GitHub account

[Connect GitHub account](#)

✓ Success! [@Hyunchan Park](#) is now connected to [@hyunchan-park](#)

Either the app isn't installed on your repository or the repository does not exist. Install it to proceed.

[Install GitHub App](#)

new messages


GitHub에서 Slack 설정 및 완료

Pull requests

Issues

Marketplace

Explore



Install Slack

Install on your personal account Hyunchan, Park

☒ All repositories

This applies to all current *and* future repositories.

☐ Only select repositories

...with these permissions:

✓ Read access to code


✓ Read access to checks, commit statuses, metadata, and repository projects

✓ Read and write access to deployments, issues, and pull requests

Install

Cancel

Next: you'll be directed to the GitHub App's site to complete setup.




GitHub

APP

10:28 PM

You've successfully installed GitHub on this Slack workspace 🎉
To subscribe a channel to a repository use the following slash command
`/github subscribe owner/repository`
Looking for additional help? Try `/github help`



GitHub

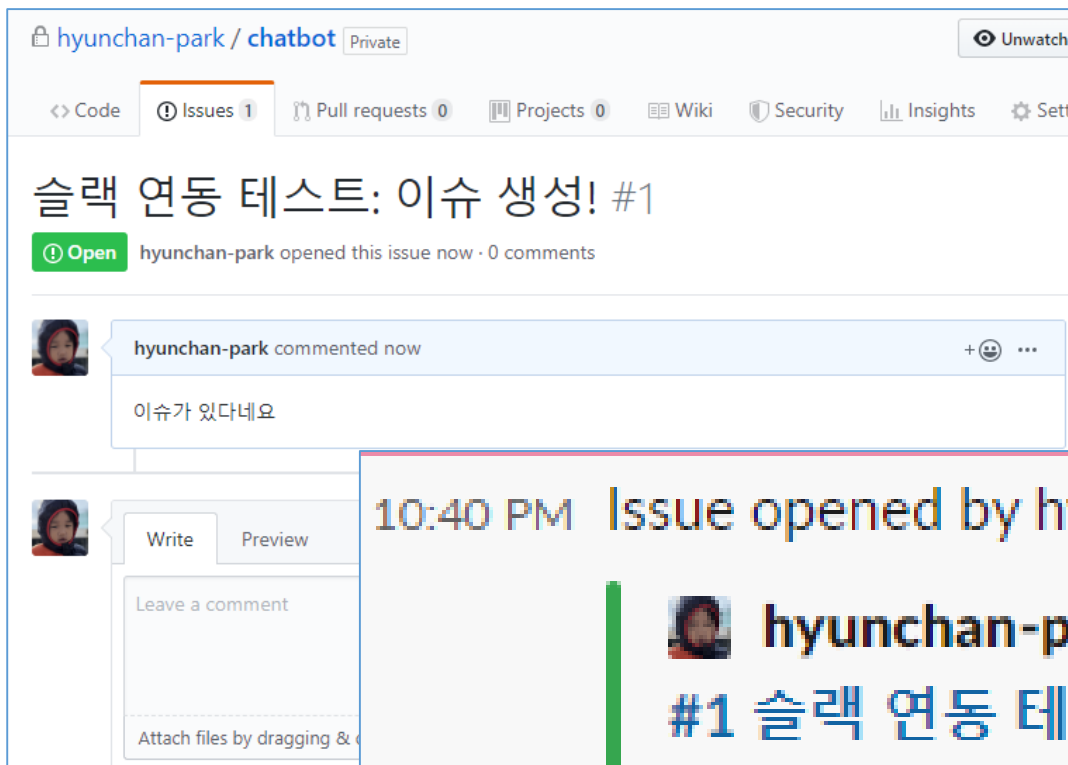
APP

10:37 PM

Subscribed to [hyunchan-park/chatbot](#)

53

테스트: 이슈 생성



10:40 PM Issue opened by hyunchan-park



hyunchan-park

#1 슬랙 연동 테스트: 이슈 생성!

이슈가 있네요



hyunchan-park/chatbot | Today at 10:40 PM

Service Environment 구성

- J-Cloud instance 추가 생성
 - 서비스 용으로만 사용되는 서버
 - 본래는 빌드 결과물인 바이너리를 받아와서 서비스를 재시작하는 동작을 해야하지만,
 - 우리는 빌드가 없으므로 그냥 git pull 로 새 버전을 받아와서 서비스를 재시작하는 동작을 수행
 - 사실 챗봇도 테스트용을 따로 만들어서 다른 Token 으로 관리해야 함
 - 기존 서비스가 동작 중에도 개발 과정이 진행되어야 하므로



개발 프로세스 준비 완료!

1. Development
2. Code Convention Check & Unit test
3. Commit
4. Pull Request
5. Code review
6. Merge
7. Integration and Build (on the Build environment, deployed by git clone)
8. Deploy the build results to the Test environment
9. Integration test
10. Q/A: Quality Assurance on the Test environment
11. Deploy to the Service environment



개인 과제 #11 : ChatBot 실습

- 다음 항목들 캡처 (ppt에 간단히 제목 붙여서 작성 후, PDF로 저장, 제출)
 - Chabot 최종 코드 (3개 파일)
 - ESLint 수행 화면 (간단한 에러를 발생 시킬 것)
 - Husky Git hooks 설정 화면 (package.json 에서 husky 부분만)
 - Integration test 코드 및 수행 화면
 - GitHub 프로젝트 화면
 - Slack과 GitHub 연동하고, 이슈를 생성해 Slack에 알림온 화면
- 제출 기한:
 - 11/24 (일) 23:59
 - 지각 감점: 5%p / day (3주 내 제출해야 함)

