와플스튜디오 x easi6
node.js 세미나

express 웹서버 제작 실습 1

easi6 CTO 한재화
(와플스튜디오 2기)

2017. 11. 15

오늘 얘기할 것들

- 프로젝트 설명
  - 음식 딜리버리 서비스를 위한 웹서버
- 코딩 따라하기
  - 개발환경 셋업
  - 실제 코딩
- 생각해볼 문제
- Q&A

음식 딜리버리를 위한 웹서버

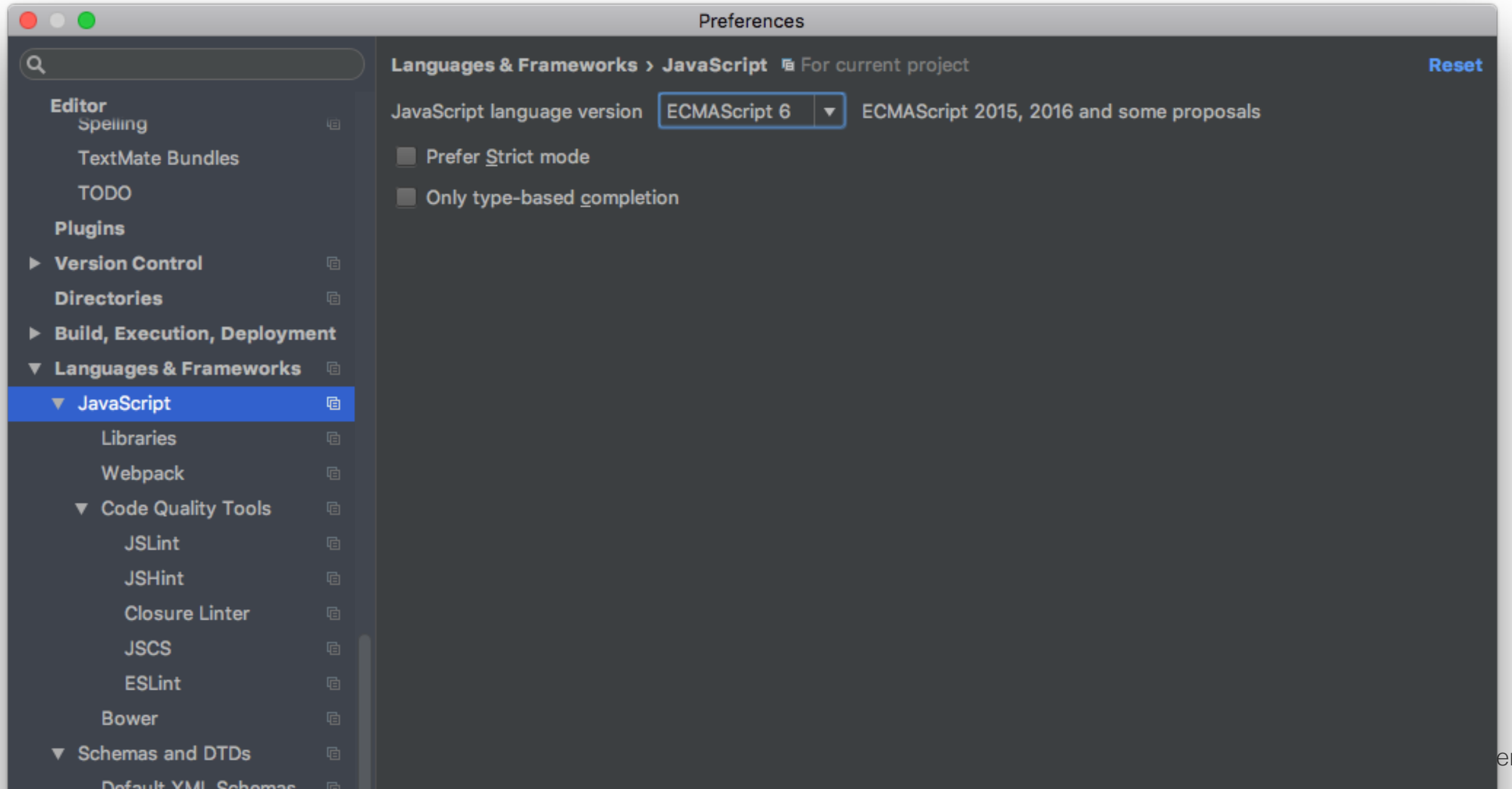
- my delivery
- 2주짜리 따라하기 수업
  - 1주차(2시간) : 셋업 및 웹서버 구동, 기초 인증까지
  - 2주차(2시간) : 비지니스 로직 작성 및 route 연결 위주
- 스펙 리스트
  - 회원가입, 로그인, 정보수정 등 회원관리가 가능해야
  - 메뉴를 선택 후에 주문을 하고 결제가 되어야
    - 결제는 카드결제만 받습니다.
  - 현재 주문의 상황을 추적할 수 있어야
    - 주문상황: 결제대기, 결제됨, 준비중, 배달출발, 배달완료

실습에서 쓰일 tech stack

- 언어: es6 (babel로 트랜스파일)
- 구동 엔진: Node.js v8.x
- HTTP 서버 레이어: express.js v4.x
- DBMS: mysql v5.7.x
- ORM: sequelize v4
- 추가 storage: Redis (인증 토큰 저장용)
- 인증 레이어 : passport + JWT + bcrypt

실습 환경 설정

- IDE 셋업
  - ~~vim~~ WebStorm 추천 (학생들은 공짜. snu.ac.kr 메일만 있으면)
  - 웹스톰에서 언어 셋업을 아래와 같이 ECMAScript6 (es6)로 맞춰줍니다.



실습 환경 설정 - package.json

- node 프로젝트에 대한 설명 및 사용할 라이브러리 셋업 혹은 사용자 정의 스크립트 정의 가능
- 이전 슬라이드에서 말했던 라이브러리들을 설치
- babel transpile을 위한 기타 모듈들도 설치 (dev dependencies 쪽)
- 테스트를 위한 모듈들 설치 (mocha, chai, sequelize-fixture 등)
- 지금 여기서 모르는 것들이 나와도 일단 복사/붙여넣기 하고 넘어갑시다.

```
{
  "name": "mydelivery",
  "version": "0.0.1-SNAPSHOT",
  "description": "food delivery service API server",
  "main": "server.js",
  "scripts": {
    "load_fixture": "NODE_ENV=test node -r 'regenerator-runtime/runtime' -r babel-register ./load_fixture_data.js",
    "test": "NODE_ENV=test npm run load_fixture && NODE_ENV=test node_modules/.bin/mocha --ui bdd --recursive test",
    "start": "PORT=9000 INTERFACE=0.0.0.0 node -r \"regenerator-runtime/runtime\" -r babel-register $NODE_DEBUG_OPTION server.js",
    "cover": "NODE_ENV=test node_modules/.bin/nyc npm test",
    "babel": "babel ./ --out-dir dist --ignore node_modules,dist,scripts,test,flow-typed --source-maps --presets=env,stage-2,flow",
    "migrate": "sequelize --migrations-path=./migrations --config config/database.json db:migrate",
    "migrate:undo": "sequelize --migrations-path=./migrations --config config/database.json db:migrate:undo"
  },
  "author": "Jaehwa Han (drh@easi6.com)",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^1.0.3",
    "bluebird": "^3.5.0",
    "body-parser": "^1.17.2",
    "config": "^1.26.2",
    "cookie-parser": "^1.4.3",
    "cors": "^2.8.4",
    "express": "^4.15.4",
    "express-paginate": "^0.3.0",
    "express-session": "^1.15.5",
    "express-wrap-async": "^1.0.3",
    "fs-extra": "^4.0.2",
    "glob": "^7.1.2",
    "jsonwebtoken": "^7.4.3",
    "lodash": "^4.17.4",
```

실습환경 설정 - contd.

- package.json (계속)

```
    "moment": "^2.18.1",
    "morgan": "^1.9.0",
    "mysql2": "^1.4.1",
    "nodemailer": "^4.3.1",
    "oauth2orize": "^1.10.0",
    "passport": "^0.4.0",
    "passport-http": "^0.3.0",
    "passport-http-bearer": "^1.0.1",
    "passport-jwt": "^3.0.0",
    "passport-local": "^1.0.0",
    "passport-oauth2-client-password": "^0.1.2",
    "request-promise": "^4.2.2",
    "sequelize": "^4.7.0",
    "sequelize-fixtures": "^0.6.0",
    "winston": "^2.3.1",
    "winston-daily-rotate-file": "^1.4.6"
  },
  "devDependencies": {
    "babel-cli": "^6.26.0",
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-polyfill": "^6.26.0",
    "babel-preset-env": "^1.6.1",
    "babel-preset-flow": "^6.23.0",
    "babel-preset-stage-2": "^6.24.1",
    "babel-register": "^6.26.0",
    "chai": "^4.1.1",
    "chai-as-promised": "^7.1.1",
    "mocha": "^3.5.0",
    "nyc": "^11.1.0",
    "regenerator-runtime": "^0.11.0"
  }
}
```

실습환경 설정 - yarn 패키지 관리자

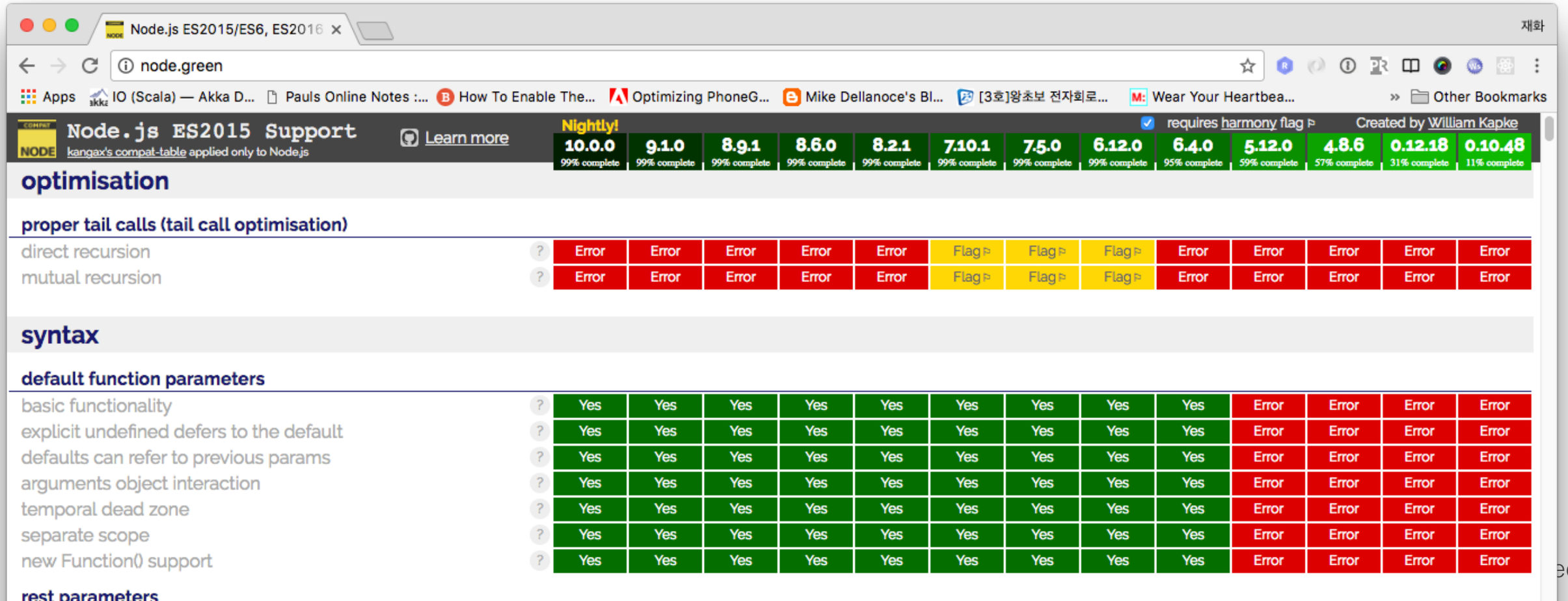
- 전통적인 패키지 관리자는 npm
  - 참고로 node 8에 딸려오는 npm v5는 문제가 많음 (아직도 그런가?)
    - ex) react 개발환경과 안맞거나 설치가 안되는 라이브러리 존재
- yarn - npm 대안으로 나온 패키지 관리자

3. drh@drh-MacBook-Pro: ~/Project/mydelivery (zsh)

```
{11:00}~/Project ➪
{11:00}~/Project ➪ mkdir mydelivery
{11:01}~/Project ➪ cd mydelivery
{11:01}~/Project/mydelivery ➪ sudo npm install -g yarn
Password:
/usr/local/bin/yarn -> /usr/local/lib/node_modules/yarn/bin/yarn.js
/usr/local/bin/yarnpkg -> /usr/local/lib/node_modules/yarn/bin/yarn.js
/usr/local/lib
└── yarn@1.3.2

{11:01}~/Project/mydelivery ➪ which yarn
/usr/local/bin/yarn
{11:01}~/Project/mydelivery ➪ yarn
yarn install v1.3.2
info No lockfile found.
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
warning " > request-promise@4.2.2" has unmet peer dependency "request@^2.34".
warning "request-promise > request-promise-core@1.1.1" has unmet peer dependency "request@^2.34".
warning " > babel-loader@7.1.2" has unmet peer dependency "babel-core@6 || 7 || ^7.0.0-alpha || ^7.0.0-beta || ^7.0.0-rc".
warning " > babel-loader@7.1.2" has unmet peer dependency "webpack@2 || 3".
[4/4] 📃  Building fresh packages...
success Saved lockfile.
✨  Done in 37.67s.
{11:02}~/Project/mydelivery ➪
```

실습환경 설정 - babel

- es6는 스펙이지 실체가 아님
  - 언어에 실체가 어딨어요
  - 구동해주는 엔진들 (브라우저 혹은 node같은)이 언어의 스펙을 지원해줘야 사용이 가능하다.
- node의 현재 상황 - node.green 사이트 참고
  - 참고: 여기 import / export 지원 여부는 안나오네요



import / export

- es6부터 도입된 모듈 syntax (ESM)
- 기존 node에서는 require 사용 (commonJS module)
- 서로간의 차이점이 존재

require	import
Dynamic evaluation	Static evaluation
Throws error at runtime	Throws error while parsing
Non lexical	Lexical

http://voidcanvas.com/import-vs-require/

- node 8.9 부터 실험적으로 import/export를 네이티브 지원
- 그러나 우리는 babel을 써서 import/export를 트랜스파일 할것

babel 트랜스파일

- node는 아직 es6를 완전히 지원하지 못한다.
- es6로 프로그래밍을 하면 좋은게 많다
  - import / export (정적분석이 조금은 가능)
  - async / await
  - spread operator (…args 같은거)
- 그래서 babel을 통해 es6 소스코드를 node가 적당히 이해할 수 있는 수준으로 트랜스파일하여 돌려줌 (babel환경을 세팅하기 나름)
- babel을 standalone으로 쓰면 소스코드를 transpile하여 다른 js 소스코드를 만듬
- 그러나 개발하는 중에는 런타임으로 얹어서 사용

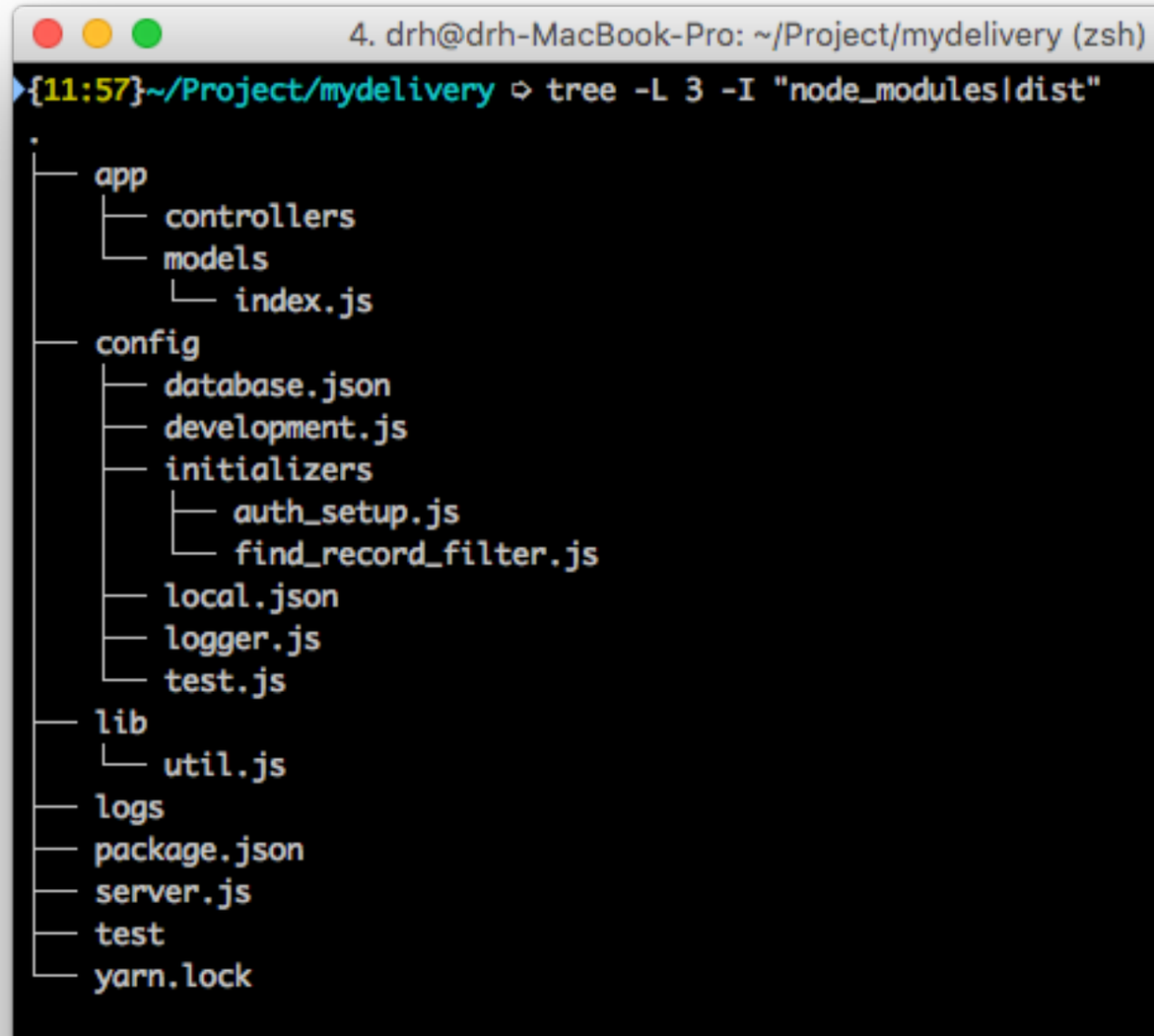
실습환경 설정 - .babelrc

```
{
  "presets": [["env",
    {
      "targets": {
        "node": "current"
      }
    }], "stage-2"],
    "plugins": [],
    "sourceMaps": "both"
}
```

- preset은
  - 트랜스파일할 규칙을 설정해줌. (저도 잘 몰라요)
  - babel은 env를 쓰라고 추천하고, target에 node: current를 집어넣으면 현재 노드 버전에 맞춰주는듯
- sourceMap은
  - 트랜스파일한 코드와 원래 코드는 다르기 때문에 실제 엔진에서 수행하다가 에러가 난 위치와 실제 코드상의 위치는 다르기 마련.
  - 그것에 대한 매핑을 유지하는 옵션

디렉토리 구조

- 우리 서버의 디렉토리 구조는 다음과 같음
- app - 서버 애플리케이션 구동 파일들
  - models - 모델 정의들
  - controllers - 컨트롤러 정의들 (라우트 핸들러 함수 모음집)
- config - 환경설정 및 초기 로딩 파일 모음
  - initializers - 인증 혹은 전역사용 필터 정의
- lib - 사용자 정의 라이브러리 파일 모음
- logs - 구동시 생성되는 로그파일
- test - 테스트 파일 디렉토리



환경 파일 설정

- config 라는 node 모듈을 사용

- DB 접속을 위한 환경
  - config/{database, mongo, redis}.json
- 환경에 따른 설정
  - config/{development, test, production}.js
- 로컬 셋업을 위한 설정
  - config/local.json

- 이렇게 세팅해두면, 실제 애플리케이션 코드에서 config.xxx 로 미리 세팅된 환경에 접근할 수 있음

```json
{
  "development": {
    "dialect": "mysql",
    "host": "localhost",
    "database": "drh_mydelivery_development",
    "username": "drh",
    "password": "gkswoghk"
  },
  "test": {
    "username": "drh",
    "password": "gkswoghk",
    "database": "drh_mydelivery_test",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "production": {
    "username": "drh",
    "password": "gkswoghk",
    "database": "drh_mydelivery_production",
    "host": "127.0.0.1",
    "dialect": "mysql"
  }
}
```

config/database.json
MySQL 접속 설정

```json
{
  "development": {
    "host": "localhost",
    "port": 6379,
    "pass": "gkswoghk",
    "auth": "gkswoghk",
    "ttl": 86400
  },
  "test": {
    "host": "localhost",
    "port": 6379,
    "pass": "gkswoghk",
    "auth": "gkswoghk",
    "ttl": 86400
  },
  "production": {
    "host": "localhost",
    "port": 6379,
    "pass": "gkswoghk",
    "auth": "gkswoghk",
    "ttl": 86400
  }
}
```

config/redis.json
Redis 접속 설정

```json
{
  "winston": {
    "transports": [{
      "type": "DailyRotateFile",
      "level": "debug",
      "filename": "./logs/all.log",
      "handleExceptions": true,
      "json": false,
      "maxsize": 104857600,
      "colorize": false
    }]
  },
  "jwt": {
    "user_secret": "mydeliveryapiserver"
  }
}
```

config/local.json
런타임에 로딩

```js
const database = require('./database.json');
const redis = require('./redis.json');

module.exports = {
  database: database["development"],
  redis: redis["development"],
};
```

config/development.js
개발환경일때 로딩됨

entry point - server.js

- 서버의 엔트리 포인트인 server.js

- 서버 구동에 필요한 것들 셋업
  - 비지니스 로직부분
    - DB접속, ORM 설정, Model 로드, 초기화 등

- HTTP 인터페이스 부분
  - route 셋업, 포트개방
  - body parser 설정, 미들웨어 설정

```
import db from './app/models';
import logger from './config/logger';
import express from 'express';
import cookieParser from 'cookie-parser';
import bodyparser from 'body-parser';
import config from 'config';
import glob from 'glob';
import cors from 'cors';
import morgan from 'morgan';
import routerSetup from './config/routes';

const router = express.Router();
const _ = require('lodash');

export default (async function() {
  /*
   * Load database
   */

  // TODO implement

  /*
   * express js app setup
   */
  const app = express();

  // middleware use setup
  // TODO implement

  /*
   * Initializer setup
   */
  const initializers = glob.sync('./config/initializers/**/*.js');
  for (const initializer of initializers) {
    logger.info(`loading ${initializer}...`);
    require(initializer)(app, db);
  }

  /*
   * request logger
   */

  // TODO implement

  /*
   * Route install
   */
```

ed

```
logger.verbose('Route setup');
routerSetup(router);
app.use(router);

// install error sender
// TODO implement

// install default error logger
app.use((err, req, res, next) => {
  logger.error(err.stack);

  // for out-going response, remove error detail. just send internal server error
  res.status(500).send('internal server error');
});

/*
 * port & interface setup
 */
const port = process.env.PORT || config.host.port || 9000;
const intfc = process.env.INTERFACE || '127.0.0.1';

// TODO listen to the port

return app;
})();
```

서비스 내의 엔티티들

- 이 서비스에서 돌아다니는 엔티티들은 (괄호 안은 테이블 이름)
  - 가정: 서비스하는 회사는 한개라고 칩시다.
  - User (users) - 사용자, 주문 넣는 주체
    - hasMany Order
  - Order (orders) - 사용자가 넣은 주문
    - belongsTo User
    - hasMany OrderEntry
  - OrderEntry (order_entries) - 주문 상세내역
    - belongsTo Order
    - belongsTo Product
    - 어떤 product를 몇개를 주문했는지 저장
  - Product (products) - 판매하는 상품 (메뉴)
    - 가격, 타이틀, 설명 등..

첫번째 엔티티 - User

- 사용자 정보
  - 전화번호를 로그인 아이디로 쓰자 (귀찮)
    - 국제 서비스 안할거니까 전화번호 형식은 국내형식만.
    - 전화번호 문자인증 하면 좋겠네요 (스킵)
    - 비밀번호가 있어서 로그인 가능하게
    - 닉네임 셋업가능

- 인증 주체
  - 전화번호/비밀번호 조합으로 로그인 가능
  - 인증 후 토큰 발급해야함

```
import bcrypt from 'bcrypt';
import _ from 'lodash';
import {Model, DataTypes} from 'sequelize';

const mobileregex = /^01\d{8,9}$/;
export default class User extends Model {
  static fields = {
    // TODO implement
    deleted_at: {
      type: DataTypes.DATE,
      allowNull: false,
      defaultValue: new Date(0)
    },
  };

  static options = {
    freezeTableName: true,
    tableName: 'users',
    underscored: true,
    paranoid: true, // mark deleted_at

    setterMethods: {
      password(password) {
        // TODO implement
      }
    },
  };

setInfo(attributes, opts={}) {
  if (!opts.transaction) {
    return this.sequelize.transaction((tx) => {
      opts.transaction = tx;
      return this.setInfo(attributes, opts);
    });
  }
  // if parameter sets explicit NULL,
  // it means it should be deleted, undefined means retain old value
  const whitelist = ['nickname'];

  const filtered = _.reduce(whitelist, (acc, key) => {
    if (attributes[key] === undefined) {
      return acc;
    }

    acc[key] = attributes[key];
    return acc;
  }, {});
```

```
    // TODO implement

  };

  dropOut(opts={}) {
    if (!opts.transaction) {
      return this.sequelize.transaction((tx) => {
        opts.transaction = tx;
        return this.dropOut(opts);
      });
    }

    // TODO implement
  }

  async comparePassword(password) {
    // TODO implement
  }

  myOrders(filter) {
    // TODO filter using filter object
    // TODO implement
  }

  static associate(models) {
    // TODO implement
    // User.hasMany(models.Order);
    // ...
    // ...
  }

  // class method definitions
  static signup({phone, password}, opts = {}) { // use phone as id
    if (!opts.transaction) {
      return this.sequelize.transaction((tx) => {
        opts.transaction = tx;
        return User.signup({phone, password}, opts);
      })
    }

    // TODO implement
  }

  static async checkPhone(phone) {
    const user = await User.find({where: {phone}});
    return !user;
  }
}
```

간단한 테스트를 해보자

4. mysql -h localhost -uroot -pgkswoghk drh_mydelivery_development (mysql)

```
{12:59}~/Project/mydelivery ➪ cat load_sequelize.js
import os from 'os';
import db from './app/models';
import Promise from 'bluebird';
import repl from 'repl';

global["Models"] = global["db"] = db;
global["Promise"] = Promise;

repl.start('> ');
{12:59}~/Project/mydelivery ➪ which sc
sc: aliased to node -r 'regenerator-runtime/runtime' -r babel-register -i load_sequelize.js
{12:59}~/Project/mydelivery ➪ sc
sequelize deprecated String based operators are now deprecated. Please use Symbol based operators for better security, read more at
 http://docs.sequelizejs.com/manual/tutorial/querying.html#operators node_modules/sequelize/lib/sequelize.js:236:13
2017-11-15T04:00:04.019Z - info: importing users.js...
2017-11-15T04:00:04.063Z - info: associating User
> db.User.signup
[Function: signup]
> db.User.signup({phone: '01031808148', password: 'gkswoghk'}).then(u => console.log(u.id, u.phone))
Promise {
  _bitField: 0,
  _fulfillmentHandler0: undefined,
  _rejectionHandler0: undefined,
  _promise0: undefined,
  _receiver0: undefined }
> 1 '01031808148'

>
(To exit, press ^C again or type .exit)
>
^C
```

db에 제대로 들어갔을까

4. mysql -h localhost -uroot -pgkswoghk drh_mydelivery_development (mysql)

```
{13:00}~/Project/mydelivery ➪ mysql -h localhost -uroot -pgkswoghk drh_mydelivery_development
mysql: [Warning] Using a password on the command line interface can be insecure.
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 336
Server version: 5.7.17 Homebrew

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from users;
+----+------------+--------------------------------------------------------------+------------+----------+-------+---------------------
------+---------------------+---------------------+
| id | phone      | password_hashed                                              | auth_count | nickname | agreed | deleted_at
     | created_at          | updated_at          |
+----+------------+--------------------------------------------------------------+------------+----------+-------+---------------------
------+---------------------+---------------------+
|  1 | 01031808148 | $2a$10$LVyVdJUiVzz4TdH2zGOVneo017J8VFB7w31EVAj7VwH9So3Nsjri2 |          0 | NULL     |      1 | 1970-01-01 00:
00:00 | 2017-11-15 04:00:27 | 2017-11-15 04:00:27 |
+----+------------+--------------------------------------------------------------+------------+----------+-------+---------------------
------+---------------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

OAuth2 + JWT 인증 셋업

- 우리 서비스의 여러 리소스들은 인증이 없으면 의미가 없음
  - 주문을 '누가' 넣었는지, 주문을 넣는 '권한이 있는지'
- server는 HTTP request를 받아서 동작을 할 것이므로, HTTP request를 인증하여
  - 어떤 사용자인지(Authentication),
  - 이 동작을 할수 있는지(Authorization)를 판단하자
- 인증방식
  - 매 HTTP request마다 서버가 발급한 토큰을 심어 날려서 이 request를 인증해달라고 서버에 요청
  - OAuth2 스펙 중 Bearer 인증방식사용, 토큰은 JWT 사용
  - OAuth2 exchange 스키마 중 password 및 refresh_token exchange를 사용

OAuth2 서버 및 passport + JWT 설정

```
import passport from 'passport';
import { Strategy as LocalStrategy } from 'passport-local';
import { BearerStrategy } from 'passport-http-bearer';
import bcrypt from 'bcrypt';
import config from 'config';
import jwt from 'jsonwebtoken';
import logger from '../logger';
import { Strategy as JwtStrategy, ExtractJwt } from 'passport-jwt';
import oauth2orize from 'oauth2orize';
import redis from 'redis';
import Promise from 'bluebird';
import _ from 'lodash';
import myutil from '../../lib/util';
import { BasicStrategy } from 'passport-http';
import User from "../../app/models/users";

Promise.promisifyAll(redis.RedisClient.prototype);
const redisClient = redis.createClient({...config.redis, password: config.redis.pass});

export default (app) => {
  app.use(passport.initialize());

  const userOAuthServer = oauth2orize.createServer();

  global.OAuthServers = {
    user: userOAuthServer,
  };

  /* functor */
  const tokenIssuer = ({model, getData, jwtSecret}) =>
    async (entity, done) => {
      const payload = {
        sub: entity.id,
        cnt: entity.auth_count,
      };

      const refreshToken = '';
      const accessToken = '';
      // TODO set refresh token and access token, save refresh token to redis
```

```
      console.log(`tokenIssuer refreshToken=${refreshToken}, namespace=${namespace}`);

      return done(null, accessToken, refreshToken, data); // null for error (follows node
style callback)
    };

  /* user OAuth setup */
  const opts = {
    model: User,
    jwtSecret: config.jwt.user_secret,
    getData: (user) => ({phone: user.phone, id: user.id}),
  };

  userOAuthServer.tokenIssuer = tokenIssuer(opts);
  userOAuthServer.exchange(oauth2orize.exchange.password(async (client, username,
password, done) => {
    // TODO user validation and password check
    const issueToken = tokenIssuer(opts);
    await issueToken(user, done.bind(this));
  }));

  userOAuthServer.exchange(oauth2orize.exchange.refreshToken(async (client, refreshToken,
done) => {
    // TODO check refresh token and user record

    // remove old refresh token
    await redisClient.hdelAsync(namespace, str);

    const issueToken = tokenIssuer(opts);
    await issueToken(user, done.bind(this));
  }));

  passport.use('user-jwt', new JwtStrategy({
    secretOrKey: opts.jwtSecret,
    jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken()
  }, async (jwtPayload, done) => {
    const entity = await User.find({where: {id: jwtPayload.sub}});
    if (!entity) {
      return done(null, false);
    }

    if (entity.auth_count !== jwtPayload.cnt) {
      return done(null, false);
    }

    return done(null, entity);
  }));
```

```
 passport.use('client-basic',
 new BasicStrategy({session: false}, (req, userid, password, done) => {
    // TODO validate client
    done(null, {user: {id: userid}});
  }));
};
```

인증 라우터 설정

- 토큰을 발급하려면 POST /auth/token 에
  - grant_type='password', username, password 또는
  - grant_type='refresh_token', refresh_token 을 보냄
  - 인자이름이 username인것은 OAuth2 표준
    - OAuth2orize 라이브러리가 강제

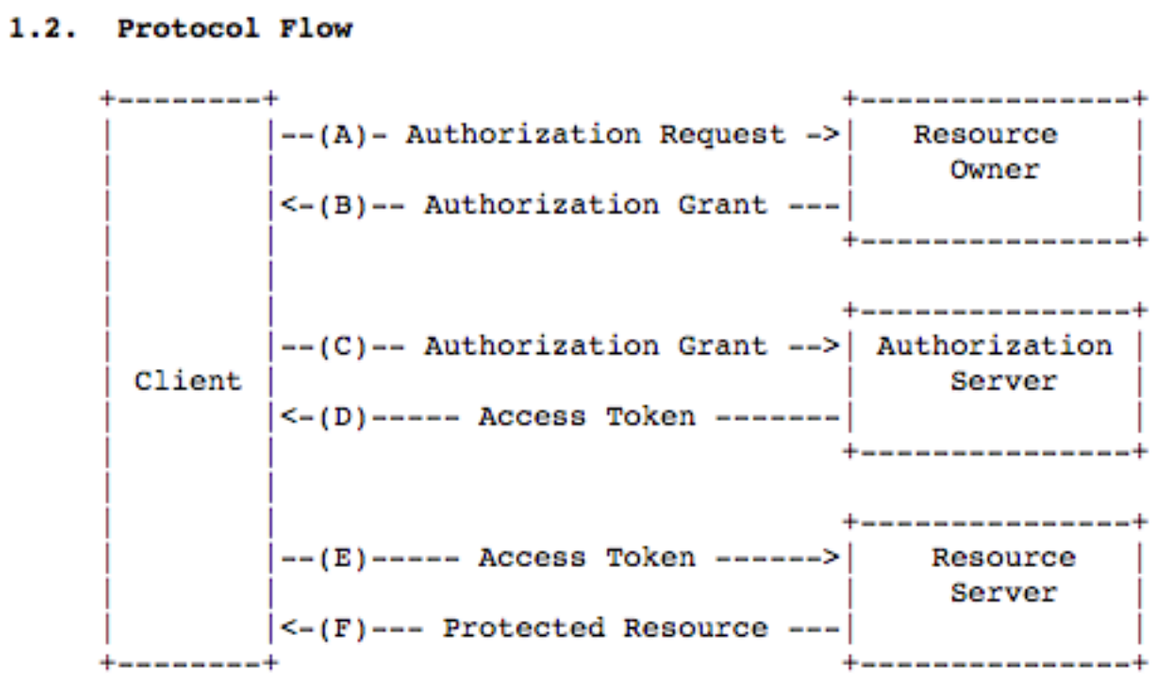


Figure 1: Abstract Protocol Flow

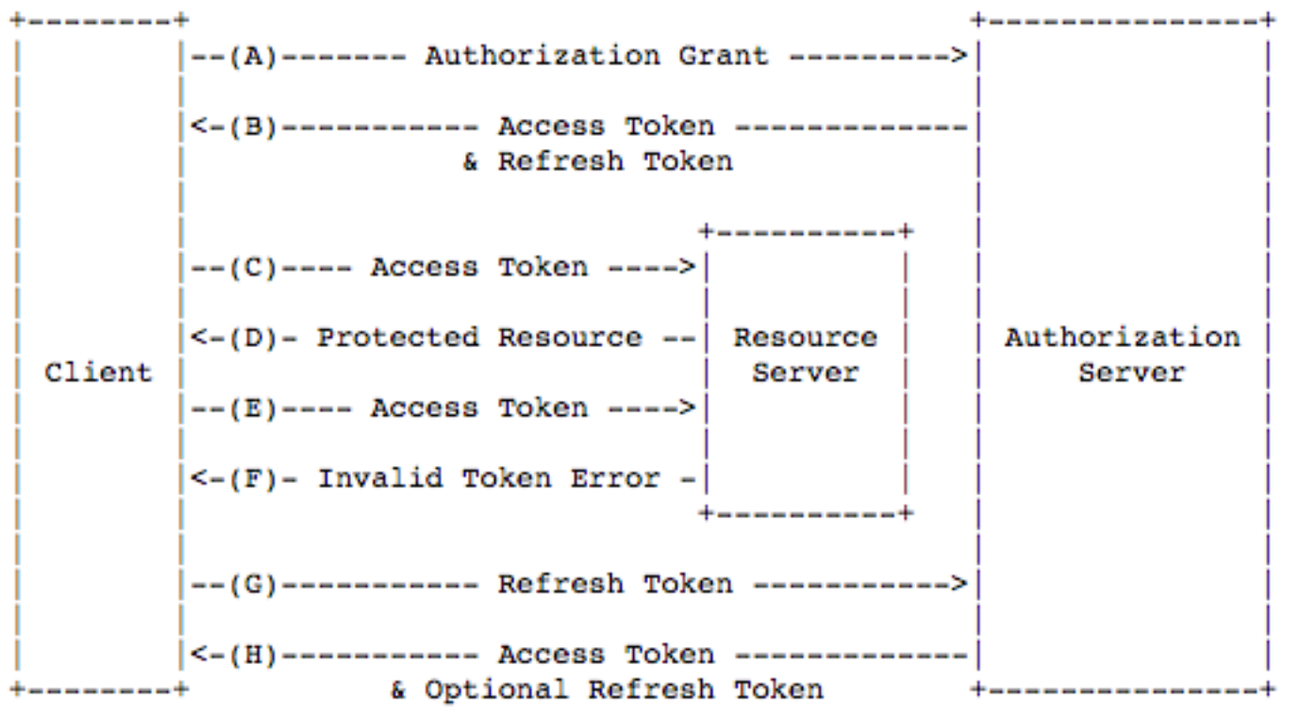


Figure 2: Refreshing an Expired Access Token

https://tools.ietf.org/html/rfc6749

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=johndoe&password=A3ddj3w
```

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=tGzv3JOkF0XG5Qx2TlKWIA
```

config/routes.js

```
import path from 'path';
import logger from './logger';
import wrapAsync from 'express-wrap-async';
import glob from 'glob';

export default (router) => {
  const controllers = {};
  // router gen function
  const R = (str) => {
    const [c, a] = str.split('#');
    const controller = controllers[c];
    const f = controller ? controller[a] : (req, res, next) => res.status(444).send(`controller \`${c}\` doesn't exist!`);
    return f && wrapAsync(f) || ((req, res, next) => res.status(442).send(`action \`${str}\` isn't implemented yet!`));
  };

  const RR = (prefix, model) => {
    router.get(`/${prefix}${model}`, R(`${prefix}${model}#index`));
    router.get(`/${prefix}${model}/:id`, R(`${prefix}${model}#show`));
    router.post(`/${prefix}${model}`, R(`${prefix}${model}#create`));
    router.post(`/${prefix}${model}/:id/update`, R(`${prefix}${model}#update`));
    router.put(`/${prefix}${model}/:id`, R(`${prefix}${model}#update`));
    router.post(`/${prefix}${model}/:id/destroy`, R(`${prefix}${model}#destroy`));
    router.delete(`/${prefix}${model}/:id`, R(`${prefix}${model}#destroy`));
  };

  // load all controllers
  const files = glob.sync(`${__dirname}/../app/controllers/**/*.js`);
  files.forEach((file) => {
    const fileName = path.relative(`${__dirname}/../app/controllers/`, file);
    const controllerName = fileName.match(/(.*)_controller/)[1];
    logger.verbose(`controllerName=${controllerName}`);
    return controllers[controllerName.replace(new RegExp('${path.sep}', 'g'), '/')] = require(file).default;
  });

  // define routes

  router.get('/', (req, res, next) => res.send('Hello, World!'));

  // TODO implement
  // 인증 라우트 연결


};
```

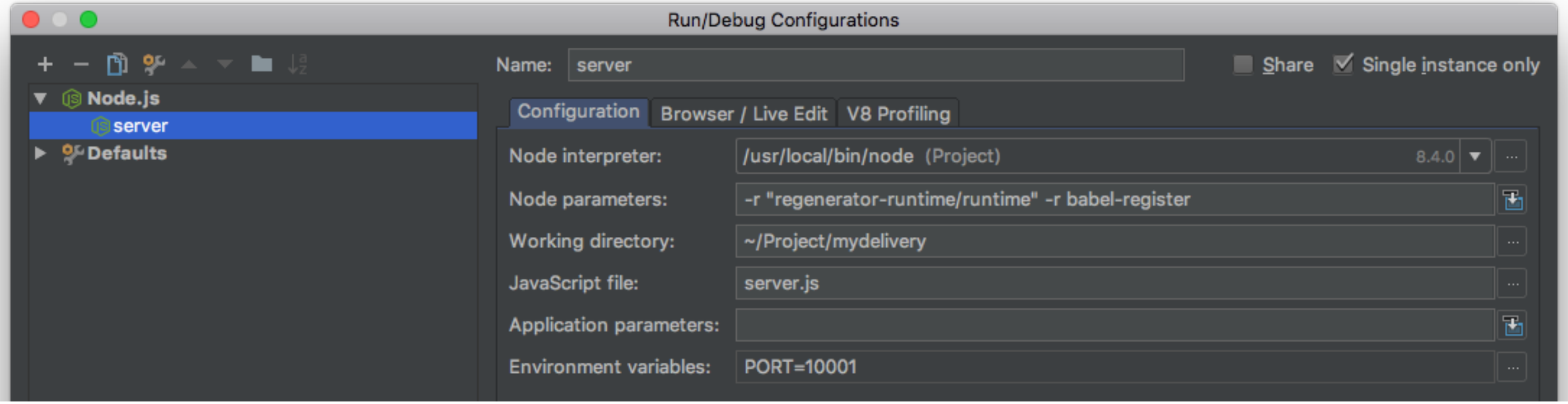
app/controllers/auth_controller.js

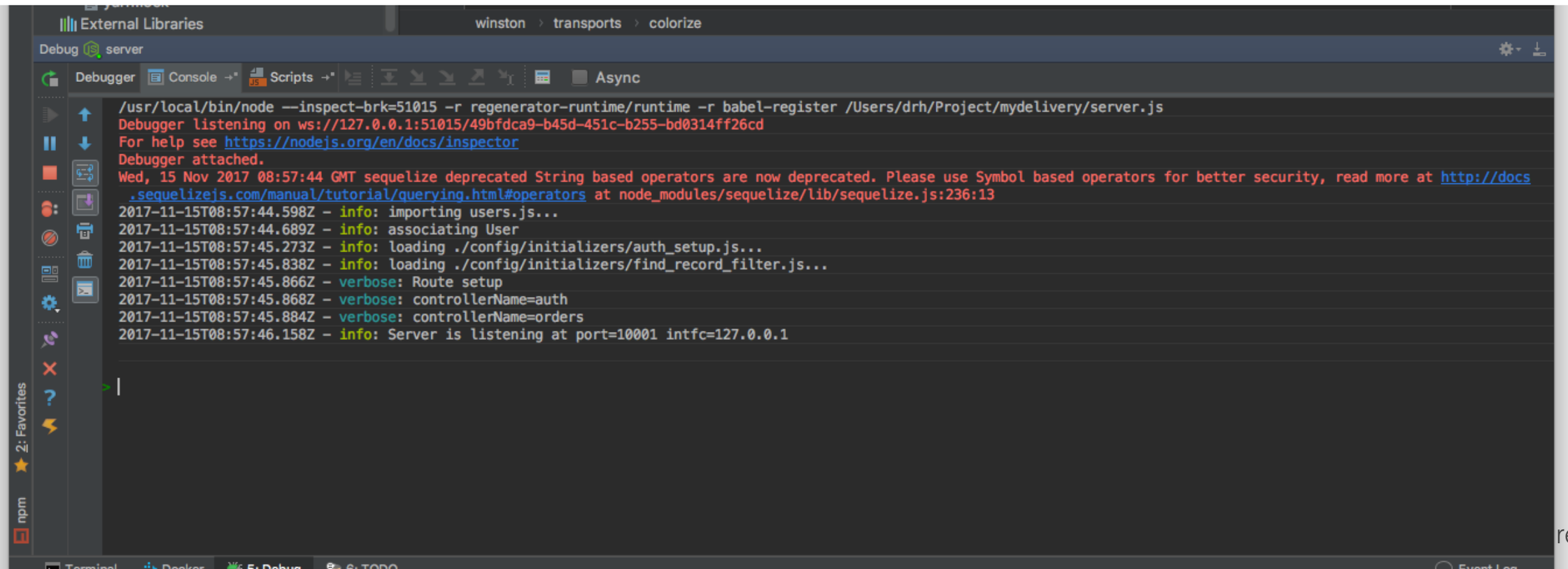
```
import passport from 'passport';
const userOAuthServer = global.OAuthServers.user;

const controller = {
  token: [
    passport.authenticate(['client-basic'], {session: false}),
    userOAuthServer.token(),
    userOAuthServer.errorHandler({mode: 'direct'}),
  ],
};

export default controller;
```

- oauth2orize는 HTTP Basic을 이용한 client auth를 강제함
- oauth2orize가 token 이라는 미들웨어 제공
  - oauth2orize는 oauth2 스펙에 따라 client auth를 하고 들어와야함

첫 서버 기동

- IDE를 사용하여 기동해보자





토큰을 얻어오는지 테스트

4. drh@drh-MacBook-Pro: ~/Project/mydelivery (zsh) — 116×17

```
{15:08}~/Project/mydelivery ➪ curl -X POST -d grant_type='password' -d username='01031808148' -d password='gkswoghk'
 testclient:clientpassword@localhost:10001/auth/token | jq .
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   392  100   334  100    58   3403    591 --:--:-- --:--:-- --:--:--  3408
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOjEsImNudCI6MCwiaWF0IjoxNTEwNzI2MTQ1LCJleHAiOjE1MTA3
Mjc5NDV9.FC0nUAeHCe_Ln0ccVolQBLVhpxsVMCfPE_cKCeKdRgE",
  "refresh_token": "96e0123ef2b6dc3c7ee6ee22404a06f0a83b4acdead3c4b3de10e814f96316e3e002457bc529f6825b28a7123b1ee753
",
  "phone": "01031808148",
  "id": 1,
  "token_type": "Bearer"
}
{15:09}~/Project/mydelivery ➪
```

4. drh@drh-MacBook-Pro: ~/Project/mydelivery (zsh)

```
{15:09}~/Project/mydelivery ➪ curl -X POST -d grant_type='refresh_token' -d refresh_token='96e0123ef2b6dc3c7ee6ee224
04a06f0a83b4acdead3c4b3de10e814f96316e3e002457bc529f6825b28a7123b1ee753' testclient:clientpassword@localhost:10001/a
uth/token | jq .
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   469  100   334  100   135  10176   4113 --:--:-- --:--:-- --:--:-- 10437
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOjEsImNudCI6MCwiaWF0IjoxNTEwNzI2MjU3LCJleHAiOjE1MTA3
MjgwNTd9.mAG8_ADAMn-uZBCTW4YpG0J2OPnU2ivv5b_4aj5954Q",
  "refresh_token": "be0ec9b7e49a7dd1a784c59669cc67f82576d726c0266094e7ee615fb9eaf28b624483fd2817fbdf17b67ad40f51b76f
",
  "phone": "01031808148",
  "id": 1,
  "token_type": "Bearer"
}
{15:10}~/Project/mydelivery ➪
```

인증을 요구하는 endpoint 작성

- 각 컨트롤러의 액션 중 특정 액션에만 인증을 요구하고 싶음
  - ex) 내가 넣은 주문 보기, 내 정보 확인 및 수정 등..
- express의 미들웨어 활용
  - 내가 만든 컨트롤러 로직에 들어가기 전에 조건을 만족하는지 보는 인증 미들웨어를 삽입
- passport가 제공해주는 미들웨어 (authenticate)
  - passport.authenticate(arguments..)
  - 인자로 넘어온 strategy를 이용하여 인증을 수행
    - strategy는 셋업할때 passport.use로 세팅했던 것들 중 하나
    - 인증이 실패하면 바로 Unauthorized를 리턴함

order list route 작성

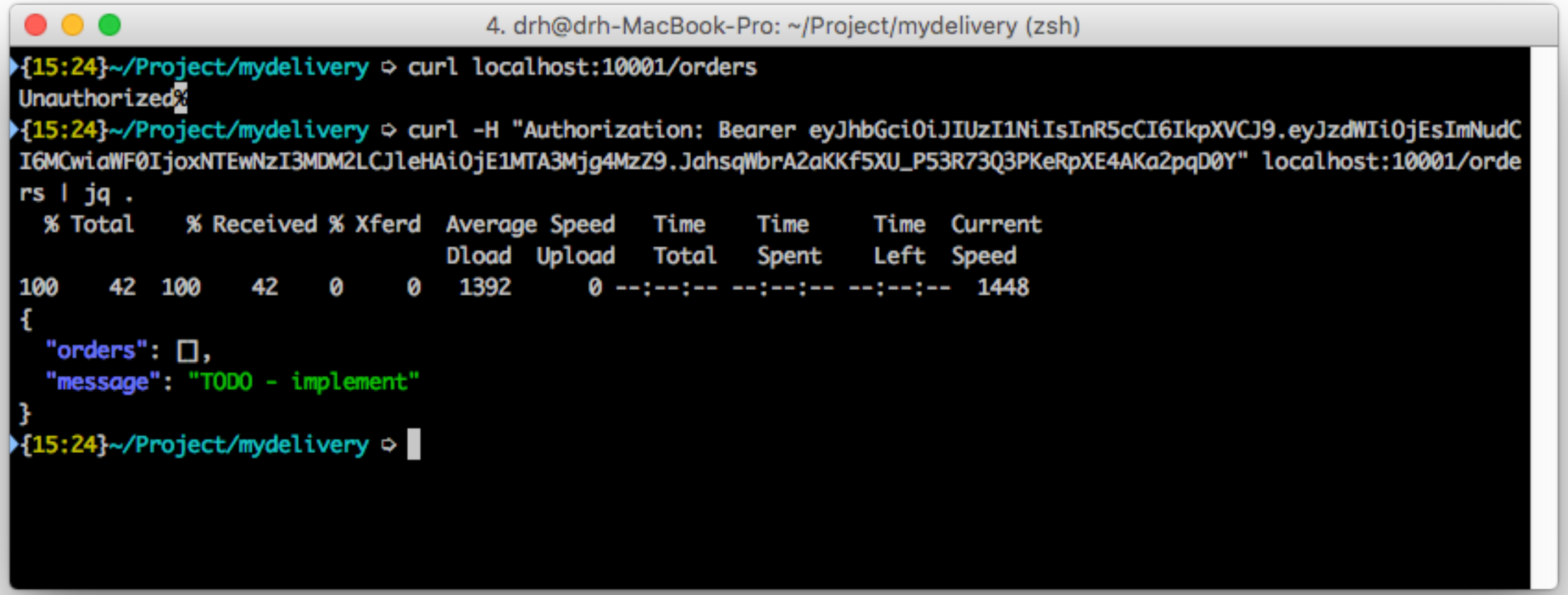
- GET /orders
  - 내가 넣은 주문들 반환
- GET /orders/:id
  - 특정 id를 가지고 있는 주문 정보 반환
  - 소유권 확인 필요
- POST /orders
  - 새 주문 집어넣음
- POST /orders/:id/pay
  - 특정 주문에 대한 결제
  - 소유권 확인 필요, 상태체크 필요 (중복계산, 완료된 주문 결제 등)
- POST /orders/:id/cancel
  - 주문 취소
  - 소유권 확인 필요, 상태체크 필요 (이미 취소된 주문 등)

orders_controller.js, config/routes.js

```
import passport from 'passport';

const controller = {
  list: [passport.authenticate('user-jwt', {session: false}), (req, res) => {
    res.send({orders: [], message: 'TODO - implement'});
  }],
};

export default controller;
```

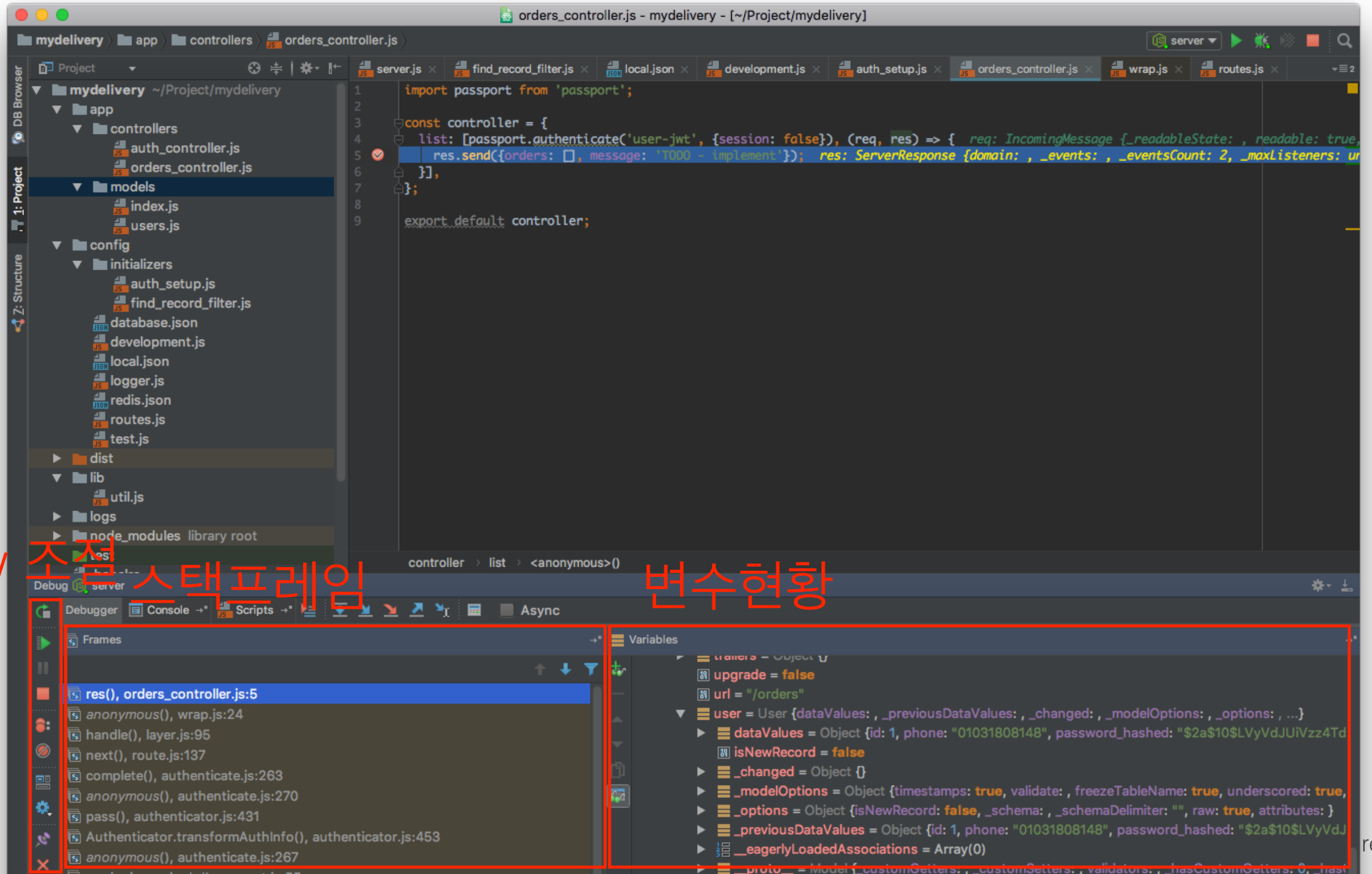
```
// define routes

router.get('/', (req, res, next) => res.send('Hello, World!'));


router.post('/auth/token', R('auth#token'));
router.get('/orders', R('orders#list'));
```

인증 테스트



디버거

- IDE를 쓰는 주된 이유중 하나
- WebStorm에 디버거가 내장되어 있음 / breakpoint 지원



flow 조절 스택프레임 변수현황

다음 시간에 할 것들

- 주문 관리에 관한 비지니스 로직 작성
  - 주요 로직을 모델 메소드로 빼는 연습
  - 히스토리 데이터의 모델링에 대한 제언
- admin(관리자) 라우트의 필요성과 구현
- 모델 및 컨트롤러 테스트 로직 만들기
- 친절한 에러 메시지에 대해
- 리팩토링

Q&A 시간