FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

LẬP TRÌNH TYPESCRIPT

BUILD TOOLS

THỰC HỌC – THỰC NGHIỆP

http://www.poly.edu.vn

Mục tiêu

- Sử dụng thư viện javascript với typescript
- lodash
- class transformer
- class validator

Mục tiêu

- Cài đặt express
- Làm việc với Types
- Middleware
- Route
- Làm việc với Controllers & Parsing Request Bodies

PHẦN 1

THƯ VIỆN JAVASCRIPT

- Lodash: một thư viện Javascript mạnh dùng để xử lý array, object, function, collection,...
  - Cải thiện hiệu suất
  - Code đơn giản
  - Dễ nhìn
  - Tích hợp với hầu hết các công nghệ Javascript hiện tại như: NodeJs, ReactJs,...

CÀI ĐẶT

- Link: https://lodash.com

- Cài đặt bằng npm

```
$ npm i -g npm
$ npm i --save lodash
```

- Cài đặt type lodash

```
npm install --save @types/lodash
```

GLOBAL VARIABLE JAVASCRIPT

- index.html

```
<!DOCTYPE html>
    <head><title>Typescript and Webpack</title></head>
    <body>
        <script src="dist/index.bundle.js" defer></script>
        <script src="dist/app.bundle.js" defer></script>
        <script>
            var GLOBAL = 'LODASH';
        </script>
    </body>
</html>
```

GLOBAL VARIABLE JAVASCRIPT

- app.ts

```
import _ from 'lodash';

console.log(_.shuffle([1,2,3]));

declare var GLOBAL: any;
console.log(GLOBAL);
GLOBAL= 'THIS IS SET';
console.log(GLOBAL);
```

CLASS-TRANSFORMER

- Chuyển đổi một đối tượng thành instance của class và ngược lại

- Cài đặt

```
npm install class-transformer --save
```

- Cài đặt reflect metadata

```
npm install reflect-metadata --save
```

CLASS-TRANSFORMER

- Ví dụ: product.model.ts

```
export class Product {
  title: string;
  price: number;

  constructor(t: string, p: number) {
    this.title = t;
    this.price = p;
  }

  getInformation() {
    return [this.title, `$${this.price}`];
  }
}
```

CLASS-TRANSFORMER

- Ví dụ: app.ts (sử dụng class-transformer)

```
import 'reflect-metadata';
import { plainToClass } from 'class-transformer';

import { Product } from './product.model';

const products = [
  { title: 'A Carpet', price: 29.99 },
  { title: 'A Book', price: 10.99 }
];

const loadedProducts = plainToClass(Product, products);

for (const prod of loadedProducts) {
  console.log(prod.getInformation());
}
```

CLASS-VALIDATOR

- Cho phép sử dung validator và non-validator dựa vào validation

- Cài đặt

```
npm install class-validator --save
```

Ví dụ: product.model.ts

```
import { IsNotEmpty, IsNumber, IsPositive } from 'class-validator';

export class Product {
  @IsNotEmpty()
  title: string;
  @IsNumber()
  @IsPositive()
  price: number;

  constructor(t: string, p: number) {
    this.title = t;
    this.price = p;
  }

  getInformation() {
    return [this.title, `$${this.price}`];
  }
}
```

Ví dụ: app.ts (sử dụng class-validator)

```
import 'reflect-metadata';
import { plainToClass } from 'class-transformer';
import { validate } from 'class-validator';

import { Product } from './product.model';

const products = [
  { title: 'A Carpet', price: 29.99 },
  { title: 'A Book', price: 10.99 }
];

const newProd = new Product('', -5.99);
validate(newProd).then(errors => {
  if (errors.length > 0) {
    console.log('VALIDATION ERRORS!');
    console.log(errors);
  } else {
    console.log(newProd.getInformation());
  }
});
```


demo

PHẦN 2

- Là framework javascript
- Dùng xây dựng web ứng dụng
- Không được tích hợp sẵn

- Cài đặt

```
$ npm install express --save
```

- Cài đặt type express: là 1 gói định nghĩa loại Express

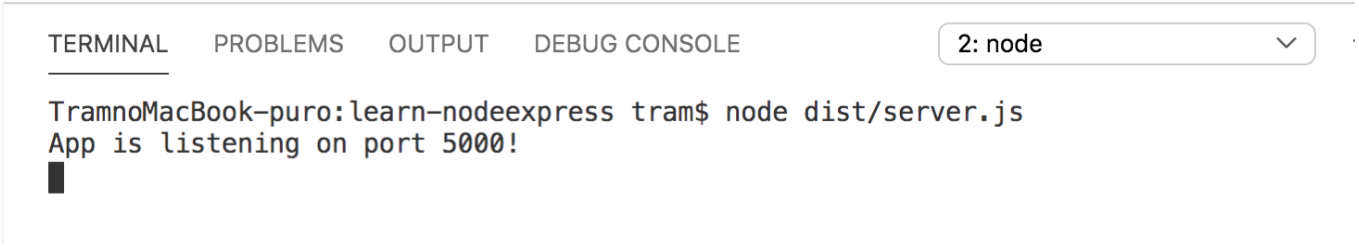
```
npm install --save @types/express
```

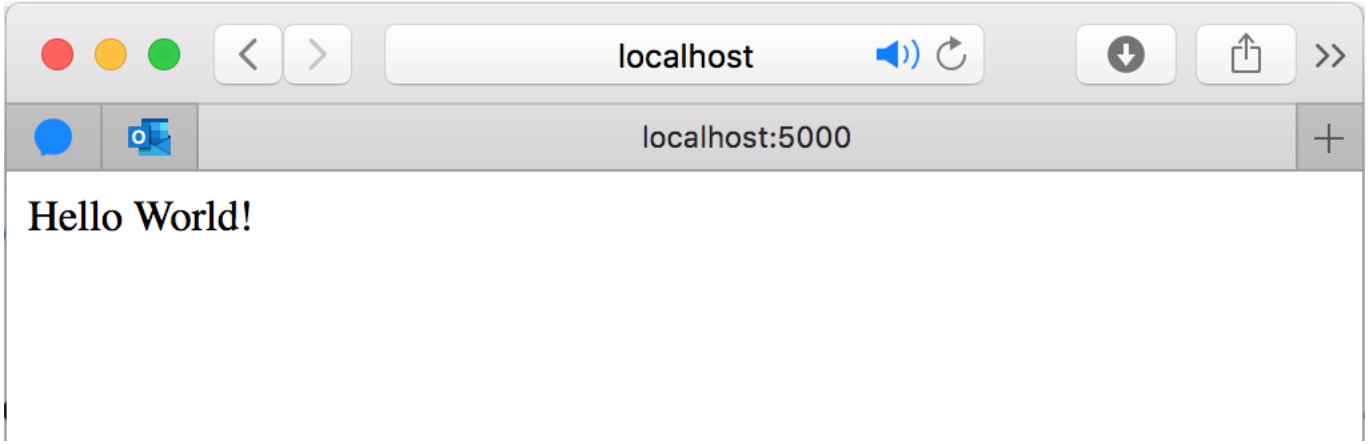
- **request** và **response** đại diện cho thuộc tính HTTP request và response.
- hàm **response.send**: response đến client
- **Listen:** ứng dụng kết nối trên cổng được chỉ định
  - *App.listen(5000)* : có nghĩa là http://localhost:5000

Ví dụ

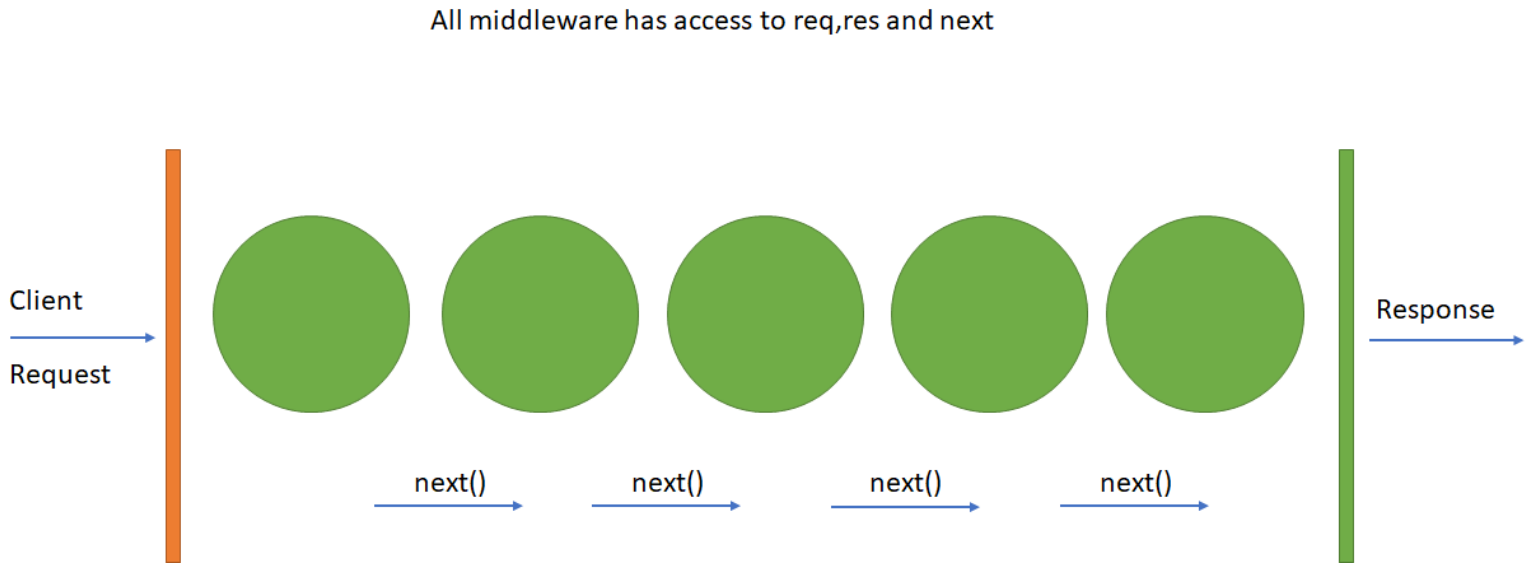
```
import express from 'express';


const app = express();

app.get('', (request, response) => {
    response.send('Hello World!');
});

app.listen(5000, function (){
    console.log('App is listening on port 5000!');
});
```

Kết quả


TERMINAL
PROBLEMS
OUTPUT
DEBUG CONSOLE
2: node
TramnoMacBook-puro:learn-nodeexpress tram$ node dist/server.js
App is listening on port 5000!

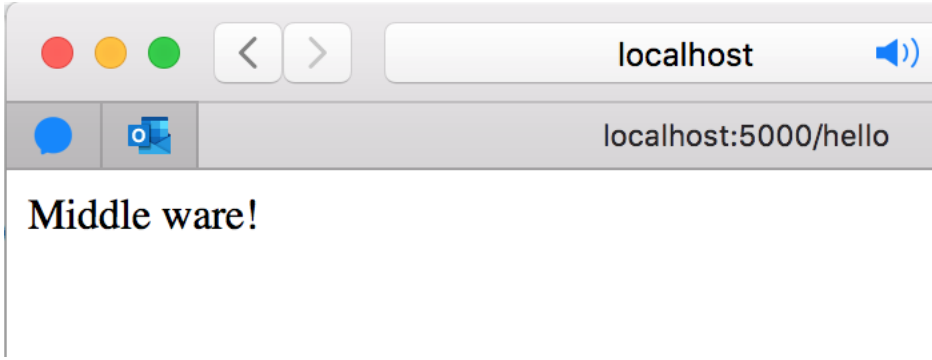

localhost
localhost:5000
Hello World!

Middle ware


All middleware has access to req,res and next
Client
Request
Response
next()
next()
next()
next()

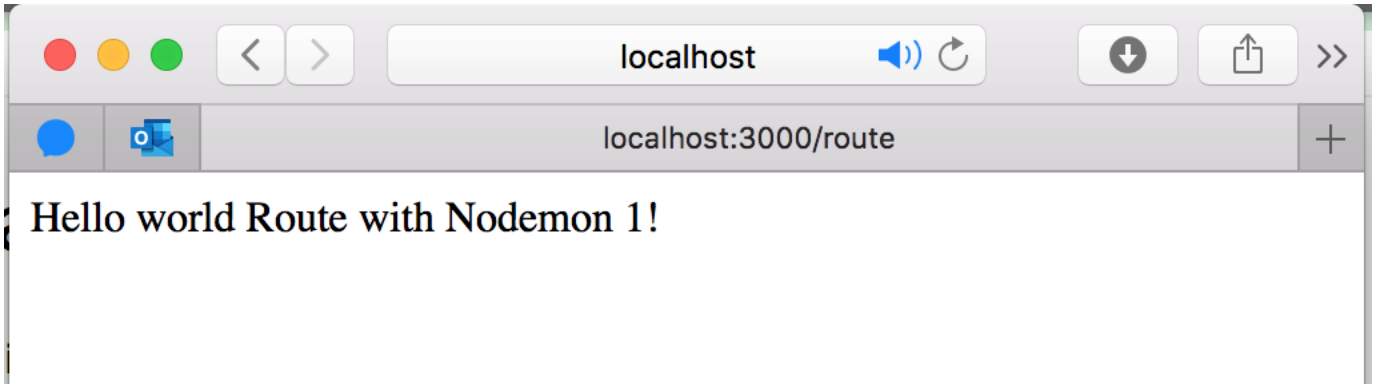
Ví dụ

```
import express from 'express';


const app = express();
//MIDDLE WARE
function loggerMiddleware(request: express.Request, response: express.Response, next:any) {
    console.log(`${request.method} ${request.path}`);
    next();
}

app.use(loggerMiddleware);

app.get('/hello', (request, response) => {
    response.send('Middle ware!');
});

app.listen(5000, function (){
    console.log('App is listening on port 5000!');
});
```

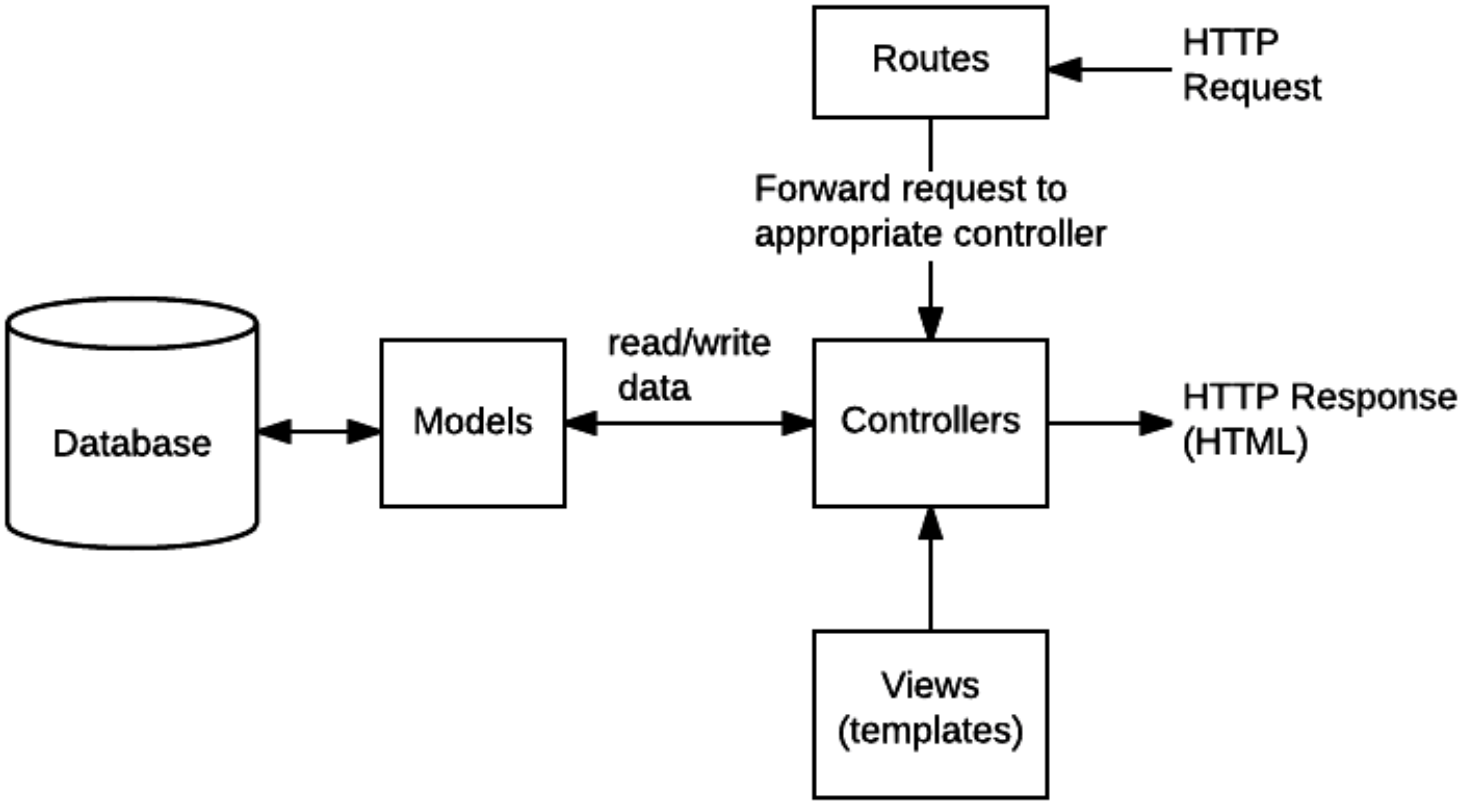


Route

```
const router = express.Router();
app.use('/route', router);
router.get('/', (request, response) => { //route/index
    response.send('Hello world Route with Nodemon 1!');
});
```



MVC in express



- Cài đặt body-parser

```
$ npm install body-parser
```

- Cấu trúc project

model.ts

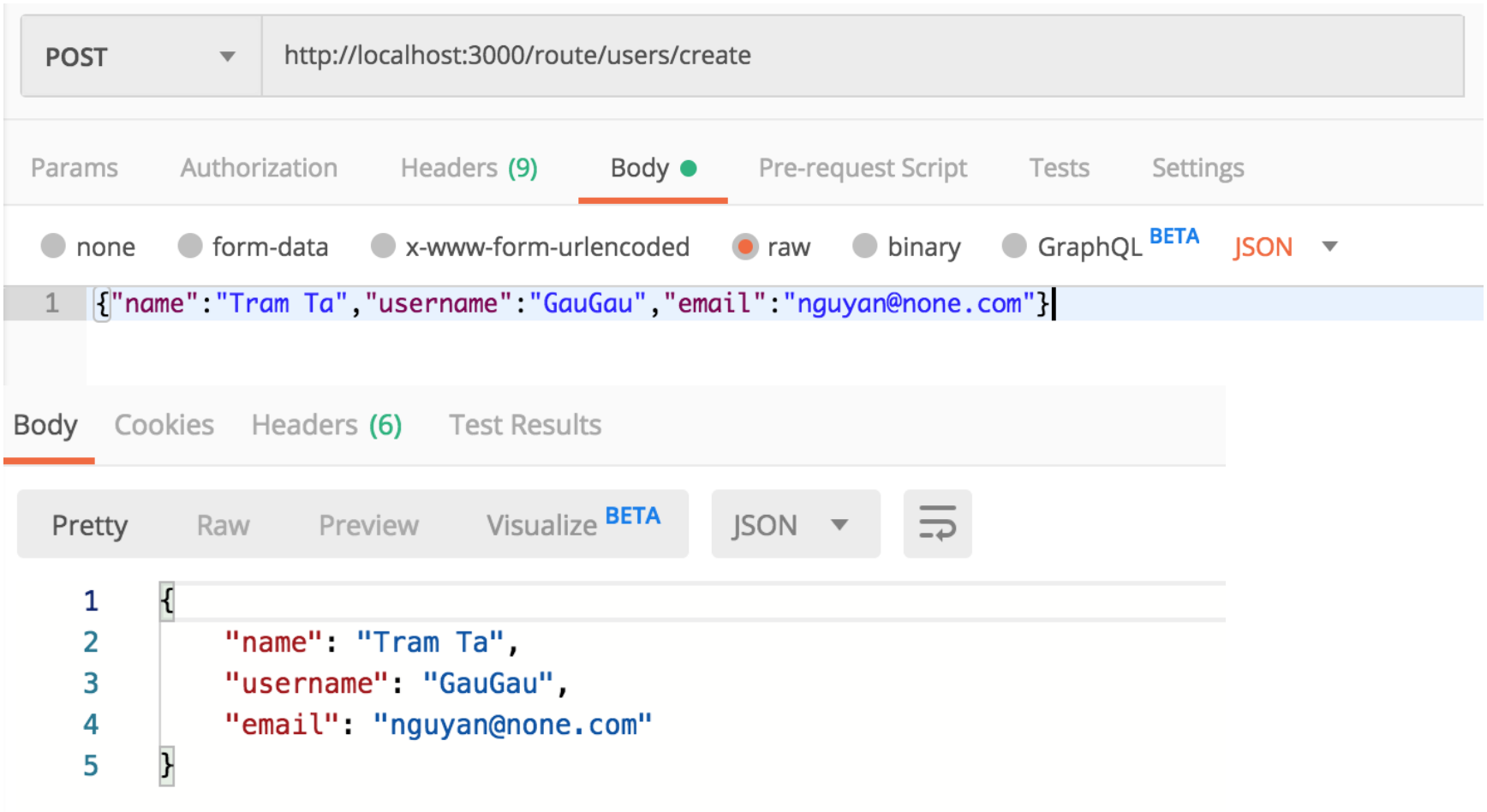
```
export class User {

    public username: String;
    public name: String;
    public email: String;

    constructor(name: String, username: String, email: String) {
        this.name = name;
        this.username = username;
        this.email = email;
    }

    getUsername() {
        return this.username;
    }
    getName() {
        return this.name;
    }
}
```

Controller.ts

```
import { Request, Response } from 'express';
import { User } from './model';

export let users = (req: Request, res: Response) => {

    let users = [
        new User('James Coonce', 'jcoonce', 'james@none.com'),
        new User('Jim Coonce', 'jimcoonce', 'jim@none.com'),
        new User('Norman', 'jcoonce', 'norman@none.com'),
    ];

    res.json(users);
};

export let create = (req: Request, res: Response) => {
    res.json(req.body);
};
```

App.ts

```
import express, { Request, Response, NextFunction } from 'express';
import bodyParser from 'body-parser';
import { json } from 'body-parser';
import * as controller from './controller';

const app = express();


app.use(json());
app.use(bodyParser.urlencoded({ extended: true }));


const router = express.Router();
app.use('/route', router);
router.get('/', (request, response) => { //route/index
    response.send('Hello world Route with Nodemon 1!');
});
router.get('/users', controller.users);
router.post('/users/create', controller.create);
```


POST
http://localhost:3000/route/users/create
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL BETA
JSON
1 {"name":"Tram Ta","username":"GauGau","email":"nguyan@none.com"}
Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
Visualize BETA
JSON
1 {
2 "name": "Tram Ta",
3 "username": "GauGau",
4 "email": "nguyan@none.com"
5 }


demo

SUMARRY

- ☑ Thư viện: lodash, class transformer, class validator
- ☑ Express:
  - ❖ request, reponse, get, post
  - ❖ listen
  - ❖ body-parse
  - ❖ route, middleware, mvc trong express


thank
you!