# Video 1:

## Tổng kết kiến thức series sau khai giảng

* Dùng @controller để biến java class thành 1 cái resfulapi webservice
* Truy xuất vào webservice thông qua url và method phải định nghĩa @RequestMapping
* @RequestParam nhận request đầu vào
* @ResponseBody trả json cho clien
* @RequestBody convert json sang object có thể chuyển hóa sang json
* Bên phía client có thể gửi nhiều type đầu vào ví dụ: text, json, html,…
* Response về cho client cũng có thể nhiều dạng dữ liệu

## Giới thiệu về @RestController

* Để build webservice ngoài @Controller còn có @RestController
* Khi mà dùng @RestController thì không cần phải sử dụng @ResponseBody(Để convert object sang json) nữa vì ở bên trong @RestController đã có @ResponseBody rồi

## Cách khác khi dùng @RequestMapping để định nghĩa url và http method

* Ngoài cách @RequestMapping(value, method)
* Còn có cách ngắn gọn hơn là GetMapping(url), PostMapping(url), PutMapping(url), DeleteMapping(url).

## Giới thiệu rule 8: don’t nest resourses

* Thay vì làm url kiểu cha con như /authors/12/articles/ tức là bên authors là 1 bên articles là nhiều và lấy id của authors là 12
* Thì làm url /articles/?author\_id=12 người khác nhìn vào dễ hiểu hơn

## Giới thiệu rule 4: dùng dạng số nhiều

## Khi đi làm phải thống nhất URL

* Bên frontend và backend phải thống nhất với nhau
* url tên là gì, method là gì, data trả ra có không

## Viêt 1 api xử lý nhiều chức năng

/api/users?buildingId=10&role=staff -> api load nhân viên trong giao tòa nhà

/api/users?role=manager -> api load toàn tộ ds quản lý

/api/users -> api load toàn bộ user

# Video 2:

## Giới thiệu rule 2: Hạn chế return về text

## Giới thiệu về Content-type và datatype

* Client phải định nghĩa conent-type để khi gửi về server nó biết đó là kiểu gì ví dụ json, text,…
* Datatpe là dữ liệu của server trả về

## Cách gọi api ở html

* Dùng ajax

## Giới thiệu về HTTP status code

* Có thể can thiệp được

**Successful responses (200-299)**

**Client error responses (400-499)**

* 400 badrequest: sảy ra khi data client gửi về có vấn đề. Ví dụ url có require param nhưng client không chuyền param vào
* 401
* 402
* 403
* 404 not found: không tìm thấy url. 2Th: 1 là server không có 2 là server 1 đường client 1 nèo
* 405 (method not allow): sai về mặt dữ kiện về mặt method. Ví dụ như là server định nghĩa là POST client lại là GET.

**Server error response(500-599)**

* 500 interal server:
  + Bug
  + Cấu hình sai hoặc quên cấu hình
* 502 bad gateway: sau khi server sập thì sẽ bị 502
* 503 servicee unvailable: server sập
* 504 gateway timeout: client chờ response quá lâu sẽ bị timeout

## Giới thiệu rule 6: Chú ý vào status code

* Không là cãi nhau giữa frontend – backend

## Giới thiệu rule 7: sử dụng status code thống nhất

* Custome http status code. Lỗi gì trả về lỗi đó

## Giới thiệu rule 5: trả về chi tiết lỗi trong response body

Ví dụ lỗi chung là field is require

Name

Hay là numberofbasement

## Giải pháp cho 2 vấn đề

1: http status code không consistent treo đầu dê bán thịt chó

2: custome error detail lặp code

* Giải pháp cho thằng 1 cũng là cho thằng thứ 2 luôn kĩ thuật đó được gọi là @ControllerAdvice

## Giải quyết vấn đề định nghĩa HTTP method dựa vào tác vụ chứ không dựa theo phát âm

TH1: nếu tòa nhà đó chưa có ai quản lý hết thì là thêm mới -> post

TH2: đã có người quản lý rồi và thay đổi người khác, xóa người cũ đi, thêm người mới vào -> có cả POST, PUT, DELETE . Vậy thì chọn đại 1 cái dùng post (nó phổ biến)

# Video 3:

## Giới thiệu @ControllerAdvice

* Là kĩ thuật sử lý vấn đề
  + Custome error detail lặp code
  + http status code không consistent
* Bắt những ngoại lệ ném ra
  + Của hệ thống
  + Ngoại lệ tự tạo
* Có thể định nghĩa nhiều Exeptionhanler

## @ResponseEntity

* Ngoài data thì còn chứa status code

## Cách custome exception để trả ra lỗi của field requied

Mặc định khi custome exception thì nó sẽ là lỗi checked(biên dịch)

Extends exception

Tạo method check value của field

## Khác nhau giữa throw new và thows

* Throw là ném ra exception
* Throws là thông báo rằng nó ném ra ngoại lệ đó

# Video 4:

## Phân quyền là gì

Vai trò này có quyền làm gì

Ví dụ: những api chỉ cho phép manager thì staff không vào được

## Giới thiệu Rule 11: Hiểu rõ về 401: authorized và 403 forbidden

* 401:
  + Chưa đăng nhập
  + Đăng nhập sai
  + Token hết hạn
* Nên là chưa sinh ra token(sinh ra ở backend nên có thể kiểm soát được dữ liệu chứa trong token)
* 403 forbidden:
  + Không có quyền truy cập vào api

# Video 1: MVC

## Debug, tìm flow dự án

## Để hết code vào 1 chỗ thì sảy ra vẫn đề gì:

* Người đọc sẽ không hiểu gì cả
* File, method lượng code lớn ( clean code khuyên không nên )
* Đổ hết tất cả logic vào cùng một method

## Giới thiệu SOLID

* S: single reposibility là nhà ai người đấy ở việc ai người đấy làm

## Giới Thiệu Mô hình MVC

Mvc

* Modal: là đối tượng luân chuyển dữ liệu
* View : là nơi trả data
* Controller: api
* Là 1 design parttern

Luồng vào

* View đẩy data vào modal, modal đẩy dữ liệu vào controller
* Controller trả dữ liệu cho modal, modal đẩy dữ liệu ra cho view

## Giới Thiệu Mô hình 3 layer

* Bản thân thằng mvc nằm trong tầng presantation của 3 layer
* Tầng bussiness logic: thực thi logic
* Tầng Dataaccess: tầng làm việc với sql
* Mỗi layer sẽ có đối tượng xử lý riêng (filter : besiness logic, entity: data access, dto: presentation)

# Video 2

## Tại sao lại phải áp dụng mô hình 3 layer

* 1 method bao gồm quá nhiều chức năng ví dụ findAll: nhân data từ client, xử lý logic, lấy dữ liệu từ sql -> quá dài, khó đọc, khó hiểu
* Áp dụng mvc 3 layer

## Luồng lấy dữ liệu lên

Resultset -> entity -> filter -> dto (modal)

Repository -> besiness logic -> mvc

## Update dữ liệu xuống

Modal -> filter -> entity-> database

Mvc -> service -> repository

## Comment trong eclipse

2 cách:

* //
* /\*\*/: comment và viết docs