# Buổi 1:

a

* Java web dùng spring framework
* Cms – trả phí (AEM, magnolia)
* J2ee/jakarta ee: serverlet, JAX-RS

13: Xây dựng web service sử dụng @controller

Thuật ngữ trong khóa học (chỉ trong khóa học thôi)

* 2 project:
  + web (spring boot web) ,
  + web service (spring boot api)
    - Có phân quyền (spring boot api with jwt)
    - Không phân quyền (spring boot api non jwt)

28: Phải nhớ phiên bản spring boot

* Spring boot 1.5.22 – spring fram work 4.3.25
* Jackson 2.8

33: Spring boot starter

* Là 1 cụm thư viện
* Có thêm một số phần cấu hình

42: jackson – databind, annotations, core

* Jackson là thằng chung gian để chuyển data về kiểu dữ liệu chung
* Không có 3 thằng này không làm được webservice

46: khi dùng parent version spring boot phải quan tâm tương đương spring framwork version bao nhiêu

Ví dụ:

* parent dùng 1.5.22.REALEASE
* Spring framework version là 4

48: tại sao dùng 2.0.9

* 2.0.9 là final của bản cuối cùng của 2.0.x

50: 2.0.9

* Spring framework: 5.0.13
* Jackson 2.9

54: properties

* Java.version: bản chất là cú pháp của parent. Mục đích là khi biên dịch nó hiểu được những đoạn code của java 1.8
* Cần thiết vì ở windown chúng ta chỉ cấu hình môi trường. Default version của nó là 1.5

1h: Xây dựng webservice phải có bài toán cụ thể

Quản lý tòa nhà bao quát khá nhiều trường hợp

1h3: Role Đặt tên

* Object + package

1h4: cách biến 1 java class -> webservice

Dùng @controller

@annotation:

2 loại:

* Dùng của người khác
* Tự viết

# Buổi 2

6: Xây dựng webservice ngoài cách @controller

* Còn có RestController

Khác nhau:

-

8: Truy xuất webservice giống như truy xuất vào website thông qua url

Chỉ khác là webservice trả ra data json, xml còn website trả ra ui

10: để định nghĩa url

* Spring framwork hỗ trợ @RequestMapping(value = “”) dưới thằng annotation này phải có method để annotation này thực thi

Ví dụ: ở trang web khi mà truy xuất vào trang web thì nó sẽ có method để đọc dữ liệu của page lên sau đó nó đẩy vào ui.

13: dùng thuật ngữ client

* Không biết nó là web, mobile hay là thiết bị nào đó có thể tương tác JSON

14: server không bao giờ trả ra ngay kiểu JSON

* Trả ra đối tượng, mà đối tượng này có cấu trúc có thể render ra kiểu JSON
* Sau đó có 1 annotation sẽ hỗ trợ chúng ta chuyển đổi sang JSON

17: Liệt kê đối tượng có cấu trúc JSON

* Java bean
  + Single
  + Danh sách
* Map interface (dùng hashmap)
  + Single
  + Danh sách

Json: cấu trúc là key – value

Java bean: key: name field – value : value

21: Client gửi data json về

* Có 1 annotation sẽ chuyển đổi json map vô trong object được tiếp nhận ở trông server

Ví dụ

* Định nghĩa là map interface nó chuyển đổi thành map interface
* Định nghĩa là java bean nó chuyển đổi thành java bean

27: Phải có rule practice

* Khi làm ở ngoài công ty là làm nhóm không phải làm mình
* Ai cũng code như ai => dễ quản lý, đọc

31: @ResponseBody

* Dùng để convert data -> JSON

32: Luồng chạy chương trình khi

Khi request vào url /api/building

* B1: nó sẽ quét toàn bộ java class và sẽ loại tất cả java class không có @controller
* B2: nó tìm url chính xác với url chúng ta request dựa vào value của request mapping
* B3: thực thi cái method. Dữ liệu trả ra là object or list object sau đó sẽ được chuyển đổi sang json bằng @ResponseBody

39: @RequestBody

* Map với đối tượng mà server nhận vào

Ví dụ: Thêm tòa nhà client gửi json thì khi server nhận vào nó sẽ chuyển đổi json -> java bean bằng @RequestBody

40: Chú ý với url của webservice

* 3: avoid using verbs in url

Không nên dùng động từ vì rule 1 với rule 3 nó liên quan đến nhau

Nếu chúng ta áp dụng rule chỉ cần phá vỡ 1 rule thì những rule sau không áp dụng được nữa

41: Dùng chung 1 url

Ví dụ : crud đều là /api/building

Phân biệt crud thông qua http method

44: HTTP method

* Gồm 4 thằng hay dùng: GET, POST, PUT, DELETE

Khi dùng url chung thì phải bao gồm http method để nó phân biệt

50: Dữ liệu có 2 loại gửi

* Body: @RequestBody
* Param

# Buổi 3

1: Collection trong postman

* Tập hợp các tệp api theo project

13: Không dùng verb để đặt tên url rule 3

14: CRUD thì dùng chung url và định nghĩa thêm thằng http method cho server nhận diện

15: Dựa vào method để sử dụng thằng http method cho đúng

* Không dựa vào cách nói để suy ra http method

Ví dụ: Thêm tòa nhà -> post, lấy tòa nhà -> get

24: cách gửi dữ liệu về server = param

Parameter động

* url?ward=avc & district=abc
  + sử dụng @RequestParam

parameter cứng

* url/id

27: nói lại về @ReponseBody và @RequestBody

36: Param

* muốn là kiểu data gì thì chuyền vô kiểu data đó

44: Debug cơ bản

* run thì chạy 1 mạch từ đầu đến cuối có lỗi thì lỗi luôn
* Debug để can thiệp từng dòng code để biết lỗi chỗ nào

Khi dùng debug có 1 khải niệm là breakpoint

F6: step over để nhảy từng dòng 1

F8: Resume tắt để chạy hết luôn

56: Skip all breakpoint

* Tắt hoạt động các breakpoint

58: Không phải frontend nào cũng tốt

* input số thì phải nhập số -> không có thằng frontend nào cũng tốt bụng làm chuyện này

1h3: Dựa vào HTTP status code

* để biết được lỗi do backend hay frontend

1h4: client truyền sai kiểu dữ liệu ở param thì server sẽ báo lỗi do thằng @RequestParam xử lý

* Tất cả data từ param đều là string hết
* @requestParam sẽ giúp chúng ta ép kiểu với điều kiện là data có fomat giống kiểu ta muốn ép

# Video 4

7: Nếu không sử dụng spring framework mà sử dụng 1 cái khác để build RESTful API webservice

* Hiểu là nó chỉ dừng ở mức String (chuyển sang định dạng nào là do chúng ta) ở request param

9: Khi nào data bị null, khi nào bị empty (Request param)

* Null : không cần gửi key và value
* Empty: không chứa value

10: Sử dụng object trong trường hợp nào

* Đầu vào đa hình dạng

18: Biến Request param thành dạng động

* Thêm require = false

41: Gửi tang-tret và nguyen-can cùng một lúc trong request param

2 cách gửi

* Request: types= tang-tret,nguyen-can (Trong server nhận là dạng mảng hoặc chuỗi
* Types = tang-tret & types = nguyen-can (nhận arraylist)

1h: @Pathvariable

* Dùng cho parameter cứng như là id

1h11: Body, raw

* Dựa theo data request mà chọn text or json, xml ,…

# Buổi 5

20: Bản chất data gửi về server theo dạng là body, không giống như get nhạn vào là param

21: 2 cách nhận list body ở method delete

* Nhận trực tiếp mảng ids thông qua @Requestbody
* Nhận cả 1 object chứa mảng ids
  + Không sai nhưng có một vấn đề là những field null không có dữ liệu là đúng hay sai

32: http status code

* Trách cãi nhau giữa fontend và backend
* 98%

42: viết code, đặt tên làm sao cho phù hợp

* Để sau này nhìn lại hay người khác dễ đọc, dễ hiểu

51: Ghi chú để mô tả method