Converting JSON body of Http request to Java class obj using@RequestBody (Complex data)

==========

=>@ReqeustBody and @ResponseBody Annotations are called Media type annotations

becoz they are useful to decide media of the incoming and outgoing content =>@RequestBody is useful to convert JSON/XML data to Java class object

and @ResponseBody is useful to convert Non-String data to JSON/XML data through DispatcherServlet

with

=>@RequestBody dealing 1D and 2D Arrays SON

1D array:: array/List/Set Collections "<variable>! [<val1>,<val2>,<val3>,...]

2D array:: Map

2D Array :: HAS-A property

"<variable>":{"<key1>":<val1>,<key2":val2,...}

"<variable>":{"<subProp1>":<val1>,"<subProp2>":val2,..}

SpringBootRestProj06-JSON ToObjectUsing RequestBody [boot]

> Deployment Descriptor: SpringBoot RestProj06-JSONToObjectUsingReque >Spring Elements

> JAX-WS Web Services

src/main/java

> com.nt

com.nt.controller

> CustomerOpeationsController.java

›

com.nt.model

Address.java

> Customer.java

>src/main/resources

src/test/java

JRE System Library [JavaSE-11]

>

>

>

>

src

Maven Dependencies

Deployed Resources

> target

WHELP.md

mvnw mvnw.cmd Mpom.xml

Customer.java

```
package com.nt.model;
import java.util.List;
import java.util.Map;
import java.util.Set;
import lombok.Data;
@Data
public class Customer {
}
private Integer cno;
private String cname;
private String[] favColors;
private List<String> academics;
private Set<Long> phoneNumbers;
private Map<String,Double> billDetails; private Address addrs;
```

Address.java

```
package com.nt.model;
import lombok.Data;
@Data
public class Address {
private String houseNo; private String streetName;
private String location;
private Long pinCode;
//Rest Controller class
package com.nt.controller;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping; import
org.springframework.web.bind.annotation.RequestBody; import
org.springframework.web.bind.annotation.RestController;
import com.nt.model.Customer;
@RestController
public class CustomerOpeationsController {
@PostMapping("/register")
public ResponseEntity<String>saveCustomer(@RequestBody Customer cust) { return new
ResponseEntity<String>(cust.toString(), HttpStatus.OK);
```

POST

(a)

}

}

(g)

(b)

Send

http://localhost:3030/SpringBoot RestProj06-JSON ToObjectUsing RequestBody/register

Params

Authorization

Headers (8) Body⚫

Pre-request Script

Tests

Settings

(c)

⚫ none ⚫ form-data

x-www-form-urlencoded

⚫raw

⚫ binary ● GraphQL

JSON ✓

(d)

{

"cno": 101,

"cname": "rajesh",

"favColors":[

"red",

"blue",

"gree"

],

"academics":[

"10+2",

"B.Tech",

"M.Tech"

"phoneNumbers": [

5535353435,

(e) type this cotent

Cookies

Beautify

(f)

While converting XML content of request body to Model class obj using @RequestBody annotation we need to add "Jackson-dataformat-xml" jar file to CLASSPATH/BuildPath

NTSPBMS616 / test6-req2-xml

POST

(1)

<

(2)

A Save

(7)

Send

E

http://localhost:4040/Boot RestProj06-ComplexJSON-XMLBidingToObjectUsingRequestBody/customerapi/register

Params

Authorization

Headers (8) Body

Pre-request Script

Tests Settings

none

⚫ form-data

x-www-form-urlencoded

raw

⚫ binary ● GraphQL XML

(4)

(5)

Cookies

در

Beautify

€

54353455,

1

<customer>

2

5353533543

3

],

"billDetails": {

4

<cno>101</cno>
<cname>raja</cname>
<billamt>90000</billamt>
5
<addrs>
6
<houseNo>1-2-3-10/2a</houseNo>
},
"x-mas tree": 6546.77,
"cake": 5345.5,
"chocolates": 6789.6
"addrs":{
"houseNo": "1-2-4/566",
"streetName": "RKStreet",
"location": "hyd",
"pinCode": 522345
7
<streetName>RKStreet</streetName>
8
9
10
<areaName>sanath Nagar</areaName>
<pinCode>500018</pinCode>
</addrs>
11
<nickNames>
(6)
12
<nickNames>king</nickNames>
13
<nickNames>janu</nickNames>
14
<nickNames>sona</nickNames>
15
</nickNames>
16
<friends>

17

<friends> karan </friends>

18

<friends> kiaran </friends>

19

<friends> tarun </friends>

20

</friends>

}

21

<idDetails>

22

<aadhar>134535677</aadhar>

23

<voterId>53456777</voterId>

24

</idDetails>

Body

Cookies Headers (5) Test Results

Pretty

Raw

Preview

Visualize

Text ✓

P

A 200 OK 227 ms 459 B Save Response

25

<phoneNumbers>

26

27

28

1

Customer (cno=101, cname=rajesh, favColors=[red, blue, gree], academics=[10+2, B.Tech, M.Tech], phoneNumbers [54353455, 5535353435, 5353533543], billDetails={x-mas tree-6546.77, cake-5345.5, wait chocolates=6789.6}, addrs-Address (houseNo-1-2-4/566, streetName=RKStreet, location=hyd, pinCode=522345))

29

<phoneNumbers>354353455 </phoneNumbers>

<phoneNumbers>3543531355 </phoneNumbers>

</phoneNumbers>

</customer>

for this

Body Cookies

Headers (5)

Test Results

Status: 201 Created Time: 571 ms Size: 465 B

TO GO

outoput

Pretty Raw

Preview

Visualize

Text

1 Customer (cno=101, cname-raja, billamt-90000.0, addrs-Address (house No-1-2-3-10/2a, streetName=RKStreet, areaName=sanath Nagar, pinCode=500018), nickNames=[king, janu, sona], friends=[karan phoneNumbers=[354353455, 3543531355], idDetails={aadhar=134535677, voterId=53456777})

1

kiaran, tarun],

(8) wait for this output

Converting JSON Data to Java class object using @RequestBody dealing with List<Object>, date, time values

For List<Object> or Collection<Object> we need to take

"<variable>": [{"key":value,"key":value,...},

{"key":value,"key":value,...},

{"key":value,"key":value,...}

1

SpringBootRestProj07-JSONToCollectionObjects UsingRequestBody [boot]

>

Deployment Descriptor: SpringBoot RestProj07-JSONToCollectionObjectsl

> Spring Elements

> JAX-WS Web Services

#src/main/java

> com.nt

com.nt.controller

> CustomerOpeationsController.java

com.nt.model

>

Company.java

> Customer.java

#src/main/resources

static

templates

application.properties

>

src/test/java

>

JRE System Library [JavaSE-11]

>

Maven Dependencies

> L Deployed Resources

> src

>

target

w HELP.md

mvnw

mvnw.cmd

M Dom.xml

Customer.java

Company.java

```
package com.nt.model;
import lombok.Data;
@Data
public class Company {
}
private String name; private String location; private Integer size;
```

//RestController

```
package com.nt.controller;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping; import
org.springframework.web.bind.annotation.RequestBody; import
org.springframework.web.bind.annotation.RestController;
import com.nt.model.Customer;
@RestController
public class CustomerOpeationsController {
```

@PostMapping("/register")

package com.nt.model;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Map;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

@Data

public class Customer {

private Integer cno;

private String cname;

private List<Company> companiesList;

@JsonFormat(pattern="yyyy-MM-dd")

private LocalDate dob;

//@JsonFormat(pattern="yyyy-MM-dd HH:mm:ss") private LocalDateTime purchase Date;

private List<Map<String,Long>> familyIds;

public ResponseEntity<String> save Customer(@RequestBody Customer cust) {

return new ResponseEntity<String>(cust.toString(), HttpStatus.OK);

application.properties

server.port=4040

server.servlet.context-path=/JSONToJavaObjectApp1

}

}

Run this application as spring Boo App

(a)

POST

NATARAZ- JAVA CONSULTANT

http://localhost:4040/JSONToJavaObjectApp1/register

(b)

Params

Authorization

Headers (8) Body ⚫

none ⚫ form-data

x-www-form-urlencoded

raw

Pre-request Script Tests Settings

⚫ binary ● GraphQL JSON V

(d)

1

{

"cno":1001,

"cname":"raja",

"companiesList":[

{"name":"HCL","location":"hyd","size":200},

{"name":"BPCL","location":"blore","size":400},

{"name":"HPCL","location":"delhi","size":500}

(e) type

],

this content

"dob":"1990-11-20",

"purchaseDate":"2015-10-21 17:10:55",

"familyIds":[

]

{"aadhar":354535,"voterId":5353543,"panNo":45435453},
{"aadhar":314535,"voterid":15353543,"panNo":24535453}

(£)

Send

Body

Cookies Headers (5) Test Results

Pretty

Raw

Preview

Visualize Text

P

200 OK 186 ms 509 B Save Response

ПС

get this output

Go to Settings to activate Wind

```
1 Customer (cno=1001, cname-raja, companiesList = [Company (name=HCL, location=hyd,
size=200), Company (name=BPCL, location=blore, size=400), Company (name=HPCL,
location=delhi, size=500)], dob-1990-11-20, purchaseDate=2015-10-21T17:10:55,
familyIds=[{aadhar=354535, voterId=5353543, panNo=45435453}, {aadhar=314535,
voterId=15353543, panNo=24535453}])
```

Activate Windows

note:: if the request body contains invalid JSON Pattern content which can not

be parsed by JSON parser then the RestController sends 400 (Bad Request) error response to browser.

Assingment ::

=> Build Json format body for the class Structure

class Courier{

class PaymentGateway{

class E_Commerce{

private String name;

private String addrs;

private Integer size;

}

private List<Courier> couriers;

private Set<PaymentGateway> gatewaysInfo;

}

}

...

...

Passing request Params to Spring Rest App

=>The GET mode request does not contain body.. So the data we want to send

=====

in GET mode request should always be in the form of Query String (?p1=val1&p2=val2&p3=val3) =>The GET mode request can carry limited amount of data (max of 2KB) that to as Query String params

(query String having

request param names

and values)

=> In other mode requests (like POST,PUT, DELETE and etc..) the content in the request goes to server in the form of request body.

For Spring RestController we can pass data in GET Mode request in two ways

a) As request Params/ Query params in the Query String of request URL

(eg: url?key1=val1&key2=val2&key3=val3)

(Supported by Spring MVC and Spring Rest)

(Best) b) As Path variable values in the requst URL

we can pass inputs along wit request from consumer app to Producer App

(@Requestest body

a) As the content of

@RequestParam

(useful in other than GET mode HEad mode)

b) As the Query String content in the request URL

c) As the Path Variable values in the request URL @PathVariable

Useful in GET mode

(path variables are preferred)

=>For query string based req params, we use @RequestPAram annotation => For path variables,we use @PathVariable annotation

eg:: http://localhost:3030/RestProj01/customer-api/report?cno=101&cname=raja

req params in query string

(Supported by Spring Rest .. not in Spring MVC)

URL

Path Variable Values

(eg: url/value/value/value)

eg:: http://loclahost:3030/RestProj01/customer-api/report/101/raja/hyd

3 {key} will be given in request path of

b.methods placed""@RestController. @xxxMapping("</path/{key1}/{key2}/...")

note:: All web technologies /frameworks (Both java and Non-Java) supports request params as the concept web programming

BASIC

(eg: servlet,jsp, php, asp.net, nodejs, express js,struts,jsf and etc..)

note:: Passing values in the request URL as PATH variable values is introduced in RestFull

progrmaming, So the all tehcnologies and frameworks supporting the Restfull programming

(both java and non-java) gives provision to work with this path variable concept

(eg: RestEesy, Jax-RS, Jersery, spring/spring boot Rest, ...)

a) Request Params/ Query Params in query String

=>To place them in request url,we need

URL?key1=val1 &key2=val2&key3=val3)

them

=>To read in the method of @RestController

@RequestParam("key") paramType paramName

(or)

@RequestParam ParamType paramname

It must match with key of the request parameter

RestController class

Example App

==========

SpringBootRestProj08-RequestParams [boot] [devtools]

> Deployment Descriptor: SpringBootRestProj08-RequestParam

>Spring Elements

> JAX-WS Web Services

src/main/java

>com.nt

com.nt.controller

> CustomerOperationsController.java

#src/main/resources

>

src/test/java

> JRE System Library [JavaSE-11]

> Maven Dependencies

> C Deployed Resources

==========

package com.nt.controller;

import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.RestController;

@RestController

public class CustomerOperationsController {

> src

> target

w HELP.md

mvnw

mvnw.cmd

M pom.xml

GET

@GetMapping("/report")

public String reportData(@RequestParam("cno") Integer no, @RequestParam String cname) {

return no+" "+cname;

http://localhost:3030/SpringBoot RestProj08-RequestParams/report?cno=101&cname=raja

Params.

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

KEY

VALUE

DESCRIPTION

✔

cno

101

✓

cname

raja

Key

Value

Description

Body Cookies Headers (5)

Test Results

200 OK 190 ms 175 B

Pretty Raw

Preview

Visualize

Text

1 101 rajal

using

=>while request params in query String we can change the order of passing req param values

report?cname=raja&cno=101

(or)

report?cno=101&cname=raja

Both are correct

=> while using request params in query String if we pass addtional params than expected that will not generate any eror

?cname=raja&cno=101&cadd=hyd

(if we pass less than required params then we get 400 error)

?cname=raja

Gives 400 error

Extra but does not generate error

=>when we pass queryString to the url in POST mode request.. they internally become request body content becoz the POST carries data as request body.

```
@RestController
public class CustomerOperationsController {
@GetMapping("/report")
@PostMapping("/report")
public ResponseEntity<String> reportCustomerData(@RequestParam int cno, @RequestParam(required =
```

```
false,defaultValue = "king") String cname){
System.out.println(cno+"........"+cname);
return new ResponseEntity<String>(cno+" ....' ."+cname, HttpStatus.OK);
we can use "defaultValue" param in combination with required="false"
public String reportData(@RequestParam("cno") Integer no,
return no+" "+cname;
@RequestParam(required = false) String cname) {
}
}
}
```

The default value of

required param is "true"

URL:: http://localhost:3030/SpringBoot Rest Proj08-RequestParams/report?cno=101 gives 101 NULL as response

```
@RestController
public class CustomerOperationsController {
@GetMapping("/report")
public String reportData(@RequestParam("cno") Integer no,
@RequestParam(required = true) String cname) { (default value of required param is true)
return no+" "+cname;
}
}
```

URL:: http://localhost:3030/SpringBoot Rest Proj08-RequestParams/report?cno=101 =>Gives 400 bad request

=>In GET mode request, we can pass any no.of req params but size should not cross 2KB

=> In POST mode request, we can pass any no.of req params with out size restriction.

Request params discussion in Spring boot

rest/Spring Rest is same as spring MVC or Spring boot MVC

In Restful Apps req params are not standard to pass,

and required="true"

Sending

======

data as Path variable values

==========

no.of

path variables are standard to pass data IN GET Mode request

che

=>Supports to minimise characters in request url while sending data

=> No need of passing seperate query String in the request URL to send data, we can pass data directly in

the request url it self

(or complex)

=>passing "& " as the value in request param values is not possible.. but can be done

easily using path

variable s

note:: In Restful Apps do not use request params to send

data along with the request in GET mode

note:: In Restful Apps prefer using Path variables to send

data along with the request in GET mode

=>In request params, to pass & as the content of param value

=> request URL with query String is not Clean url becoz it needs more chars (becoz both keys and values are required) and queyString needs seperate syntax to fallow

use %26

http://localhost:3131/BootRestProj07-RequestParams/actor-

api/report?id=1001&name=jack%26jhon

=> Rquest URL with path variable values are part of request url itself and no need of following seperate syntax for it. (URL is clean URL)

syntax :: rquest url (or) Path/<value1>/<value2>/<value>/ .....

Path Variable Values

To read path variable values in @RestController methods

@PathVariable Datatype paramname

(or)

@PathVariable("key") dataType param name

at @RestController in XxxMapping(...)

be

{key} for those values will give n

method

param name must match {key} of request path

method while defining the request path eg:@GetMapping("/report/{cno}/{cname}/{cadd}") ↓

↓

static path

Dynamic path

(method param name and

.with

=>The request path contains two parts while working path variables

a) static path (fixed path) (/<path>)

b) Dynamic path (key name whose value s

comes from request url) (/{key})

eg:: GetMapping("/report/{no}/{name}")

static path Dynamic path

example rquest url

the {key} request path need not to match

(1)

http://localhost:3035/SpringRestProj9-PathVariables/report/101/raja

12)

@GetMapping("/report/{no}/{name}")

✓ (c) @PathVariable String name){ (3)

public ResponseEntity<String> showReport(@PathVariable("no") int cno,

}

static path variable values path (Dynamic path values)

✓ SpringBoot RestProj09-PathVariables [boot] [devtools]

> Deployment Descriptor: SpringBootRestProj09-PathVariables

>Spring Elements

> JAX-WS Web Services

src/main/java

> com.nt

com.nt.controller

> CustomerOperationsController.java

>#src/main/resources

> #src/test/java

> JRE System Library [JavaSE-11]

>

Maven Dependencies

> C Deployed Resources

> src

> target

WHELP.md

mvnw

mvnw.cmd

Mpom.xml

//RestController

==========

package com.nt.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable; import

```
org.springframework.web.bind.annotation.RestController;
@RestController
public class CustomerOperationsController {
@GetMapping("/report/{no}/{name}")
public String fetchData(@PathVariable("name")String cname,
@PathVariable Integer no) {
return _no+ "<----->"+cname;
}
}
```

http://localhost:3030/SpringBoot RestProj09-PathVariables/report/101/raja (b)

Body Pre-request Script Tests Settings

GET (a)

Params Authorization

Headers (6)

Query Params

KEY

Key

VALUE

Value

Body

Cookies Headers (5) Test Results

Pretty

Raw Preview Visualize Text

```
1
101<----->raja
```

| (d) observe the output

While working with request param we can change the order of passing their values

/report?cno=101&cname=raja

is same as

/report?cname=raja&cno=101

with

While working parth variable values we can not change the order of passing values

request path in @RestController method is :: GetMapping("/report/{no}/{name}")

/report/101/raja --> valid

/report/raja/101 ---> invalid -->Gives 400 Bad request if

@PathVariable Integer no is taken

```
@GetMapping("/report/{no}/{name}")
```

public String fetchData(@PathVariable("name")String cname,

@PathVariable Integer no) {

return no+ "<----->"+cname;

}

Raises

Error

DESCRIPTION

Description

CED

200 OK 14 ms

Activate

/report/raja/101 ---> valid --> if both paramare taken as String values

but the wrong will be stored in method paranns

so the b.logic will be distrubed

@GetMapping("/report/{no}/{name}")

public String fetchData(@PathVariable("name")String cname,

@PathVariable String no) {

return no+ "<----->"+cname;

Does not rais error

but cname holds 101 and no holds raja which

}

gives 101.... raja

in

may distrub

=>Giving extra path variable values (nothing but extra words path) in request url results

404 error

request path for @RestController method :: /report/{no}/{name}

request url :: http://localhost:3030/SpringBoot RestProj09-PathVariables/report/101/raja/hyd/india

=>if we give more or less values as path variable values than expected then we get

404 error (requested resource is not found)

eg::

@RestController

public class CustomerOperationsController {

@GetMapping("/report/{no}/{name}")

public String fetchData(@PathVariable("name")String cname,

return no+ "<----->"+cname;

@PathVariable Integer no) {

}

}

request url :: http://localhost:3030/SpringRestProj10/report/101/raja

gives 101<---> raja (success)

request url :: http://localhost:3030/Spring RestProj10/report/101/raja/hyd

404 error

request url :: http://localhost:3030/SpringRestProj10/report/101

404 error

request url :: http://localhost:3030/SpringRestProj10/report__

404 error

@RestController

public class CustomerOperationsController {

extra

{no}/{name} are the path variable names

101, raja are the the path variable values cno,name of endpoint method are method param names

(c)

Send

of

b.logics execution.

Extra dynamic values

(less values in request path)

(less values in request path)

@GetMapping("/report/{no}/{name}")

public String fetchData(@PathVariable(name="name",required = false) String cname,

@PathVariable(required = false) Integer no) {

The real purpose of required=false is discussed

return no+ "<----->"+cname;

}

}

request url :: http://localhost:3030/SpringRestProj10/report/101.

in the last example of this document please refer

we expect 101 <---> null should come as the output but

we get 404 error becoz no.of levels in request path are 3 and we are giving

only two,So it says requested resource is not found

request url :: http://localhost:3030/SpringRestProj10/report,

we expect null <------> null but we get

404 error (requested resource is not found)

note:: "defaultValue" param is not there in @PathVariable annotation

if multuple methods are having similar request paths having same no.of levels then the request that is having more static level matchings will get priority (higher priority)

```
@RestController
public class CustomerOperationsController {
}
@GetMapping("/report/no/name")
public String fetchData1(@PathVariable(name="name",required = false)String cname,
@PathVariable(required = false) Integer no){
return "from FetchData1";
}
@GetMapping("/report/no/{name}")
public String fetchData2(@PathVariable(name="name",required = false)String cname,
@PathVariable(required = false) Integer no){
return "from FetchData2";
}
@GetMapping("/report/{no}/{name}")
public String fetchData3(@PathVariable(name="name",required = false)String cname,
@PathVariable(required = false) Integer no){
return "from FetchData3";
}
@GetMapping("/report/{no}/name")
public String fetchData4(@PathVariable(name="name",required = false)String cname,
@PathVariable(required = false) Integer no){
return "from FetchData4";
}
```

http://localhost:3030/SpringBoot RestProj09-PathVariables/report/101/name

output: Fetch Data4

http://localhost:3030/SpringBoot RestProj09-PathVariables/report/no/name

output: Fetch Data1

http://localhost:3030/SpringBoot Rest Proj09-PathVariables/report/no/rajesh

output:: Fetch Data2

http://localhost:3030/SpringBoot Rest Proj09-PathVariables/report/101/rajesh

output: Fetch Data3

of

=>if the method param is @PathVariable for certain {key} dynamic path and that {key} dynamic path is missing in the request path then we need

to place required=false in in the @PathVariable annotation

```
@GetMapping("/report/no/name")
public String fetchData1(@PathVariable(name="name",required = false)String cname,
@PathVariable(required = false) Integer no){
return "from FetchData1";
}
```

In the above rest operation method, "no", "name" are the static paths

of the request path.. but method params expecting to get values from

{. no},{_name} path variables.. but that is not possible becoz they are not there

in the request path.. In that situation prefer using required=false in @PathVariable annotation

```
@RestController
public class CustomerOperationsController {
}
@GetMapping("/report/101/raja")
public String fetchData1(){
return "from FetchData1";
}
@GetMapping("/report/{no}/{name}")
public String fetchData2(@PathVariable(name="name",)String cname,
@PathVariable Integer no) {
return "from FetchData2";
}
```

url:: http://localhost:3030/SpringBoot RestProj09-PathVariables/report/101/raja

url::

output: fetchData1

http://localhost:3030/SpringBoot RestProj09-PathVariables/report/101/ramesh

output: fetchData2

if two methods of @RestController is having same request path with same no.of levels then there is possibility of getting

IllegalStateException during the application startup

```
@RestController
public class CustomerOperationsController {
@GetMapping("/report/101/raja")
public String fetchData1() {
return "from FetchData1";
}
@GetMapping("/report/{no}/{name}")
```

public String fetchData2(@PathVariable(name="name",required = false) String cname,

@PathVariable(required = false) Integer no) {

return "from FetchData2";

}

@GetMapping("/report/101/raja")

public String fetchData3(@PathVariable(name="name",required = false)String cname,

@PathVariable(required = false) Integer no){

return "from FetchData3";

}

Gives

Caused by: java.lang.IllegalStateException: Ambiguous mapping. Cannot map 'customerOperationsController'

method

}

note:: Though Path variables are designed for GET mode request to pass additional data along with the request

we can also use them to send data in other modes requests like POST/PUT/DELETE/PATCH and etc.. (little data)

With

with

s

note:: Working PathVariables is more industry standard compare to working request param while sending small amounts

of data along with requests

What is the difference b/w Request params and path variables way of passing data in spring Rest App ? (important)

==========

Request params

==========

a) Syntax to pass data along with request url is

url?key=val&key=val&.....

b) syntax to read request param values is @RequestParam datatype param

(or)

@RequestParam("key") datatype param

param name

c) while passing request param values in

url the order need not

the query string of

request

to match

we

d) if pass more than

in

query

String

the

we

wilfot

required request params

error

Path variables

==========

=======

(Restful web service)

a) Syntax to pass data along with request url is

url/<static path>/val1/val2/...

{key}/{key}/.. will be defined in @RestController

class methods as part of @XxxMapping(-) annotation s

b) syntax to read path variabe values is

@PathVariable datatype param

(or)

@PathVariable("key") datatype param

c) while passing path variables values in

the request url the order must be matched

d) if we pass more than required

path variable values in the request

url then we get 404 error

less

e) if we pass less than required request params e) if we pass than required

in query Strung then we get error only when

required=false is not taken@RequestParam

path variable values in the request

url then we get 404 error if we

do not take multuple request paths in @XxxMapping annotation of different levels

f) Does not allow to pass '&' as direct values f) Does not allow "/" as value

(we must use

%26 for that)

g) Supported by both spring MVC and Spring Rest g) Supported only in Spring Rest

h) In all web technologies that are there to

develop

web applicaitons in different domains

supports this feature basic concept to pass data

are

i) URL is not clean URL (more characters required

in the url to pass data)

j) Easy to read and interpet

h) supported only in Restfull programming

of all domains..

are

i) URL is clean URL (less characters required

in the url to pass data)

j) complex to read to interpet.

k) Bit slow while sending data

l) Recomaned to use in non RestFull Apps

(web applications)

with

m) Generally used while working GET mode requests

becoz data goes as query String

k) Bit faster while sending the data

l) very useful in Restful apps..

M) works with all modes of of request

becoz data goes directly from url itself

n) if we

can

make

query string going as the

N) we can not hide the data in this model

data of the

request

body

(In POST,PUT modes)

then

wean hidata from the enduser

in

any

request mode becoz path variable

becoz

it does not

appear

in the

browser's address bar

(o) we can give default value for certain request

param by using "defaultValue" param

annotation

Can u give multiple request paths for

RequestParam

values go directly in the reuest url itself

o) There is no "defaultValue" param in @PathVariable annotation

the methods of the @RestController?

@XxxMapping

Ans) yes, we can do that becoz value param in @GetMapping is String[]

The required=false param /attribute of @PathVariable

annotation is very useful to provide multiple request paths

to endpoint values having different levels

eg::

In @RestController

as shown above

```
@GetMapping(value={"/report/{eno}/{ename}","/report/{eno}","/report"))
public ResponseEntity<String> showReport2(@PathVariable(name="eno") String no,
System.out.println(no+"....."+ename);
@PathVariable(required = false) String ename){
return new ResponseEntity<String>(no+"...."+ename, HttpStatus.OK);
}
```

gives 101 .... raja

http://localhost:3131/Boot RestProj11-PathVariables/employee-api/report/101/raja

http://localhost:3131/BootRestProj11-PathVariables/employee-api/report/101

gives 101 .... null

http://localhost:3131/Boot Rest Proj11-PathVariables/employee-api/report/raja

gives raja ....null

http://localhost:3131/Boot RestProj11-PathVariables/employee-api/report

gives null.... null

if we give multiple request paths of different modes for the same endpoint method can u tell me which request

path will be

taken?

Ans) The request path that is specified as the first request path will be taken as the final request path of the endpoint in @RestController class

```
@PostMapping("/report")
@GetMapping("/report")
@PutMapping("/report")
@PatchMapping("/report")
@DeleteMapping("/report")
this will be taken as final request path of this endpoint
public ResponseEntity<String> showData4(){
System.out.println("CustomerOperations RestController.showData4()");
return new ResponseEntity<String>("from ShowData4",HttpStatus.OK);
}
```

note: Different endpoints methods of a @RestController can have

same request path with different request modes

note: if we keep multiple request paths on same endpoint with different modes then we first request will be taken as the final request path