=>Spring Rest = spring mvc++

=>Spring Boot Rest = Spring boot mvc ++

Boot

First Spring Rest Application Development

==========

=> The service provider /publisher /Server App of Restful Webservice of Spring Rest App

will be developedd as @RestController class or @Controller + @ResponseBody class @RestController= @Controller + @ResponseBody

=> The methods of @RestController class are mapped/linked with request mode + request path +params

using xxMapping("<path>") annotations.. These methods can be called afandler methods or Rest Operations or Endpoints b.method of service provider/server/publisher App.. These methods directly or indirectly gets @ResposeBody annotation.. Due to this these methods directly can send output/results to Client/consumer App through FrontController Servlet with out taking the support of View comps And ViewResolvers.

Rest

Sample Rest Controller (legacy style) (Restful App's Provider/Server/Producer App)

```
@Controller
@ResponseBody
@RestController
@RequestMapping("/message") // global path
public class Wish MessageRendererController{
@GetMapping("/wish") // path specific to each b.method
public ResponseEntity<String/<object>/<collection> showMessage(){ ResponseEntity<T>
operation
...
b.logic ordelegation
...
return ResponseEntity object;
} //method
}//class
```

Sample RestController (modren style)

```
@RestController
@RequestMapping("/message") // global path
public class MessageRenderController{
@GetMapping("/wish") --> request path
public ResponseEntity<String/Collection/Array> showMessage(){
b.method or
rest operation or
...
```

b.logic or delegation logic

rest end point method or rest handler method

return ResponseEntity object;

@XxxMapping anntoations are

@GetMapping -->for select operations @PostMapping for insert operations @PutMapping --> for update operations @DeleteMapping --> for delete operations @PatchMapping --> for partial update operations

ResponseEntity object contains two parts

a) Response content/body represents the results/output) b) Response Status code (100 - 599 numbers (or)

contstants of ResponseStatus enum like HttpStatus.OK -- 200) of the

Becoz of @ResponseBody that is added on the top method .. the method becomes b.method of server/provider/publusher comp of Restfull webService and becomes ready to send output/response directly to client/cosumer app through Front Controller Servlet

=> The return of type b.method in server/provider/Publisher App (@RestController class or @Controller+ @ResponseBody class)

is ResponseEntity<T> where <T> represents Generic Type ..

-> if the <T> is String .. then the response content/body or output/results goes to client/cosumer app as plain text

-> if the <T> is object or collection or array the response content/body or output/results will be converted into JSON key values and will be given to DispatcherServlet (Frontcontroller)

There are no annotations for

TRACE, HEAD mode/methods of request becoz these modes/methods of requests are not pratically executable in Rest API

note:: To convert into xml contnet.. we need to add special apis/libaries to classpath. (By default <T> content (other than String) will be converted into JSON Content keys and values becoz jackson api comes automatically to the Project once we add spring mvc web starters)

note:: There is no seperate spring rest starter spring web starter itself acts as the spring mvc starter.. and spring rest starter

} }

API = Service API = Server Provider = provider App= Producer App It is ultimately =Server App= Publisher App

@Controller+@ResponseBody class or @RestController class

In the @Controller class, if the method contains @ResponseBody + @XxxMapping + retutrn type as ResponseEntity<T> then the controller becomes server/producer/publisher App of Spring Rest style Restfull webservice App. (Distributed App)

In @Controller class, if the method contains @XxxMapping (only @GetMapping, @PostMapping pssible) and does not contain @ResponseBody.. more over the return of the method is other than ResponseEntity<T> then that method actshandler method of Spring MVC controller class and this method takes the support of view comp and View Resolver to send response to the browser through Frontcontroller Servlet. (web application)

=>Simple @Controller class handler methods

of MVC web application can deal with only GET, POST requests becoz they allow only browser as the client and browser can send only GET,POST MODE request s

Limitation with

web application

=>The @RestController class handler methods /operations of Restful application can deal with multiple modes

GET, POST,PUT, DELETE, HEAD, PATCH, OPTIONS, TRACE and etc.. advantage with

of requests becoz they allow different types of Clients or consumer Apps and these apps can send different modes of requests.

Distributed App

=(API):

Procedure to develop first Spring Rest server/producer/publisher app as @RestController Comp and testing that comp using browser

=========

step1) make sure that following software setup is avaiable

=>eclipse JEE IDE with STS plugin (2020+)

=========a

=> Tomcat10x server (This is optional if ur planning to use spring boot suppled embedded Tomcat) => jdk 1.8+

10.x

step2) make sure that Tomcat server is configure with Eclipse IDE

a) complete the installation Tomcat

b) window menu ---> preferences ---> servers --- Run time env.. --->

add --> select apache tomcat -->

Name:

Apache Tomcat 10.x

Tomcat installation directory:

E:\Tomcat 10.x

JRE:

apache 10.x

Workbench default JRE

--> apply --->ok

Go to servers tab ---> click add server link ---> select apache tomcat 10...>.... step3) create spring boot stater Project adding "Spring web Starter", "devtools"

File menu ---> new Project ---> spring boot ---> spring starter project ---->

development

API development is web services nothing but developing the @RestController class or @Controller +@ResponseBody class

note:: In other parts of JAva API means

ages with classes, interfaces, enums and annotations eg:: servlet api,jdbc api and etc..

note:: In Webservices,the API development means we are developing @RestController class

(provider/Publisher/Server provider comp)

In Java every API is RestController class linked with FrontController servlet comp

note: any resusable class or file is called component

Service URL

https://start.spring.io

Name

SpringRestProj01-FirstProviderApp

Use default location

Location

G:\Worskpaces\Spring\NTSPBMS615-Rest-MS\SpringRestProj01- Browse

Type:

Maven

✓ Packaging:

Java Version:

17

✓ Language:

War Java

Group

nit

Artifact

Version

Description

SpringRestProj01-FirstProviderApp

0.0.1-SNAPSHOT

Demo project for Spring Boot

Package

com.nt

Working sets

Add project to working sets

Working sets:

New... Select...

--> next ---> select spring web, devtools startters ---> finish

step4) Develop the following @Restcontroller class as server/producer/publisher app in com.nt.controller package.

//MessageRenderController.java

```
package com.nt.controller;
import java.time.LocalDateTime;
```

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/messageapi") // global path
public class MessageRenderController {
@GetMapping("/wish") // method path
public ResponseEntity<String> showMessage(){
// get System Date and time
LocalDateTime ldt=LocalDateTime.now();
// Generate Wish Message
String msg=null;
int hour=ldt.getHour();
if(hour<12)
msg="Good Morning";
else if(hour<16)
else if(hour<20)
}
msg="Good Afternoon::";
msg="Good Eveing::";
else
msg="Good Night";
return
// create and return Response Entity object having response contnet and status code
ResponseEntity<String> entity=new ResponseEntity<String>(msg,HttpStatus.OK); //(body, status) entity;
code
}
```

step5) provide embedded tomcat server port number in application.properties

application.properties

```
#Embedded server port
server.port=4041
# Application context path while running embedded Server
server.servlet.context-path=/RestApp01
```

step6) Run the App using External Tomcat

Right click on the Project ---> run as ---> run on server ---> select Tomcat 10

-->next --> ....

Test the application using browser (becoz browse can send GET,POST mode requests)

http://localhost:2020/SpringRestProj01-FirstProviderApp/messageapi/wish

25

note:: while using external Tomcat server 10 version Change we must the dynamic web module version from 6.0 to 5.0

(In browser address bar)

SpringRestProj01-FirstProviderApp [boot] [devtools] Deployment Descriptor: SpringRestProj01-FirstPrc Spring Elements

>JAX-WS Web Services

#src/main/java

Tomcat 10.1 supports Dynamic web module version 6.0 Tomcat 10 supports Dynamic web module version 5.0

Right click on project ---> properties ---> Project facets---> Dynamic web mobule 5.0

In External server/Tomcat

The project name is context path

>

>

step7) Run the App using Embedded Tomcat server..

==>Right click on Project ---> run as---> spring boot App ...

Test the application in browser

com.nt

>ServletInitializer.java

> SpringRestProj01 FirstProviderAppApplicati

com.nt.controller

>MessageRenderController.java

src/main/resources

static

global path

templates

application.properties

http://localhost:4041/RestApp01/messageapi/wish

context path

given in

application.properties

>src/test/java

> JRE System Library [JavaSE-16]

method/operation path

>

Maven Dependencies

>

Deployed Resources

> src

=> Browser sends GET mode request in the following situations

a) using url typed in the browser address bar

b) using hyperlink

c) using <form method="GET">

d) using <form> (default is GET mode)

> target

WHELP.md

mvnw

mvnw.cmd

Mpom.xml

note:: while using Tomcat server as the external server in eclipse IDE

a) make sure that it is not already started becoz of the default "automatic" startup type

(Use services.msc Tomcat server properites)

b) make sure that port number changed to other than 8080 (use server.xml file in the servers section of Project explorer)

Test the app using browser or POSTMAN tool

note:: since ResponseEntity<String> is taken and handler method type is "GET" (in this controller)

Using browser

==========

we can send the request even from browser (otherwise not possible from browser)

(Not recomanded to use becoz it can send only GET or POST mode requests

where as @RestController can have GET,POST, DELETE,PUT,PATCH and etc.. modes handler methods)

a) Run the Application using Run as server option

or Rest operations

b) use the following from the browser

http://localhost:3131/Boot RestProj01-FirstApp/message-api/wish

Using POSTMAN Tool (Recomanded to use)

(A good tool to Test the Resftull web services /service providers developed

to

(It is readymade simulator for consumer app which can send different modes of request)s

in any language/technology /framework having capability generate different modes of request and ability recieve text/JSON/XML content based response)

=> Using POST MAN tool, we can save the tests and we can re-run tests

a) download and install postman tool

(by skipping the registration)

https://www.postman.com/downloads/ ----> use windows 32/64 bit for download

b) Create a new Collection

=> Postman home page ===> skip registration and go to the app (look at botton the page)===> use (+) symbol to create the collection.

c) Send request by using the request url

Right click on the above created Collection --> Add Request --->

be

The rest APIS or Provider Apps can tested

in two ways in java env...

a) Postman (generic tool for all)

b) Swagger (apfafiven testing) (best)

GET

(a)

http://localhost:3131/Boot Rest Proj01-FirstApp/message- (b) Enter URL

api/wish

J

Params

Authorization

Headers (6)

Body Pre-request Script Tests Settings

Query Params

KEY

Key

VALUE

Value

Body Cookies

Headers (5)

Test Results

Pretty

Raw Preview

Visualize

Text

```
1 Good Morning
```

response

(d)

DESCRIP

Descripti

SEND

=>In Postman, we take

(c)

one collection for each RestAPI Testing

In that collection, we add multiple requests

to test multiple operations/methods of RestAPI

endpoints

Benifits with POSTMAN style testing

(a) Allows to save and track the requests-response testing

A

200 OK

(b) Allows to give different modes of requests (GET, POST, DELETE, PUT, PATCH,

(c) allows to set our choice values as request header values

What is the difference between @Controller and @ResstController ?

@Controller

@RestController

OPTIONS

(allows to set different types of content like text content,XML content, JSON content as the

request body/payload

(Allows to view/see response header values

and etc..

() we can get response in different formats (Text/XML/JSON/....)

note:: Using Browser we can send only GET,POST mode requests where as using POSTMAN tool we can send GET,POST,HEAD, PUT,DELETE,PATCH,OPTIONS mode requests (TRACE is not possible from POSTMAN)

a) Given in the spring 2.5 version

a) Given in the version spring 4.0 (relatively new annotation)

(bit old annotation)

b) Specialization@Component

(it is @Component++)

c) makes the java class webController

class having capability to handle

http requests by taking them through DispatcherServlet

d) Can be used in both spring MVC

and spring Rest Apps (Even in Spring boot MVC

b) Specialization of @Controller (It is @Controller++)

(@RestController= @Controller + @ResponseBody)

as

c) makes the java class Rest Controller or Restfull service provider or Rest API

or API

class having capability to handle http requests by taking them through DispatcherServlet

d) Recomanded to use only in spring Rest Apps (Restful Apps)

and spring boot rest Apps)

e) Every Handler method does not

get @ResponseBody automtically, So

we need to add it expicitly if needed

of the

f) Based on the return type handler

method it decides wheather view comp

should be invovled or not

(if the return type is otherthan ResponseEntity<T> then it involves ViewResolver and view comp)

(g) Here methods are calle handler metthods

(Also in Spring Boot Rest Apps)

(end point method)

e) Every handle method automatically gets @ReponseBody

So there is no need adding it seperately

of

(endpoint)

f) Makes the handler method sending its

output/results as response to client

direcly through DispagcherServlet

withvolving View Resolver and View comp

(@Becoz of @ResponseBody)

(g) Here methods are called Rest Opeations or Endpoints or Rest Endpoints

note:: Instead of comparing @RestController with @Controller .. plese use

it as convient annotation given over @Controller providing easyness to develop

Restfull Service providers (@Contrller + @ResponseBody)

Http request methods/modes

==========

GET

POST

(total 9

PUT

but 8 are

DELETE

OPTIONS

in operation)

TRACE

PATCH

HEAD

CONNECT (Resrved for Future)

or APIs

=>Spring/spring boot MVC Apps are front end apps having UI logics other logics

=> spring/spring boot Rest Apps distributed apps and these are always Back End Apps

Generally we use the following Http request

methods/modes in Restful applications while performing

CURD Operations throuhg service,DAO classes from Rest operations (The b.methods/handler methods /RestOperation methods /endpoints

GET ---> for Read Operations (R) ->selecting records

POST ---> for Create opertions (C) -> inserting records

DELETE ---> for DELETE opertions (D) -> deleting records

of @RestController class)

PUT ---> for Update opertions (U) -> modifying records (full modifcation of the records)

PATCH ---> for Update opertions (U) -> modifying records (partial modification of the records)

note: PUT for complete modification of the record. (eg:: modify every info in aadhar card based on aadhar no)

PATCH for partial modification of the record. (eg:: modify only phoneNumber in aadhar card based on aadhar no) @PutMapping("/....) public <RT> updateEmployee (Employee emp){

}

use

PUT

mode request (Complete Employee record modification)

we can add the following @XxxMapping annotations on the Rest Operation methods of the RestController class

genrerted

@PatchMapping("/....)

public <RT> updateEmployee Email(String newEmail,int empno)

use PATCH (only partial mode request modification)

=>Since HEAD request mode HttpResponse does not contain body/output/results, So it can not be used for CURD operations

for Read /Select Opertions (Useless in restfull webservices)

=> Since TRACE is given to trace/debug componets involved for the SUCCESS or FAILURE of request processing

it can not be used for CURD Operations (useless in restful web services)

=>Since _OPTIONS_mode request gives the possible Http request methods/modes that can be used to generate request to web comp.. its HttpResponse contains purely list Modes/methods that are possible to give request

to web comp.. It also can not be used for CURD operations (useless in restful web services)

@GetMapping --> on the endpoint that performs select operations @PostMapping -->--> on the endpoint that performs INSERT operations @DeleteMapping--> on the endpoint that performs Delete operations @PutMapping --> on the endpoint that performs full object update operations @PatchMapping --> on the endpoint that performs Partial Object update operations

HEAD, TRACE and OPTIONS mode/method request related responses

do not contain body/payload .. so they are so useless while performing CURD Operations in Restful web services

or Rest API comp

Servlet,jsp, html files comps are called web comps

@WebServlet("/testurl")

public class TestServlet{

if we give OPTIONS mode request to this

public void doGet(req, res){

web comp we get response as

Allow: GET,HEAD,OPTIONS

}

}

(as response header)

}

@WebServlet("/testurl")

public class TestServlet{

if we give OPTIONS mode request to this

public void dopostreq,res){

@RestController

@RequestMapping("/message")

public class Wish MessageOperationsController{ @GetMapping("/wish") public ResponseEntity showMessage(){

<String>

if we give OPTIONS mode request to API/Restcontrloller having the URL (http://...../message/wish) then we get the reponse like

allow:: GET,HEAD,OPTIONS

(response header)

note1:: Request headers carry additional data given by browser /consumer app along with the request

->contentType, contentLength, referrer, user-agent, accept, accept-language,.....

by

note2: Response headers carry additional data given server /web container along with the response

-> refresh, contentType, contentLength, connection, content-disposition, date,server and etc..

web comp we get response as

Allow: POST,OPTIONS

we can make the methods of @RestController class (API) handling different modes of requests

with specified request paths with the support of the following annotations

}

@GetMapping

}

@PostMapping

@DeleteMapping

@PutMapping

@PatchMapping

note:: GET MODE Request related response

contians response

body representing the results

=>if the request url is not valid then we get 404 error (requested resource not found)

note:: HEAD MODE Request related response

does not contain response

=> if the request url based web comp is not ready

body representing the results

to process given modefequest then we get 405 error (method not allowed)

=> if the request fails in Authentication then we get 401 error

=> if the request fails in Authorization then we get 403 error (resource is forbidden)

=> if the web comp(servlet/jsp/producer) fails to instantiate for the given

request then we get 500 error

(end pojnt)

=>if the return type of producure method is other than <String> generic in ResponseEntity object

then the producer methods send JSON data along with the HttpResponse body

=>if the return type of producer method is <String> generic in ResponseEntity object

then the producer methods send text data along with the HttpResponse body

Every Restfull application /Project contains

as

a) server App/prodcuder App/ Service provider App /API (spring MVC App with @RestController with methods)

(also called Rest API)

(are called rest

[The request url of @RestController and other required infomation for operations) sending request are called

End points]

OPTIONS

note:: BOTH

HEAD mode requests does not contain body and their generated responses also does not contain body

note:: GET mode request does not contain BODY

but the related response contains BODY

note: POST mode request and its related response both contains BODY

Http response status codes

=> 100-599 are http response status code

b) Client App /Service Cosumer/ Consumer App

App

Angular

ReactJS

PHP

IOS

(Programable

IOT Devices

client Apps)

request headers vs req pameters

.net Java

python

andriod

POSTMAN Tools for Swagger Testing spring RestTemplate (spring basedClient)

other REST API comp

(Programmable Client Apps)

100-199

Information

200-299 :: Success

300-399 :: Redirection

400-499

Client Side error

500-599: Server Side Error

(browser/consumer)

=>req headers are Client generated

inputs that go along with request

contains more info about

client given by the client (browser or consumer app)

eg:: accept, accept-language, user-agent,referer,contentType and etc.

automatically having fixed names (header names)

=>req params are enduser supplied values

are

as query String /form data ..req param names not fixed ..

and they are user-defined (In other than GET/HEAD mode request they act as request body/payload)

?sno=101&sname=raja&sadd=hyd

request param values

req params/query params

names

==>

API development means Developing Spring Rest Server App/Service provider App (In spring/spring boot, It is going to be @RestController class development)

=> Giving API End points means providing request url and other related information to

developers for developing client Apps/service consumer App for cosuming the services offered by Service provider

note:: The API/ services developed in SOAP based webservices can not consumed

using Rest Client and vice-versa.

(end points)

note: The Restful webServices developed using one kind of Rest API/framework (like jesry /spring rest, ja x-RS,Restlet and etc..)

can be consumed using same Rest API or different Rest APIS (becoz both are in Restfull webService env..)

(public apis)

can be

eg:: There are multiple open/free apis /service providers devleoped in different technologes/frameworks

of Restfull webservices and they consumed in our Apps using our choice rest apis..

eg:: weather report api

API/RestController Development having

different @XxxMapping annotations based

b.methods /Rest Operations

Google

Maps api

Gpay APIs

ICC API

Covid APIs

Payment Gateway APIS

RestController class

@RestController class = API

Methods in @Restcontroller class are called

API Endpoints

SpringBootRestProj02-Different MethodsPOC [boot]

> Deployment Descriptor: SpringBootRestProj02-DifferentMethodsPOC

>Spring Elements

> JAX-WS Web Services

src/main/java

> # com.nt

com.nt.controller

› CustomerOperationsController.java

src/main/resources

>

>

> JRE System Library [JavaSE-11]

src/test/java

› Maven Dependencies

Deployed Resources

> src

› target

WHELP.md

mvnw

mvnw.cmd

M pom.xml

HEAD,TRACE,OPTIONS mode requests are useful

in the apps development using which we can

monitor the websites and APIs of web services

```
package com.nt.controller;

import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import
org.springframework.web.bind.annotation.DeleteMapping; import
org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.PatchMapping; import
org.springframework.web.bind.annotation.PostMapping; import
org.springframework.web.bind.annotation.PutMapping; import
org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/customer")

public class CustomerOperationsController {

@GetMapping("/report")
```

```
public ResponseEntity<String> showCustomersReport(){
return new ResponseEntity<String>("From GET-ShowReport Method", HttpStatus.OK);
```

In One Rest API or Rest Controller we can place

any no.of EndPoints having either same request

modes or different request modes

```
@PostMapping("/register")
public ResponseEntity<String> registerCustomer(){
return new ResponseEntity<String>("From POST-RegisterCustomer Method", HttpStatus.OK);
}
@PutMapping("/modify")
public ResponseEntity<String> updateCustomer(){
return new ResponseEntity<String>("From PUT-UpdateCustomer() Method", HttpStatus.OK);
}
@PatchMapping("/pmodify")
public ResponseEntity<String> updateCustomerByNo(){
return new ResponseEntity<String>("From PATH-UpdateCustomerByNo() Method", HttpStatus.OK);
}
@DeleteMapping("/delete")
public ResponseEntity<String> deleteCustomer(){
return new ResponseEntity<String>("From DELETE-deleteCustomer Method", HttpStatus.OK);
}
```

GET http://localhost:3030/Spring BootRestProj02-DifferentMethods POC/customer/report

Send

POST

http://localhost:3030/SpringBoot RestProj02-DifferentMethodsPOC/customer/register

Send

DELETE http://localhost:3030/Spring Boot RestProj02-DifferentMethods POC/customer/delete PUT http://localhost:3030/SpringBootRestProj02-DifferentMethods POC/customer/modify PATCH http://localhost:3030/SpringBootRestProj02-DifferentMethods POC/customer/pmodify

Send <

Send

Send

=>when we deploy the Restful service provider app or web application in the external server like Tomcat then then the name of the project becomes context path automatically..

=> when we deploy the same Restful service provider app in the Embeded tomcat server then

the the apps runs with out context path...by default.. To provide context path to that app

take the support an entry in application.properties file

#Context path of the application server.servlet.context-path=/SecondProviderApp

Q) Can we inter change the request modes on the endpoint methods(@XxxMApping methods) of RestAPI(@RestController)

Ans) yes we can change, but not recommended becoz it kills the redability of the REST API and gives problems in the

development of

Rest Consumer App