Developing

Consumer App using RestTemplate

for

=> It allows to develop the consumer/Client App RestFull webService as Programmable Client App in java env..

(provider app)

=> We need to take seperate WebService/MVC Project for this having logics to consume webService /API by calling methods.

=> This object (RestTemplate) does not come through AutoConfiguration Process.. It must be created either using "new" operator or using @Bean method of @Configuration class

eg::

RestTemplate template=new RestTemplate();

(or)

In @Configuration class

@Bean("template")

public RestTemplate createTemplate(){

}

return new RestTemplate();

(RestTemplate)

of

note:: So far we have used POSTMAN as the Client/cousumer which is a tool.. if u want to develop the real stadalone app or web application as the consumer App then we need to use this RestTemplate class. note: JdbcTemplate, RestTemplate, JndiTemplate, NamedParameterJdbcTemplate and etc.. classes are given based template method design pattern which says that the template class takes care of common logics and the programmer should take care of specific logics.

=> This object provide methods to generate different modes requests like GET/POST/PUT/DELETE/.... to consume the Restfull webService /API.. i.e we can call methods /operations Restful webService Server App/provider App from the consumer app using this RestTemplate

n

to pass

=> While using this object to cosume RestFull WebService/API we need detailed inputs (nothing but end points) like base url, http method type, http header info like content type and etc..

n

=> It estampaxxxForEntity(....) methods like getForEntity(...), postForEntity(...) and etc.. taking url, request obj(body,header) to send different modesfhttp requests as method calls to cosume the the Restfull web service (Server/Provider App)

=> Do not forget WebService is given to link two different Apps that are developed either in same language or in diferent languages and

to same machine or different machines.

Restfull WebService

running same server or different servers belonging

[Server App/Provider App) Http request (b)

operations/b.methods{

(c)

...

b.logics

..

}

network

(d) http response

Restful WebService

[Client App/consumer App)

...

(a) [request generation]

logic to consume

the services of server App/API

(e) [response gathering]

on

Consumer App for Rest API can be done in two ways

a) Using RestTemplate (old and legacy)

b) Using WebClient (latest) (best)

The company who develops the Restful API/ provider app uses the POSTMAN as the simulator to Consumer App for testing Restuful API eg: Paypal company who developed the Payment Broker App

as restfull api, tests the API using POSTMAN tool/Swagger tool

eg:: VISA company who developed the Payment gateway API as the restful api, tests the API using POSTMAN tool/swagger tool note: POSTMAN can be used only as the Tool test the API where as swagger can be used to test the API and to provide the documentation about the API (end points details)

=> The Companies who wants to consume APIs in thier Apps they take the support of RestTemplate for consumption if the App is there in spring/spring boot env.. (java env..)

eg1:: Flipkart.com uses RestTemplate support to

consume the servcies of paypal API

eg2:: Paypal(API and consumer) uses RestTemplate support to consume the servcies of VISA API (Payment Gateway API)

=>JAva App acting as the client wants to consume Rest API services use RestTemplate

=> One Rest service who wants to consume another Rest Service also uses the RestTemplate

Client/Cosumer App

Server/Provider App

PayTM

PaymentGateway <--

App

PayTM/GooglePay/PhonePay<--

WeatherReport App <-

ICC Score Comp

(Rest API to Rest API)

Flipkart/Amazon (Rest API to Client) yahoo.com/Tourist.com (Rest API to client) CrickInfo.com, CrickBuzz.com and etc.. (Rest API to client)

One API /provider App of Restful env..

can consume another API/provider services web application

Provider App)

browser->flipkart.com -----> UPI Payment ------> BankApp

=>The RestFullWebService (Server/ provider App) must be the web application beloging to different languages

(Consumer)

=> The consumer App can be the standalone App or mobile App or IOT App or Web application or etc.. beloging to different languages

ed

So far we have devleop only Restful WebService(Server/provider App) and we tested that server App using tools like POSTMAN/Swagger ... Instead of using these tools we can develop programmable Real client Apps with the support RestTemplate in spring Env.. /spring boot env..

=>RestTemplate can be used only in Spring or Spring Boot env.. (not in other java api of restful webservices)

Example App

============

(spring Rest/Spring boot Rest)

(provider App)

step1) Develop Restful WebService App as web application (Server /provider App)

(old style App)

staters :: web, dev tools, lombok api

step2) Develop API/Rest controller

```
package com.nt.controller;

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("/actor")

public class ActorOperationsController {
```

|-->like jax-rs,resteasy, jersy and etc..

✓5 SpringBootRestProj13-ProviderApp [boot] [devtools]

>. Deployment Descriptor: SpringBoot RestProj13-ProviderApp >Spring Elements

> JAX-WS Web Services

#src/main/java

>com.nt

com.nt.controller

> ActorOperationsController.java

> #src/main/resources

src/test/java

Maven Dependencies

>

>

JRE System Library [JavaSE-11]

>

>

Deployed Resources

>

src

> target

WHELP.md

mvnw

mvnw.cmd Mpom.xml

(Consumer to BankApp & Provider for Flipkart)

API Development

Provider App in Spring/Spring boot env (@RestController)

--> Consumer App in Spring/Spring boot

[RestTemplate ------> spring/spring boot web/rest] (Normal app) [WebClient > spring /spring boot webflux] (reactive Programming)

The Restful webservices /RestAPI can have the following types of client apps/consumer apps

a)standalone apps

b) desktop apps

c) web applications / websites

d) mobile apps

e) IOT Apps

f) Embedded System Apps

d) another RestFull API

and etc..

@GetMapping("/wish")

public ResponseEntity<String> displayWishMessage(){

return new ResponseEntity<String>("Good Morning",HttpStatus.OK);

}

}

step3) Run The application...on server (Run As --->Run on SErver)

(Consumer App using RestTemplate)

step4) Develop the Consumer App as seperte Project

(starters :: web, dev tools, lombok)

(mandatory) (optional)

step5) place the following entries in application. properties

server.port=4040

step6) Develop the Runner App

package com.mc.runner,

import org.springframework.boot.CommandLineRunner; import org.springframework.http.ResponseEntity;

✓ M5 SpringBootRestProj13-ConsumerApp [boot] [devtools]

>

Deployment Descriptor: SpringBootRestProj13-ConsumerApp >Spring Elements

>JAX-WS Web Services

#src/main/java

>com.nt

com.nt.runner

> ActorSeviceConsumingRunner.java

#src/main/resources

static

templates

application.properties

>

src/test/java

>

JRE System Library [JavaSE-11]

>

Maven Dependencies

> L Deployed Resources

> src

import org.springframework.stereotype.Component;

import org.springframework.web.client.RestTemplate;

@Component

public class ActorSeviceConsumingRunner implements CommandLineRunner {

@Override

public void run(String... args) throws Exception {

//create RestTemplate class object

RestTemplate template=new RestTemplate();

//Define service url

> target

WHELP.md

mvnw

mvnw.cmd

nom yml

String serviceUrl="http://localhost:3030/SpringBoot Rest Proj13-ProviderApp/actor/wish";

// Generate Http reqeust with GET mode to consume the web service(API)

ResponseEntity<String> response=template.getForEntity(serviceUrl, String.class); //display the recieved details from the response

result class

System.out.println("Response body (output) ::"+response.getBody()); "Container Good Morning message" System.out.println("Response status code value ::"+response.getStatusCodeValue()); System.out.println("Response satus code ::"+response.getStatusCode().name());

//System.exit(0); //optional

}

}

note:: Here we do not need any JSON/XML conversion APIs becoz the

the provider is sending only text content as the response content and it comes onsumer app directly as the text content .. So no conversions are required in Consumer App

step4) Run Consumer App as spring boot App that uses Embedeed Tomcat server.. Run As ----> spring Boot App/ jav aApp

for provider App development)

(Becoz external is already used

(Uses Embedded Server) ed

note:: The above consumer App can also be develop as standalone app (package type is jar) adding spring web starters.. as shown below

✓ M5 SpringBootRestProj13-ConsumerApp-Standalone [boot]

>Spring Elements

>

#src/main/java

✓ com.nt

>SpringBootRestProj13ConsumerAppStandaloneApplication.java

com.nt.runner

> ActorSeviceConsumingRunner.java

#src/main/resources

static

templates

application.properties

src/test/java

> JRE System Library [JavaSE-11]

> Maven Dependencies

> src

› target

w HELP.md

mvnw

mvnw.cmd

Mpom.xml

Runner class code

and application.properties content is same as the above Consumer App..

[Uses the Embeded Tomcat server]

Q) Why the RestTemplate class obj is not coming as spring Bean in AutoConfiguration process?

Ans) The setup required for the sprng boot based rest webservice provider app and consumer

App remains same and do not need RestTemplate class object in Proiver App .. if the RestTemplate class obj is comming as spring bean through AutoConfiguration then it will be wasted. More fportanly we use

same setup even to develop spring boot mvc apps.. if RestTemplate class obj is give though AutoConfiguration it will be purely wasted in web applications. and provider Aps

Q) what is the difference between xxxForEntity() and xxxForObject() methods? (or)

Q) what is the difference between getForEntity() and getForObject() methods?

Ans) getForEntity() /xxxForEntity() methods return ResponseEntity<T> object which contains all the details recived response like response body (result), response headers, response status code and etc...

getForObject() /xxxForObject() methods return <T> object which

contains only response body(result)

if the recieved JSON content is having date and time values those values

can be mapped with java 8 LocalDate,LocalTime, LocalDateTime class objs

only after performing the following opeations

a) add the additional jar file

<groupId>com.fasterxml.jackson.datatype</groupId> <artifactId>jackson-datatype-jsr310</artifactId>

<version>2.15.0</version> <!-- Use the latest version -->

</dependency>

b) register the module with ObjectMapper

ObjectMapper mapper-new ObjectMapper(); mapper.registerModule(new JavaTimeModule());

eg1::

/create TestTemplate class object

RestTemplate template =new RestTemplate(); //prepare base url

String baseUrl="http://localhost:4040/Boot Rest Proj12-ActorService-API/actor/wish";

// invoke the Service/Operation of Provider App

ResponseEntity<String> response-template.getForEntity(baseUrl,String.class);

System.out.println(" response body ::"+response.getBody());

System.out.println("response status code ::"+response.getStatusCode());

eg2:

//create TestTemplate class object

RestTemplate template =new RestTemplate();

//prepare base url

String baseUrl="http://localhost:4040/Boot Rest Proj12-ActorService-API/actor/wish";

// invoke the Service/Operation of Provider App

String result=template.getForObject(baseUrl, String.class);

System.out.println("result ::"+result);

When should i create RestTemplate class obj manually in the Consumer App and when should i go for @Bean method

based object creation?

Ans) =>if RestTemplate class obj is required in multiple parts of consumer app then create it using @Bean method in @Configuration class, So that the same spring bean can be injected in multiple spring beans as needed..

@Bean

public RestTemplate createTemplate(){

return new RestTemplate();

if RestTemplate class obj is required in only in one place of Consumer App then create it directly using

new operator

eg: RestTemplate template=new RestTemplate();

main class

=========

Consumer App for

MiniProject using @XxxMapping annotations of RestTemplate

==========

=========

```
==============================
@SpringBootApplication
public class Boot Rest Proj12ConsumerAppApplication {
@Bean(name="template")
public RestTemplate createTemplate() {
return new RestTemplate();
}
public static void main(String[] args) {
SpringApplication.run(Boot RestProj12ConsumerAppApplication.class, args);
}
}
Runner1
=======
@Component
public class ShowAllActors Runner implements CommandLineRunner {
@Autowired
private RestTemplate template;
@Override
public void run(String... args) throws Exception {
//prepare baseURL
/*
//use getForEntity(-,-) mehtod
String serviceUrl="http://localhost:4041/actor-api/all";
}
}
ResponseEntity<String> response=template.getForEntity(serviceUrl, String.class);
//process the response
System.out.println("response body(result)::"+response.getBody()); System.out.println("response headers
::"+response.getHeaders());
System.out.println("response status code ::"+response.getStatusCode().value());*/
//iuse getForObject(---)
String result-template.getForObject(serviceUrl, String.class);
System.out.println(result);
Runner3
=======
@Component
public class SaveActor Runner implements CommandLineRunner {
```

```
@Autowired
private RestTemplate template;
@Override
public void run(String... args) throws Exception {
//prepare baseURL
String serviceUrl="http://localhost:4041/actor-api/save";
//prepare json body
//Http heders
String json_body="{\"aname\":\"Jr.ntr\", \"addrs\": \"hyd \", \"remuneration\": 854545.0, \"active_SW\":\"active\"}";
HttpHeaders headers-new org.springframework.http.HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);
//prepare HttpEntity obj having headers, body
HttpEntity<String> entity=new HttpEntity<String>(json_body, headers);
//use PostForEntity(-,-) mehtod
ResponseEntity<String> response-template.postForEntity(serviceUrl,entity,String.class);
//process the response
System.out.println("response body(result)::"+response.getBody());
System.out.println("response headers ::"+response.getHeaders());
System.out.println("response status code ::"+response.getStatusCode().value());
}
}
Runner4
========
@Component
public class UpdateActorRunner implements CommandLineRunner {
@Autowired
private RestTemplate template;
@Override
@Component
Runner2
=======
public class ShowActor Byld Runner implements CommandLineRunner {
@Autowired
private RestTemplate template;
@Override
public void run(String... args) throws Exception {
//prepare baseURL
```

```
String serviceUrl="http://localhost:4041/actor-api/find/{id}";
//use getForEntity(-,-) method
ResponseEntity<String> response-template.getForEntity(serviceUrl, String.class, 1002); //process the response
System.out.println("response body(result): :"+response.getBody());
System.out.println("response headers ::"+response.getHeaders());
System.out.println("response status code ::"+response.getStatusCode().value());
/*
//iuse getForObject(-.-)
String result-template.getForObject(serviceUrl, String.class);
System.out.println(result); */
System.exit(0);
}
}
public void run(String... args) throws Exception {
String serviceUrl="http://localhost:4041/actor-api/update";
//prepare baseURL
//prepare json body
String ison body="{\"aid\": 2002, \"aname\":\"Jr.ntr\", \"addrs\": \"mumbai \", \"remuneration\": 954545.0,
\"active_SW\":\"active\" }";
//Http heders
HttpHeaders headers=new org.springframework.http.HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);
//prepare HttpEntity obj having headers, body
HttpEntity<String> entity=new HttpEntity<String>(json_body, headers);
}
//Runner5
//use put(--) mehtod
template.put(serviceUrl,entity);
System.out.println("Actor updated");
System.exit(0);
@Component
public class UpdateActorRemuneration ByldRunner implements CommandLineRunner {
@Autowired
private RestTemplate template;
@Override
public void run(String... args) throws Exception {
//prepare baseURL
```

String serviceUrl="http://localhost:4041/actor-api/rupdate/{id}/{amount}";

template.setRequestFactory(new HttpComponentsClientHttpRequestFactory()); //use patchForObject(-,-) method

String result-template.patchForObject(serviceUrl,null, String.class,1002,45678901.0);

//process the response System.out.println("response body(result)::"+result);

System.exit(0);

}

}

Runner6

======

@Component

public class DeleteActorByIdRunner implements CommandLineRunner {

@Autowired

private RestTemplate template;

@Override

public void run(String... args) throws Exception {

//prepare baseURL

String serviceUrl="http://localhost:4041/actor-api/delete/{id}";

//template.setRequestFactory(new HttpComponentsClientHttpRequestFactory());

//use patchForObject(-,-) method

template.delete(serviceUrl,1002);

//process the response

System.out.println("Actor deleted");

System.exit(0);

}

While sending

patch

mode request add this extra dependency

<dependency>

<groupId>org.apache.httpcomponents.client5</groupId>

<artifactId>httpclient5</artifactId> <version>5.2.1</version>

</dependency>

Methods in RestTemplate for sending different modes of requests

======

======================

getForEnttiy (---) /getForObject(---) ----> for GET mode postForEnttiy (---) /postForObject(---) ----> for POST mode

put(---) ----> for PUT mode

patchForObject(---) ----> for PATCH mode

delete(---) ----> for DELETE mode