Repository https://github.com/natarazworld/NTSPBMS615Repo.git

https://github.com/natarazworld/NTSPBMS615Repo.git

Git Hub : https://www.youtube.com/watch?v=_frCuw5JWdQ&list=PLVlQHNRLflP-C43xWt10PB3gKC8DDPB12

Gradle : https://www.youtube.com/watch?v=5JkqrUu4kGM

https://www.youtube.com/watch?v=CfATtI9_nMA

https://www.youtube.com/watch?v=A12Z8LjJEMo

https://www.youtube.com/watch?v=jTJwieViWQQ&t=7074s

Microservices Introduction Videos: Spring Rest & Microservices

Day-1 https://youtu.be/Tlp45YhOE-8
Day-2 https://youtu.be/EpaNqWAVadI
Day-3 https://youtu.be/BWfLY15xj_w
Day-4 https://youtu.be/UQJ8ToGWN4Y
Day-5 https://youtu.be/05Kg91stFtU
Day-6 https://youtu.be/p9MJ-aSNlM0

Junit5, HttpUnit & Mockito Workshop Session-1 | by Mr. Nataraj

https://www.youtube.com/watch?v=6VVplfYp40M
https://www.youtube.com/watch?v=AI3b3FY2iD0
https://www.youtube.com/watch?v=7Xq4j0Q7aPE

===>Spring JDBC Videos ==========
https://youtu.be/4ccrm2cw7yw
https://youtu.be/ZLCtgHyl5ug
https://youtu.be/4J1Tw518YSA
https://youtu.be/ko2gvNZxhEc
https://youtu.be/n6E40xBGyZM
https://youtu.be/HtXBV4-7MIs
https://youtu.be/qwouM75mJWQ
https://youtu.be/mWnIcgnu9cE
https://youtu.be/A1R3Vmm5KVQ
https://youtu.be/xV4wfz85ZuQ
https://youtu.be/AKObOoiXtGc
https://youtu.be/9aExLuLkVKo
https://youtu.be/iyHrP0mn8co
https://youtu.be/UY-OOJL8XuY

spring Tx mgmt vedios
==========
https://youtu.be/tn7oXoXAKfI
https://youtu.be/68R_a2lmlII
https://youtu.be/nD_86olXt1I
https://youtu.be/5MNnR-EnJ6A
https://youtu.be/fePDtdnuQe0
https://youtu.be/e5chgAoS6pc
https://youtu.be/nMfEM1wTULg
https://youtu.be/dg9aXbc2MKk
https://youtu.be/-yH5cwcWjaA
https://youtu.be/8xxv4u-co2A
https://youtu.be/gdc_ICB7vE8
https://youtu.be/cgZ_tus_g8k
https://youtu.be/KwRjaI-EZeU

====Code Coverage tool =====
JaCoCo (Java Code Coverage) | by Mr. Nataraj
https://youtu.be/wc0eN73-2tM

====Junit5, HttpUnit & Mockito===
https://www.youtube.com/watch?v=6VVplfYp40M
https://www.youtube.com/watch?v=AI3b3FY2iD0
https://www.youtube.com/watch?v=7Xq4j0Q7aPE

Table of Contents

1. NTSPBMS615- spring Boot MVC Intro -- May 23rd- 2022

Course Titile :: Spring Rest and MicroServices

pre-requisites :: servlets, jsp ,spring boot, spring boot MVC + common sense

Duration :: 50 to 60 Sessions
Timings :: 7:30 pm to 8:45 /9 pm
Course fee :: 3000/-
Modules covered in this course
=>spring boot Rest
=> spring boot security
=> spring boot oauth
=> spring boot JWT
=> spring boot Messaging (JMS using Active MQ)
=> spring boot kafka RabitMQ
=> spring cloud (MicroServices) (only 25 to 30 Sessions)
=> MicroServices Desing Patterns (Important)
=> Misc topics
=> Realtime Java Tools
maven, gradle , GIT , Docker , Jenkins , Log4j, slf4j,postman, swagger, agile-JIRA and etc.. (15+ tools)

=> Multiple Mini Projects (3+) will covered by applying
->relatime coding standards
-> Design patterns
-> Tools
-> Processes
-> Unit Testing
and etc..

Spring Rest (Spring MVC ++) | Spring Boot Rest (Spring Boot MVC++)

note:: officially there is no module called spring Rest in spring framework.. It is extension of Spring MVC module to develop RestFull Apps as Distributed Apps.

=> The module of spring can be programmed using spirng framework or using spring boot framework .. (Spring boot is latest)

note: if MVC module is used in spring style then it is called spring MVC module
if the same MVC is used in spring boot style then it is called spring boot MVC

note: Since spring style progamming is becoming old and outdated.. it better learn all the modules of spring in spring boot style.

Spring boot= spring ++

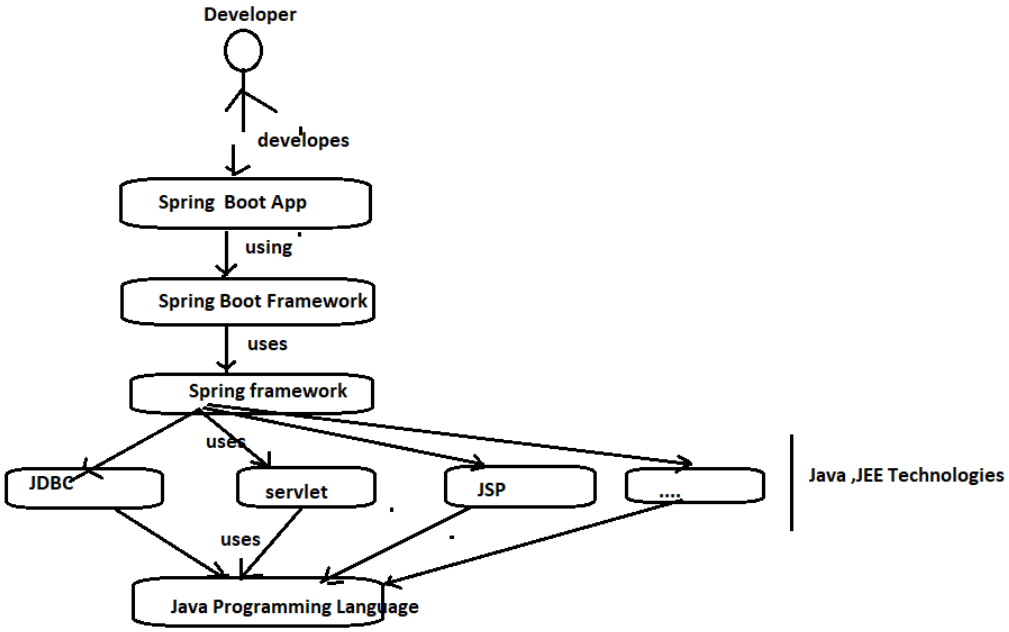
=>spring framework provides abstraction on JAVA ,JEE Technologies and simplifies the application devleopment process.
abstraction:: Hiding unneccessary detials .. and showing onlu necessary details
=>spring boot framework provides abstraction on spring framework to simplifies the application devleopment process more further,.

=> Clothes washing using hand (manual process) :: Like working with Java, JEE Technologies directly
=> Clothes washing using semi automated washing mahcine:: Like working with spring framework
=> Clothes washing using full automated washing mahcine:: Like working with spring boot framework

Spring boot = spring + AutoConfiuration + Embeded Servers / DBs

(Based on the libaries/
jar files we add to
classpath/build path
certain java classes automatically
becomes spring beans and
cetains Injections takes place
automatically)

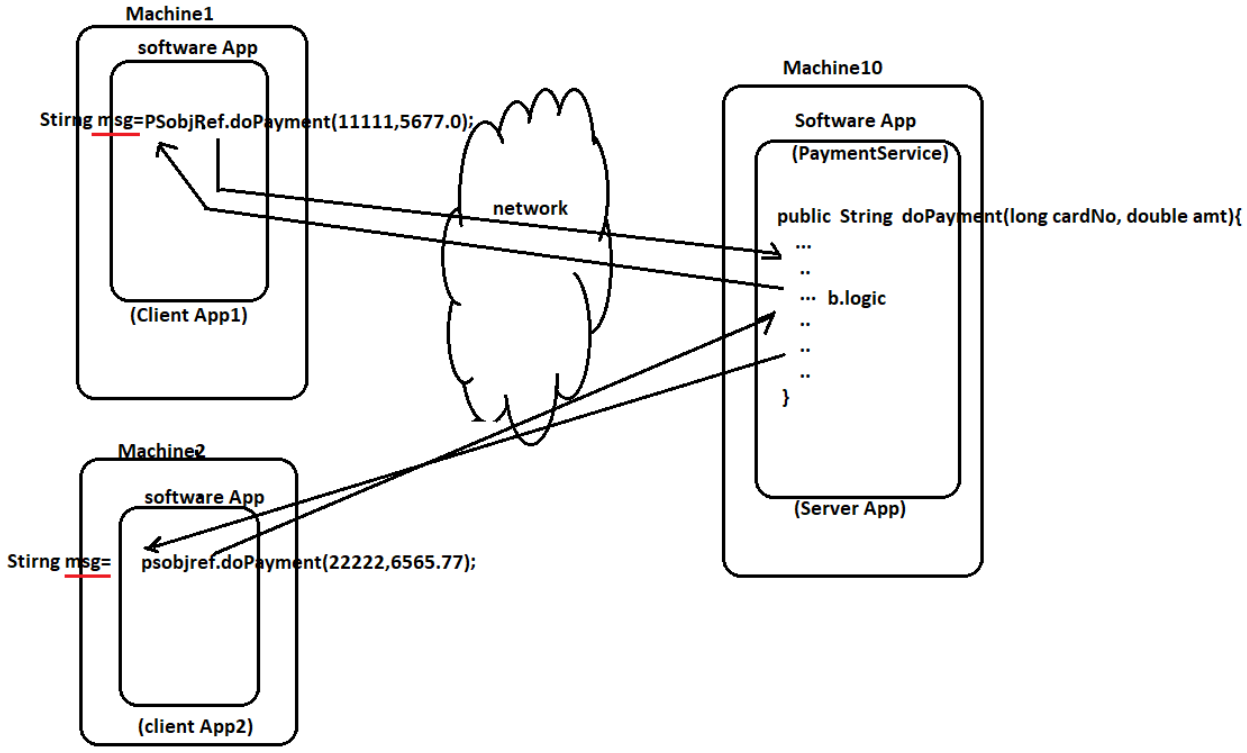
note: spring boot App can developed and tested with out installing extenal servers and extenal Db s/ws..



=>spring Rest or spring boot Rest is given to develop Distributed Applications.

Distributed App
==========
is
=>It Client - Server App where Both Client And server Apps are the software Apps interacting with each other .

=> Distributed App development or Distributed Programming speaks about App to App or Server to Server Communication.



eg1: Flipkart.com interaction with paypall
amazon.com interaction with paypall

note:: paypal is payment broker who interacts with PaymentGateway Apps like VISA/Master/Mastero and etc..

eg2:: Paypal interaction with VISA App
Paypal interaction with master App
paypal interaction with Rupay App

note:: The Server App distributed App can have different types of Local Clients and Remote clients
The possible Client types are :: Mobile Apps , Web Application, Desktop Apps , IVR Apps , IOT Apps and etc..

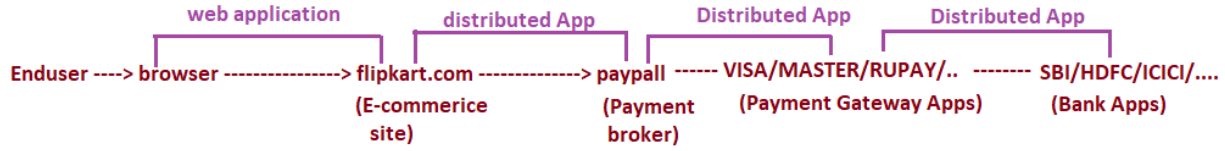
IVR Apps :: Interactive Voice Response Systems (Giving instructions to Server App using Phone key pad)

eg:: Gas cylindar booking
eg:: Indian Railway PNR Status
eg:: Bank Customer Care operations

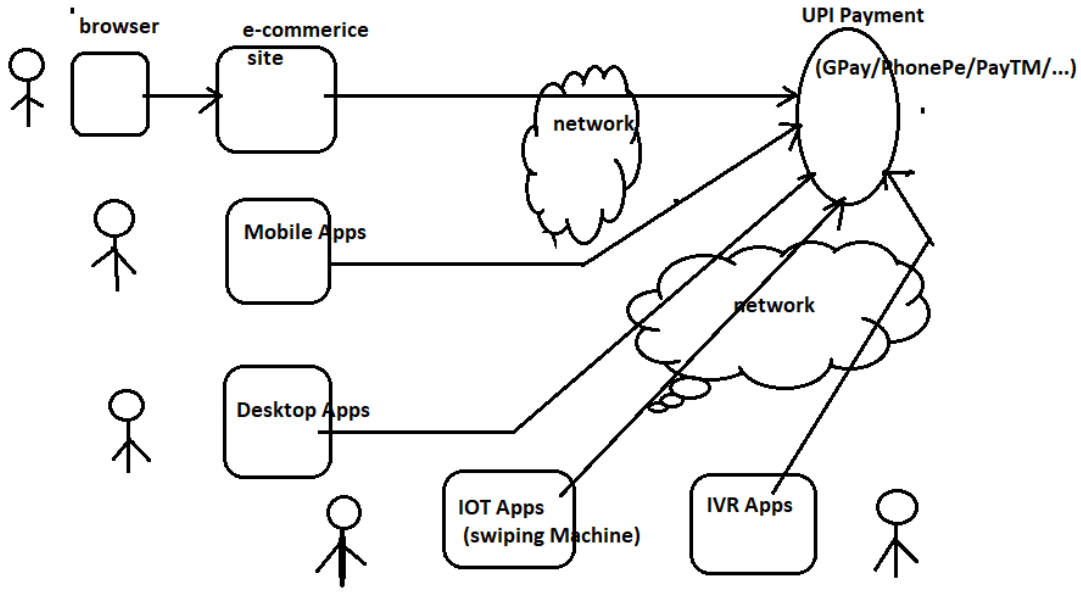
and etc..

IOT Apps :: Internet of Things i.e devices will use Internet Directly to get some self intellence
eg:: Alexa , Siri , Google Home , Wifi ACs and etc..

A typical E-commece App flow while doing Card Payment (Debit or Credit Card)



Gpay , PhonePe , Amazon pay and etc.. are what ? web applications or distributed Apps
Ans) they Are Distributed Apps

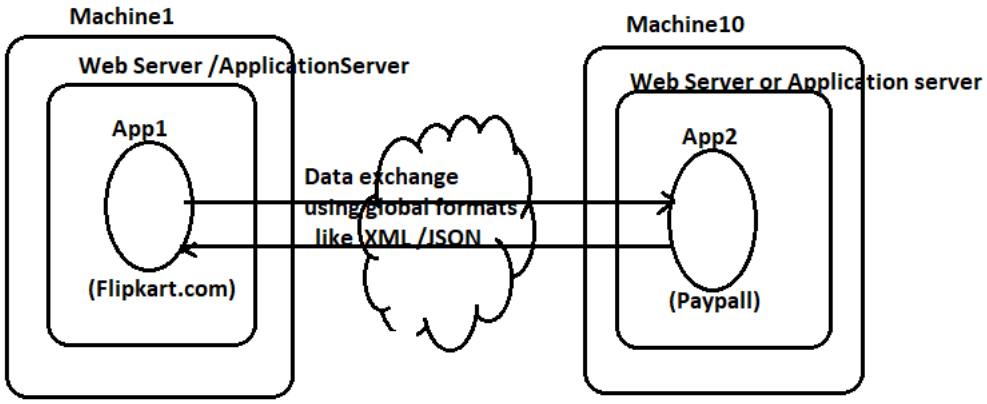


2. NTSPBMS615-More on Distributed Apps- May 24th 2022

What is the difference between web application (website) and Distributed App (Remoting App)?

Web application	Distributed App
a) It is client -server App dealing with browser to web application interaction	a) It is client -server App dealing with App to App interaction
b) It is browser to server communication becoz we will deploying the web application either web server or Application server	b) It is Server - Server Communication becoz the both client - server Apps can be deployed in two different servers.
c) Deals with request -response model of interaction	c) Deals with method invocation model of interaction
d) It is Thin Client - Fat Server model based Client -server App	d) It is Fat Client and Fat Server model of Client Sever App
e) Allows only browser as the client	e) Allows different types of Clients like web appliations, Deskop Apps , Mobile Apps, IOT Apps , IVR Apps and etc..
f) Supports only http , https protocols https: http over SSL SSL : Securee Socket Layer	f) Supports different types of protocols like http, https, JRMP, IIOP and etc.. JRMP :: Java Remote Method Invocation IIOP :: Inter Internet ORB Protocol ORB :: Object Request Broker
g) we can use following java technologies and frameworks to develop the web applications =>servlet,jsp (techonlogies /apis) => struts => jsf => webwork => spring mvc/spring boot mvc =>ADF and etc.. frameworks	g) To develop these apps we can use the following technologies /api and frameworks RMI EJB CORBA Technologies /api WebServices (JaxWS, Jax-Rpc, Axis ,apache cfx [For SOAP based web services] jax-RS , Spring Rest , Jersty , Rest Easy [For Restfull webServices] [Framework Env..]
h) eg:: flipkart.com , amzon.com, nareshit.com and etc..	h) eg:: paypall , visa/master/rupay/.. Apps , Gpay , phone pay Apps , weather report Apps, ICCScoreApps , IRCTS Apps, BSE Stock Apps and etc..

=> Distributed App talks about App to App or Server to server communication which can be there either in same Computer/machine or different computers of a network.



App Server = WebSErver ++

=>XML/JSON data is global format data becoz it is platform , language and architeture independent data

=> Two pass data between two incompitable Apps i.e which are there in two different platforms or languages or architectures we take the support global formats

=> Compare to Xml , the JSON format is having more popularity becoz of its simplication in data designing..

students info using xml

(tag based data)

```
<students>
  <student>
    <sno>101</sno>
    <sname>raja</sname>
    <sadd> hyd</sadd>
  </student>
   <student>
    <sno>102</sno>
    <sname>rajesh</sname>
    <sadd> vizag</sadd>
  </student>
</students>
```

Two students information is placed there but compare to real data .. the tags are more (decoration is more)

studnets info using JSON (Java Script Object Notation data)
---------- (key-value pairs based data)

```
{
  {
   "sno" : 101,
   "sname": "raja",
    "sadd" : "hyd"
  }
  {
   "sno" : 102,
   "sname": "rajesh",
   "sadd" : "vizag"
   }
}
```

A typical App contains 4 layers

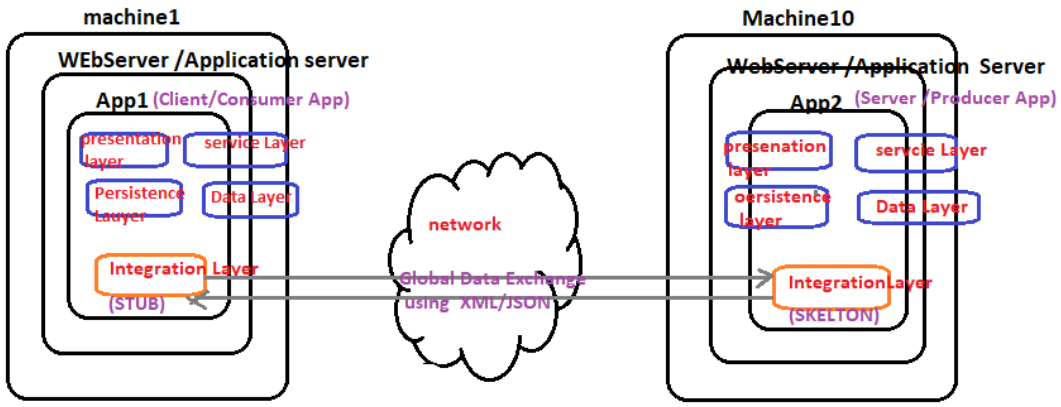
a) presentation layer ---> UI logics [html ,css, js , angular, reactjs and etc..)

b) Service layer ---> b.logics with dealing calculations, analuzations, filterings, sortings and etc..
[spring , sprinb boot , POJO java classes, java beans and etc..)

c) DAO Layer /Persistence Layer ---> persistence logics dealing with CURD Operations and DB Connections
[hibernate, jdbc, spring jdbc, spring orm, sprin data jpa and etc..)

d) Data Layer ----> The place where data will be persisted (DB s/ws)

=> While Developing Distributed App the Client App (cosumer App) and the Server App (Producer App) contains the above layers but also contains an extra layer called Integration layer .. The logics this integration layer will be developed by using Distribiuted Technologies or Frameworks.



=> The Integration logics of Consumer/Client App is technically called as "Stub" logics

=> The Integration logics of SErver/Producer App is technically called as "Skelton" logics

=>To develop the stub and skelton logics in Client - Server Apps of Distribute Apps we take the support Distributed Technologies like RMI, EJB,CORBA, WebServices and etc...

=> spring ,spring boot frameworks are having 20+ modules .. if certain programming is done using spring framework then it spring <module> .. if same module programming is done using spring boot framework then it is called spring boot <module>

=> To develop distributed Apps in spring we have "Rest" module .. if that module Apps are developed in spring style then it is called "Spring Rest" module app. if that module Apps are devleoped in spring boot style then it is called "Spring Boot Rest" module App.

spring boot = spring ++

List of Java Based Distributed Technologies and Frameworks

=>These are given to develop Integration layer logics (stub ,skelton) logics of Client/consumer or Server /Producer Apps to get Interaction b/w Client /consumer and Server/Producer Apps of same or different servers belonging to either same computer or different computers.

a) RMI (Remote Method Invocation)

b) EJB (Enterprise Java Beans)

c) CORBA (Common Object Request Broker Architecture)

d) WebServices (The specification or methodology to get App to App interaction using the protocol http)

Java is not just platform independent .. It is more powerfull becoz
it is Architecture Neautral

The popular computer manifacturng architecture are (popular)

a) IBM architecture
b) Apple/MAC Architecture
c) Sun Architecture

All these archictures are given create computers .. but their internal funcationality is different.

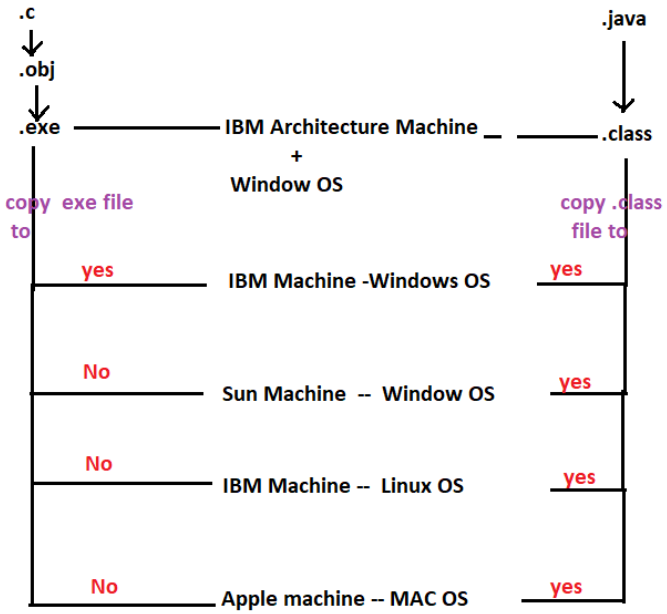
The Four Wheeler manifacturing Architectures are

a) Car Architecture
b) Jeep Architecture
c) Van Architecture
d) Streening Architecture
e) Mini Lorry Archicteture
f) Mini Bus Archicture
and etc..

All are four wheeler but manifacturing style is different..

=> All Del , samsung , HP, LG, Sony , lenovo brand laptops
and desktops are IBM Architectiure machines.
=> MAC /Apple PC are given basedp on Apple/MAC Architecture
=> Sun Spark machines are given based on Sun Architecture.

=> C,c++ Apps are not only platform dependent.. they are also Architecture dependent Apps where as JAva Apps are paltform independent and Architecturally neautral apps..



RMI (Remote Method Invocation)
====
=>It is Language dependent (Both Client and Server Apps must be there in Java)
=>It is Platform Independent (Both Client and Server Apps can be there in two different OS)
=>It is Architecturally Neautral (Both Client and Server Apps can be there in two different architectures)
=> Does not give Built-in Middleware services
(Ready to use services like logging, auditing ,security and etc...)
=> Allows to keep the Client And Server Apps in the Local Area Network .. but not on Internet network.

=>Uses JRMP(Java Remote Method Protocol) Protocol for communication

=> OutDated after the arrival of EJB (Enterprise JAva Beans)
=> It is still part of JDK s/w (part of JSE module)

EJB (Enterpirse Java Beans)

====

=> It is language dependent
=> It is platform independent
=> It is architecture neautral
=> Gives Built-in middleware services like logging, auditing ,security, Tx Mgmt and etc..

Logging :: keeps track of App flow
Auditing :: keeps track user activities
Security :: Authentication + Authorization
Tx Mgmt :: Executes the related logics by applying Do every thing or nothing principle

Tx Mgmt :: Transaction Management

=> To execute the EJB Comps (server App) we need the EJB container which is part of the heavy weight application server softwares like weblogic, glassfish, wildfly and etc..

=> EJB comps are heavy weight comps
=> EJB comps are complex to develop and need more resources for execution
=> EJB is outdated becoz its complexitity and heavyness

note:: RMI ,EJB Apps/comps (Server App logics) are not inter operable becoz they are language dependent i.e both Client and Sever Apps logics must be there in Java .

CORBA (Common Object Request Broker Architecture)
======

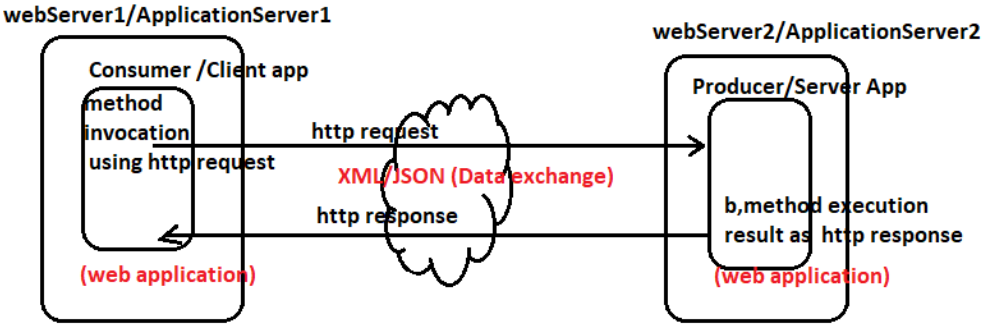
=> It is language indenpendent i.e the Client and server Apps can be there in two differnet languages
=> It is platform independent
=> It is Architecturally neuatral
=> CORBA is specification .. but it will be implemented in a special language called IDL (Interface DefinitationLanguage)
=> In the CORBA , the Client and Server Apps use IIOP protocol for communication
IIOP : Inter Internet ORB Protocol
ORB : ObjectRequestBroker
=> CORBA as shown multiple features as specfication but failed in the implentation.. So CORBA is outdated technology
=> CORBA is complex to learn and complex to use.

WebServices
==========

=> Web SErvices is the specification/mechanism that makes the Client And server Apps or Consumer and Producer Apps interacting with each other using protocol http.

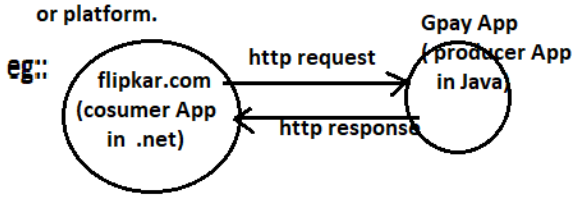
=> Here The client app /consumer invokes the the b.method of server/product app as http request and gets the output from server/producer app back to Client /Consume App as http response.

=> WebServices Apps are language independent
=> WebServices Apps are platform independent
=> WebServices Apps are Architecture independent
=> Web service comps/Apps are interoperable (client App can be there in any env.. and server app can be in any env..)



=> Java and other languages have given multiple technologies/api and frameworks to develop web service style consumer and producer Apps.

=> In WebServices env.. we can develope consumer apps in our choice language or architecture or platform .. similarly server /procucer App can be there in its choice language or architecture or platform.



Two implementations of WebServices

a) SOAP WebServices
b) Restfull WebServices

SOAP :: Simple Object Access Protocol
Restfull :: Representation Stateful

SOAP WebServices

=> Here the consumer and producer Apps exchange data in the form of soap messages over protcol http (xml messsages)

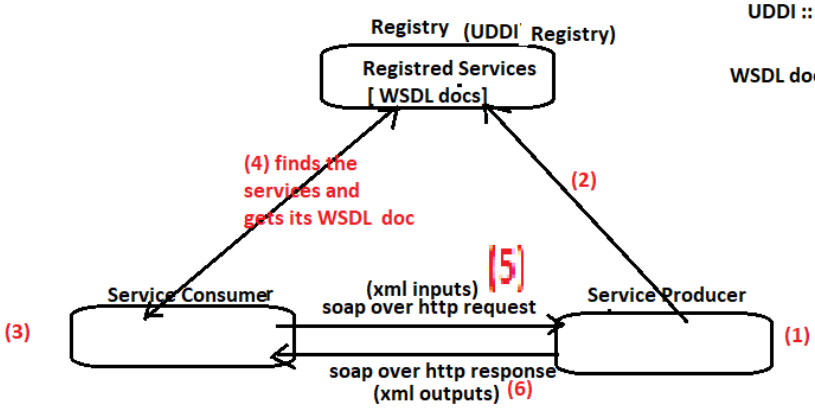
=> SOAP webServices do not allow exchange of data in the form of JSON content

=> http request contains SOAP message carrying inputs of b.method calls similarly http response contains SOAP message carrying the outputs of the b.method execution.

=> High level architecture diagram of SOAP WebServices

UDDI :: Universal Discovery , Description and Integration

WSDL doc : WebService Description language Document (xml doc)



(1) Service Prodiver/producer App development

(2) The producer Service will be registered UDDI registry having WSDL doc

(3)&(4) Service Consumer finds the Prodcuer service from UDDI registry and its WSDL document

(5) Service cosumer developes the logics based on WSDL document to cosume the services offered by the SErivce producer by using SOAP over http request

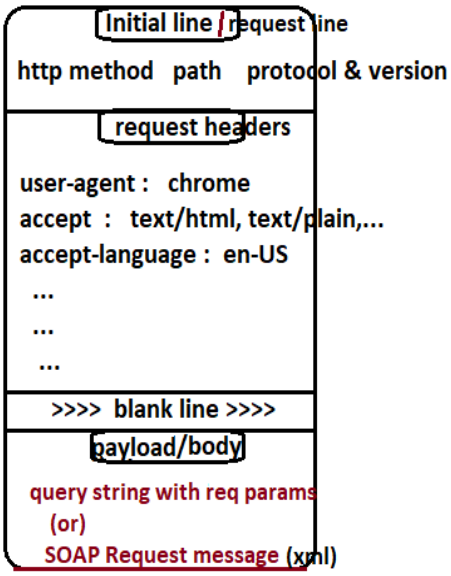
(6) The producer execute the b.method and its sends its output to Serivce consumer as SOAP over http response message.

The typical http request structure contains 3 parts

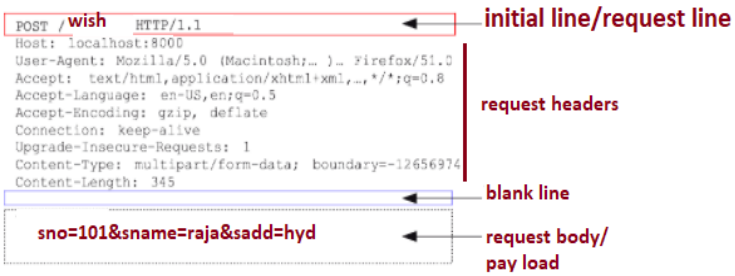
a) request line (basic info)

b) request headers

c) request body /payload



Example Http request structure

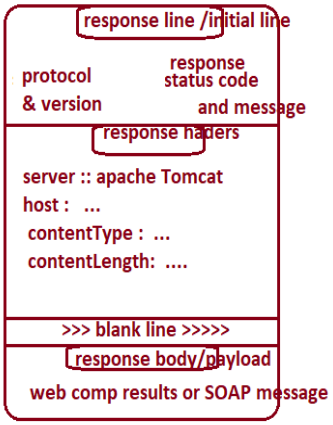


Soap request message over http request

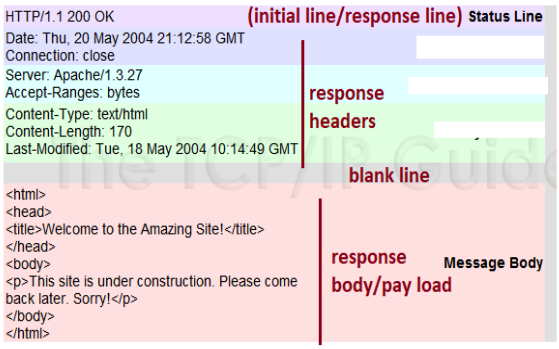
A typical http response contains 3 parts
a) response line / initial line (basic details)
b) response headers
c) response body/payload

Http response status code::
100 - 199 :: Information
200-299 :: Success
300- 399 :: Redirection
400- 499 :: Client Error
500 - 599 :: SErver Error

Http response structure
==========

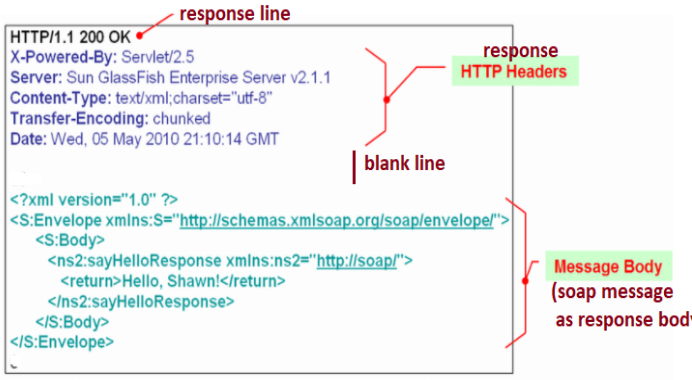


Example
HttpResponse

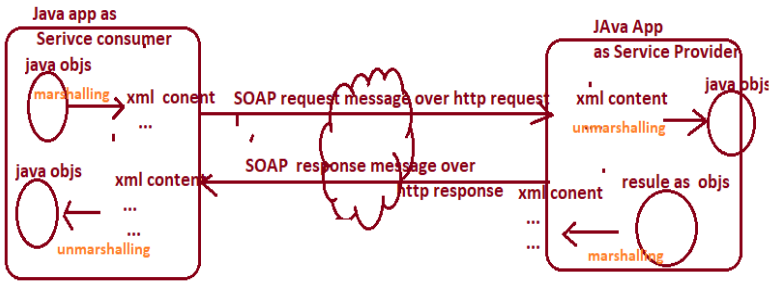


Normal http response contains the web comp genertad ouput (generally html code or plain text) as response body where as soap over http response contains the soap message(xml) as the response body

Example
SOAP over http response



=> if Both consumer and producer apps of SOAP webServices are developed in Java the consumer needs to convert objects inputs to SOAp Based xml requst messages also needs to convert SOAP bsaed xml response messages to java object output content. Similarly the Producer app also need to similar activies on the input xml content and generated object outputs



For converting xml content to java objs and vice -versa we take support of JAXB api
JAXB :: Java API Xml binding

JAXB API can perform both marshalling and unmarshallaing activities

=> The Process of converting Java Objects data to Xml content is marshalling and reverse is called unmarshalling

3. NTSPBMS615-May28th-2022 -SOAP WebServices and RestFullWebServices

=>To develop SOAP based web services jdk is supplying api or techology called Jax-ws and we have frameworks like apache axis , apache cfx and etc.. to develop the same SOAP based web services

=> Publishing service in registry .. and getting the service from the registry is technically called SOA architecture (SOA: Serivce Oriented Architecture)

Limitations of SOAP based webservices

=> It is based on the complex SOA (Service Oriented architecture) where seperat registerites (UDDI) are required to publish and find the services.
=> UDDI Registries are bit complex to arrange and manage
=> Supports only XML style global data format to pass data between consumer and producer
=> Send Xml messages as the SOAP based messages along with http request and http response is quiet complex
=> Seperate API called JAXB is required in Java env.. to perform marshalling(java objs to xml content) and unmarshalling activities (Xml content to java objs)

=> Publishing Service provider/producer details by preparing WSDL document is very complex
=> Developing Consumer app by reading the details about Service Provider from WSDL document is not so much easy process.

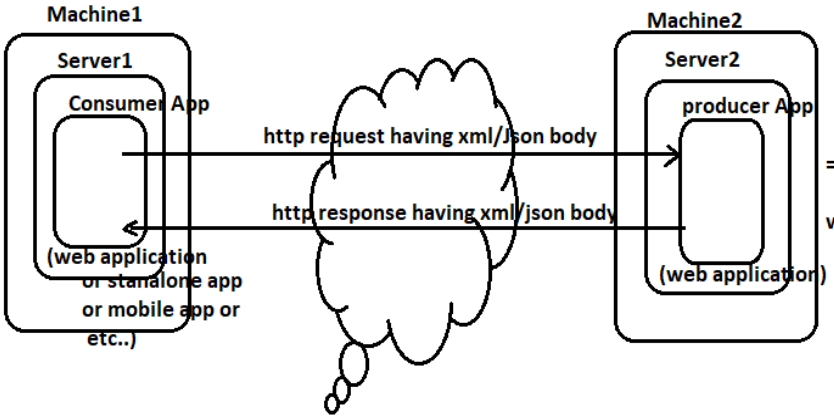
=> SOAP based web services development is complex to learn and use.

To overcome these problems .. use RestFul webServices

RestFul WebServices

=> It is also protocol http based Client-server or Consumer -Producer communication based application development.
=> It Does not use SOAP messages over http request and http response.. it keeps client inputs or Server outputs directly http request/http response body either in the form raw xml content or JSON Content (JSON is Recomanded)
=> In ReStfull webServices "Rest" is not a procotol it is a mechanism or methodology to send comps data in http request , http response as body in the form xml /json content.

=> It is based Consumer- Producer Architecture i.e no registries are quired to publish / find producer/publisher service



=> Every Producer/Publisher app in RestFull webServices is indentified with its URL /URI

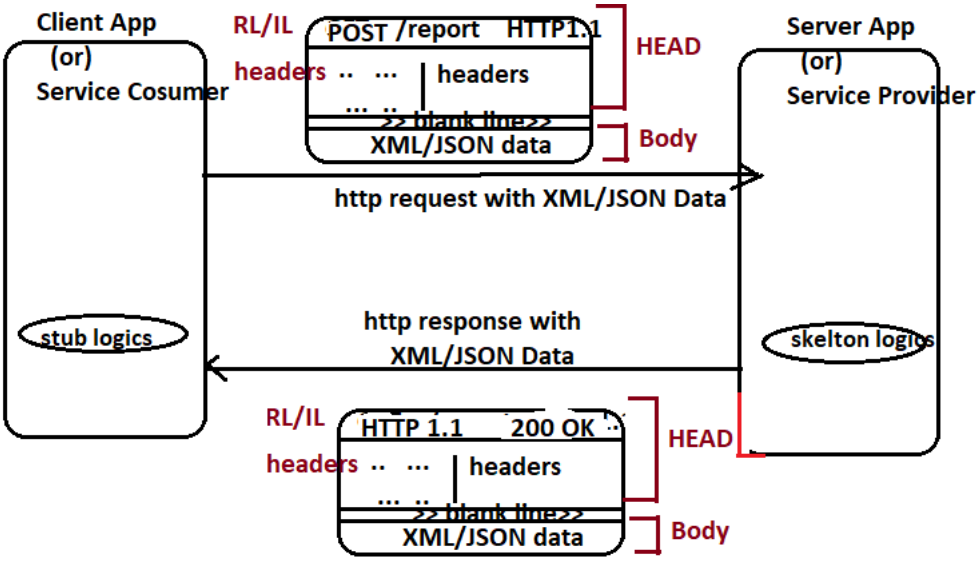
URL :: Universal/Uniform Resource Locator
URI :: Uniform Resource Indicator

=> Restfull webservices means "Representing the State of the comps/Apps" in the protocol http as body /payload

=> Restfull webservices is language independent
=> Restfull webservices is platform independent
=> Restfull webservices is Architecture independent

=> In java , we have Jax-RS as an api or technology to develop RestFul webSevices (both producer and consumer Apps) and we have Spring Rest (extenstion of spring mvc) , spring boot rest (spring boot mvc++), Restlet , RestEasy , Jersy and etc.. framework s to develop the Restful webservices.

=> In Restfull webservices ... No need of maintaning any registries becoz the server/provider/producer is identified with its uri/url and cosumer can use that url/uri to develop stub logics and to invoke the services/methods of producer/provider app.



=> While developing the server/producer/Provider App the skelton logics will be generated dymaically based on the api or frameworks we are using

=> While developing the Client/consumer App the stub logics will be generated dymaically based on url /uri we provide to api or framework that is to develop the client App.

Http request with JSON BODY

Method URI Version

```
POST http://localhost:8081/api/contacts HTTP/1.1
User-Agent: Fiddler
Host: localhost:8081
Accept: application/json
Content-Type: application/json
Content-Length: 121

>>>>> blank line >>>>>

{
  "Name": "Jane User:,
  "Address": "1 Any Street",
  "City": "Any city",
  "State": "WA",
  "Zip": "00000",
  "Email": "janeuser@example.com"
}
```

request line

request Headers

request Body (Content)

Http response with JSON Body

HTTP Version | Status Code | response line or initial line

```
HTTP/1.1 200 OK
```

response Headers

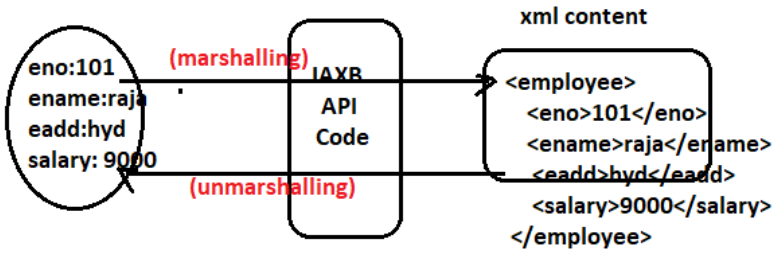
```
Content-Type: application/json; charset=ut
Server: Kestrel
X-Powered-By: ASP.NET
Date: Sun, 11 Feb 2018 18:34:00 GMT
Content-Length: 69
```

response Body

```
{
  "name":"Product",
  "category":"Appliances",
  "subcategory":"Microwaves"
}
```

=> To convert java objs to xml content (marshalling) and xml content to java objs (unmarshalling) we need to use special api called jaxb (Java api for xml binding)

```
Employee emp=
  new Employee(101,"raja","hyd",9000);
```



Jax-WS , Jax-RS apis and the frameworks like apache axis, apache cfx , Spring Rest, Resteasy and etc.. internally uses JaxB Apl for marshalling and unmarshalling activity..

=> Compare to Xml format , developers and industry prefers JSON format to define global data becoz JSON lightweight , simple to create and use.

xml content

```
<employee>
  <eno>101</eno>
  <ename>raja</ename>
  <eadd>hyd</eadd>
  <salary>9000</salary>
</employee>
```

Json content

```
{
 "eno": 101,
 "ename": "raja",
 "eadd": "hyd",
 "salary": 9000
}
```

=> JSON data with data defined Java Script notation

Java

```
int a=10;

String name="raja";


Student st=new Student();
st.setSno(1001);
st.setSname("raja");
st.setSadd("hyd");
```

st(Student class obj)

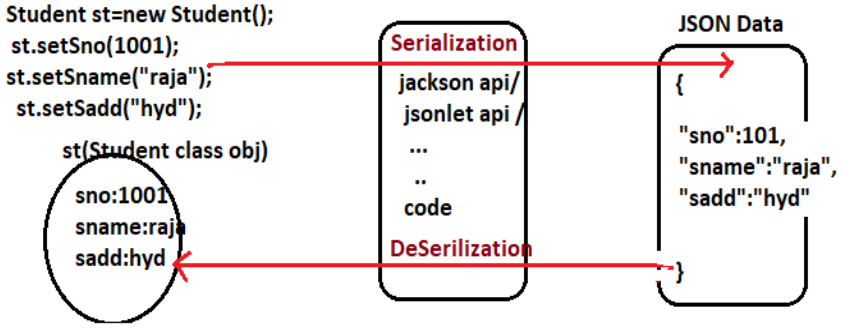
sno:1001
sname:raja
sadd:hyd

Java script

```
let/var a=10;

let/var  name="raja";


let/var  Student st={
                      "sno":101,
                      "sname":"raja",
                      "sadd":"hyd"
                     }
```

=>The process converting Java objs data to JSON data is called "Serialization" and reverse is called "DeSerialization" .. It is no way related to IOStreams Serialization and DeSerialization

=> we have multiple api /convertors to perform JSON style serialization and deserializsation activities
eg: jackson api (popular) , jsonlet and etc..



=> jax-RS api/technology and other restfull frameworks like spring rest, resteasy, jersry and etc.. internally uses jackson api to perform Serialization and Deserilization activities...

=> Developing the server/producer/provider App in restfull websevices is nothing but developing "API"

API :: Application Programming interface.

=>Earlier days API in java means collection of packages having classes ,interfaces, enums and annotations. (This api comes in the form of jar files)

=> After arrival Restfull web services especially if it is implemented .net or js or python or php .. API means the development server/producer/Prodiver App in RestFull webServices

Interviewer(HR/Technical) question

Have u been to API devleopment?

(RestFull WebSevice Server/ provider/ producer App)

the

What are end points to cosume the service?

ans) These are RestFul webservice comp (server/provider/producer App) url , request params and path variables details

Sunmary on webservices (Similarities and differences)

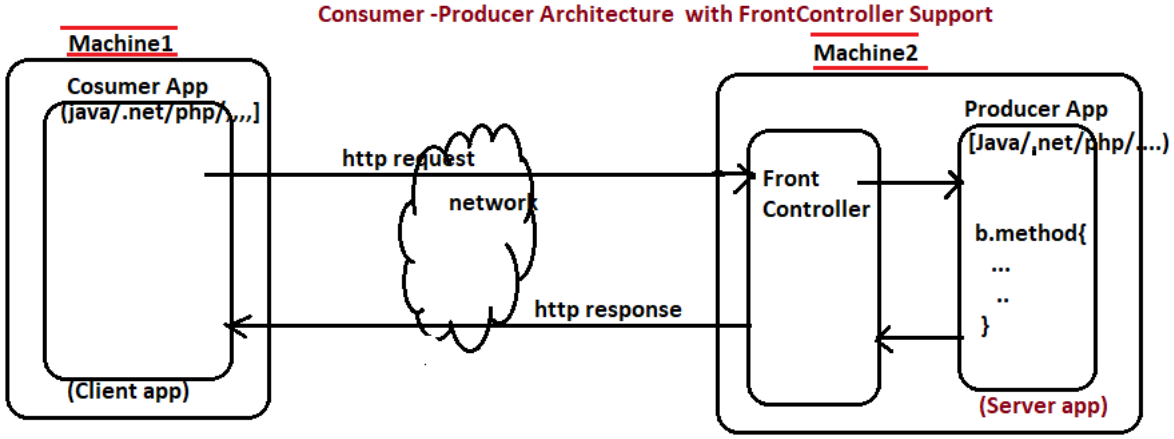
==========

WebServices

SOAP based	Restfull (Representational State Transfer)
[old and legacy]	[new and modren]
It is based on SOA (Service oriented Architetecture)	It is based cosumer -producer ARchicture with FrontController support
uses SOAP over Http protocol	Uses only protocol http
SOAP messages are xml messages i.e data always carried in the form of xml	Data here can passed xml or json or html or plain text and etc.. json is recomanded
Jax-WS java api/ technology to implement these websevices	JAX-RS java api /techology to implement these web services
Apache CFX , apache axix and etc.. are frameworks to implement these web services	Spring Rest , Rest easy , jersry , Restlet and etc are the java frameworks to implement these web services
These are Interoperable webservices	These are interoperable web services
Since SOAP messages are hevy messages more bandwith is required to transfer data	Since data travells in simple forms more network bandwith is not required
Bit complex .. not easy to learn and develop	Bit lightwight , easy to learn and develope
Gives bit extra security becoz of the complex SOAP messages	Bit less security .. becoz the data travels in raw format (query string data or req/res body data)
UDDI registry is required	No Registrires are required..
WebService comp (Producer App) is identified with wsdl doc details (xml based)	WebService is indetified with it its url/uri (End point details)
Allows different types of Clients	Allows different types of Clients

==> RestFul webServices are based on Consumer-proddcuer having FrontController web comp involvment in the middle.

Consumer -Producer Architecture with FrontController Support



=> if both client/cosumer and server/producer App are developed as Java classes.. the Producer App can not take http request directly .. for that we need web comps like servlet or jsp comp. To solve this problem .. The Producer app takes the support front controller web comp.

=> A front controller is a special web comp of a web application that traps and takes either all or multiple http requests , applies the common system services like logging ,auditing, security and et.. and delegate the request to server/producer app .. there executes the b,method gets the results and sends those results to client /cosumer App as http response

=> In simple words The FrontController acts as entry and exit point for either all or multiple requests and responses.

=> JAva frameworks are providing multiple readymade frontController servlets

Struts ---- ActionServlet
spring MVC/spring boot MVC ----- DispatherServlet
JSF ---------- FacesServlet

Spring Rest /Spring Rest = Spring MVC ++ /Spring boot MVC ++

=>In Spring Rest /Spring boot Rest .. we are going to use DispatcherServlet comp as the FrontController Servlet comp

In Spring Rest/Spring Boot Rest App

=>FrontController -----> DispatcherServlet
=> Producer App/Server App/provider App ----- @RestController class (Java class)
(API comp /Service) (@Controller + Response Body/content)
(@ResponseBody)

note:: For this @RestController class we can take @Service(service class) ,
@Repository (DAO classes) as the supporting classes

note:: Serivce class contains b.logic (calculations, analyzations, filtering,sorting and etc..)
DAO class contains persistence logics (CURD operations, DB connections and etc..)

=> @RestController class contains multiple b.methods as business operations mapped with request paths and request methods/modes
(GET,POST, HEAD, PUT,DELETE,TRACE,OPTIONS, PATCH and etc..)

```
@RestController  // (@Controller ++)
public class EmployeeOperationsController{

 @GetMapping("/all")
 public String getAllEmployess (){
    ....
    ...
  }
```

b.method mapped with request path and mode (/all) (GET)

```
 @PostMapping("/register")
 public String registerEmployee (Employee emp){
   ...
   ...
   ..
 }
...
..
...
..
}
```

b.method mapped with request path and mode (/register) (POST)

While developing RestController comp as Server /Producer / Provider App we use different http methods for different varieties of operations

(a) GET ----> for getting Data from server (Select Operations)
(b) POST ---> for giving /posting data to server (INSERT Operations)
(c) PUT ----> for updating data of the server (UPDATE operations)
(d) DELETE ----> for deleting data from the server (Deletee Operations)
(e) PATCH----> for partial data Update operation (Update operation)

note:: OPTIONS , TRACE , HEAD are used for monitoring and dubugging activities.. they are not given for the main business activties (CURD operations)

CONNECT is http method/mode is reservered for future.

List out http methods/modes u know /use?

GET, POST , PUT,DELETE , PATCH,HEAD , TRACE,OPTONS, CONNECT

=> While developing spring Rest/Spring Boot Apps the folling thing involved

(a) Cosumer - producer Communication Archictecture
(b) FrontController Desing PAttern
(c) MVC Archicteture
(d) protocol http
(e) XML /JSON fomat of data sending and fetching

Q) Why spring mvc /spring boot MVC deals with GET,POST modes/methods of request and why Spring Rest/Spring Boot Rest deals with many request modes/methods (GET,POST, PUT, DELETE, PATCH and etc..)

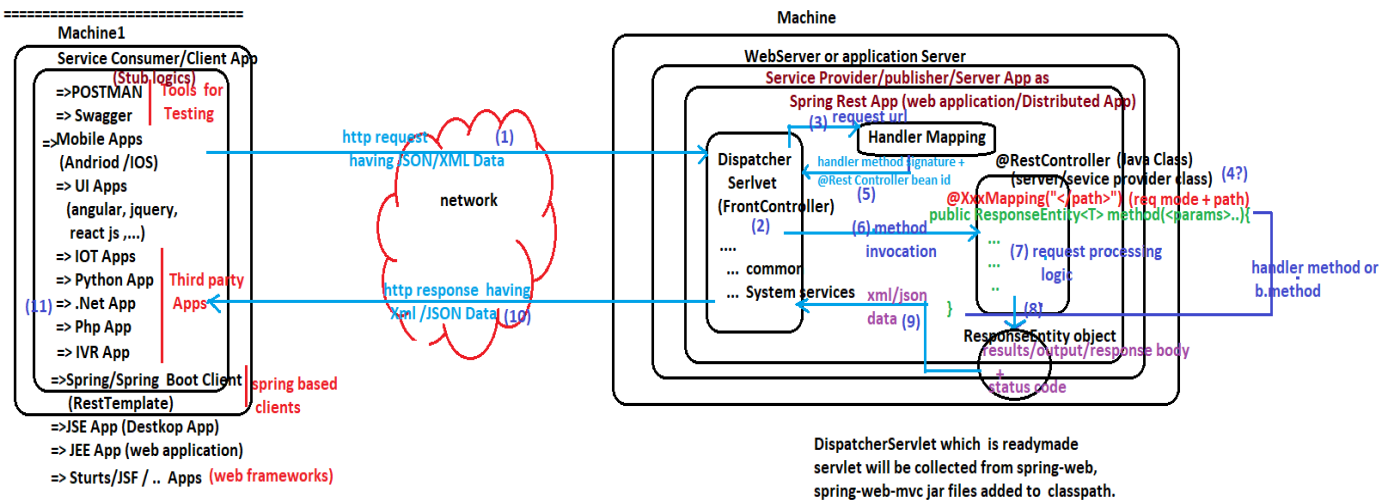
Ans) spring mvc /spring boot mvc are given develop web applications whose client browser who can send only GET ,POST mode requests ,SO other modes are not supporrted

spring rest /spring boot rest are given develop Distributed applications whose client can be a tools (postman /swagger api) , destktop app, android app , IOT apps , web applications and etc.. having capability to send different modes of requests like GET,POST,PUT,DELETE, PATCH and etc.. So we deal with more request modes/methods in spring rest/spring boot rest..

Spring Rest /spring boot Rest

=> It is extension module of spring mvc or spring boot mvc
=> It uses DispatcherServlet comp (ready made Servlet comp) as the Front controller Comp
=> Here we develop service provier/publisher /server App as spring bean style java class annotation with @RestController
=> This @RestController class can take the support of service, DAO classes while processing the requests
=> In spring MVC/Spring boot MVC Web application , the browser is client which is non-porgammable app..
=> In Spring Rest/Spring Boot Rest Distrobuted App , the Client App is another software app i.e it is programmable application invoking the b.methods of Service provider/publisher/server App using HttpRequeets.

Spring Rest/Spring Boot Rest Flow (Spring MVC/Spring Boot MVC Flow with out View comps)



=>In spring Rest/Spring Boot Rest Flow there is no ViewResolver ,View comps becoz the by having the return type ResponseEntity<T> and by getting @ResponseBody from @RestController the handler method of RestController class (Service provider class) sends response/output directly to Client /cosumer App through the FrontController Dispathcer Servlet

@RestController = @Controller + @ResponseBody

note:: if the handler method of controller/ RestController class is annotated with @ResponseBody then generated response automatically goes to client app directly through the FrontController Servlet.

=> Deployment :: The process of keeping the App in the server is called deployment
=> UnDeployment :: The process of removing the App from Server is called undeployment

What is the difference b/w spring MVC/spring boot MVC flow and spring rest /spring boot rest flow?

Ans) In spring rest/sprign boot Rest Flow there will not be any View Resolvers and view Comps becoz the handler methods of @RestController class (server /service provider class) contains @ResponseBody annotation directly or indirectly giving capability to handler method to deliver its output/results directly to client app/cosumer app through the FrontController Dispatcher SErvlet.

With respect to diagram
==========

(1) Consumer App gets Service provider App uri/url and write its stub logics to send http request having xml/json data

(2) The FrontController Servlet (DispatcherSerlvet) traps and takes the request and applies the common system services

(3) DispatcherServlet (DS) handovers the request to Handler mapping giving request url

(4) HandlerMapping searches for matching request path handler method in all the RestController classes

(5) Handler Mapping returns Handler method signature and Restcontroller class bean id back to DS

(6) DS invokes the handler method of RestController

(7) The handler method process the request either directly or by taking the support of SErvvice, DAO classes and generates the results

(8) Handler method returns ResponseEntity obj having results + response status code

(9) Based on the apis we haved to the Project , the results of ResponseEntity object will be converted to Xml/JSON data

(10) The frontController (DS) takes results as xml/json data and sends to cosumer app as http response having xml/json data.

(11) Consumer App collects xml/json data from http response and process it accordingly..

4. NTSPBMS615-June2nd-2022 -Frist Spring RestApp Development

=>Spring Rest = spring mvc ++
=>Spring Boot Rest = Spring boot mvc ++

=> 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
using XxxMapping("<path>") annotations .. These methods can be called handler methods or
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.

5. Spring Rest & MicroServices

Day-1- https://www.youtube.com/watch?v=Tlp45YhOE-8

Day-2- https://www.youtube.com/watch?v=EpaNqWAVadI

Day-3- https://www.youtube.com/watch?v=BWfLY15xj_w

Day-4- https://www.youtube.com/watch?v=UQJ8ToGWN4Y

Day-5- https://www.youtube.com/watch?v=05Kg91stFtU

Day-6- https://www.youtube.com/watch?v=p9MJ-aSNlM0

Sample Rest Controller (legacy style)
==========

```
@Controller
@ResponseBody
@RequestMapping("/message") // global path
public class WishMessageRendererController{


 @GetMapping("/wish")   // path specific to each b.method
 public ResponseEntity<String/<object>/<collection>  showMessage(){
    ....            ResponseEntity<T>
    ...
    ...
  return ResponseEntity object;
   } //method

}//class
```

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)

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)
note:: To convert into xml contnet .. we need to add special apis/libaries to classpath.

Sample RestController (modren style)
==========

```
@RestController
@RequestMapping("/message") // global path
public class  MessageRenderController{

@GetMapping("/wish")
 public   ResponseEntity<String/Collection/Array> showMessage(){
    ...
    ...
    ..
  return ResponseEntity object ;
   }
}
```

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 acts handler 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)

==========

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

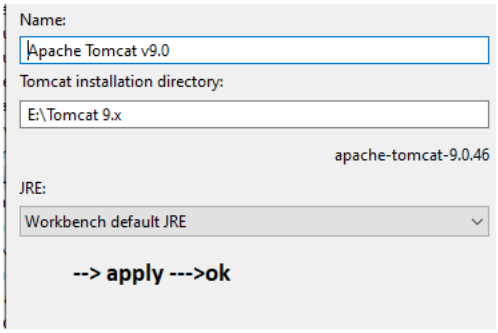
==========a

step1) make sure that following software setup is avaiable

=>eclipse JEE IDE with STS plugin (2020+)
=> Tomcat 9.x server (This is optional if u r planning to use spring boot suppled embedded Tomcat)
=> jdk 1.8+

step2) make sure that Tomcat 9.x server is configure with Eclipse IDE

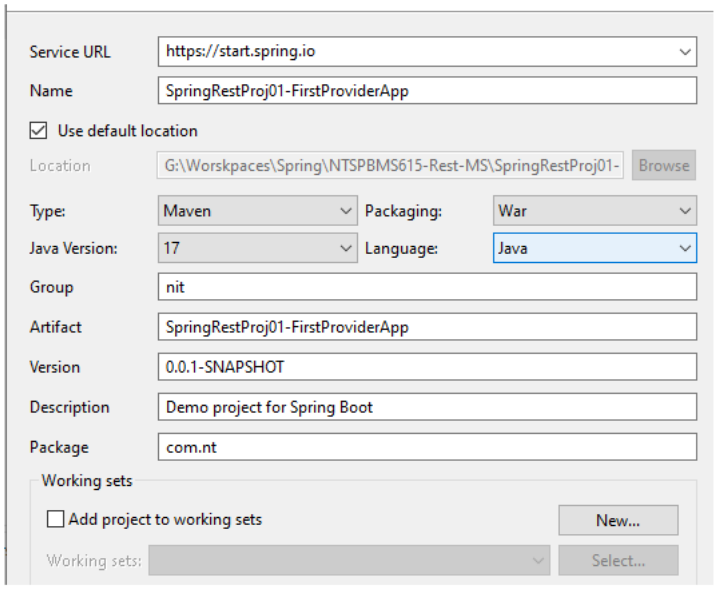
a) complete the installation Tomcat 9.x
b) window menu ---> preferences ---> servers --- Run time env.. --->
add --> select apache tomcat 9 --->



Go to servers tab ---> click add server link ---> select apache tomcat 9 ---> ....

step3) create spring boot stater Project adding "Spring web Starter" , "devtools"

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



--> 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)
                  msg= "Good Afternoon::";
            else if(hour<20)
                  msg= "Good Eveing::";
            else
                  msg="Good Night";
            // create and return Response Entity object having  response contnet  and status code
             ResponseEntity< String>  entity=new ResponseEntity<String>(msg,HttpStatus.OK);  //(body, status)
      return          entity;
       }


  }
```

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

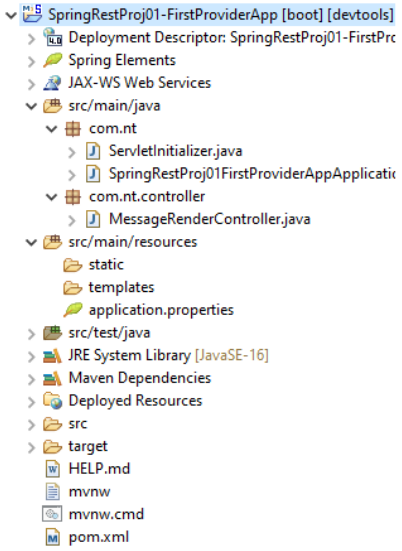
Right click on the Project ---> run as ---> run on server ---> select Tomcat 9

-->next --> ....

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

http://localhost:2020/SpringRestProj01-FirstProviderApp/messageapi/wish **(In browser address bar)**

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

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

global path

context path given in application.properties

method/operation path

6. NTSPBMS615-June 3rd-2022 -Frist Spring RestApp Testing Using POSTMAN Tool

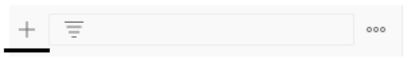
POSTMAN is tool that is given to test RestApis (Server/Producer/Publisher App of Restful webService)

=>A browser s/w as client can send only GET,POST mode requests
=> POSTMAN tool can send all the 8 modes of requests (GET,POST,HEAD,PUT,DELETE,PATCH, OPTIONS,TRACE)

Steps to test RestAPI using POSTMAN Tool
==========
step1) download and install postman tool
https://www.postman.com/downloads/
(alway skip registration prcoess)

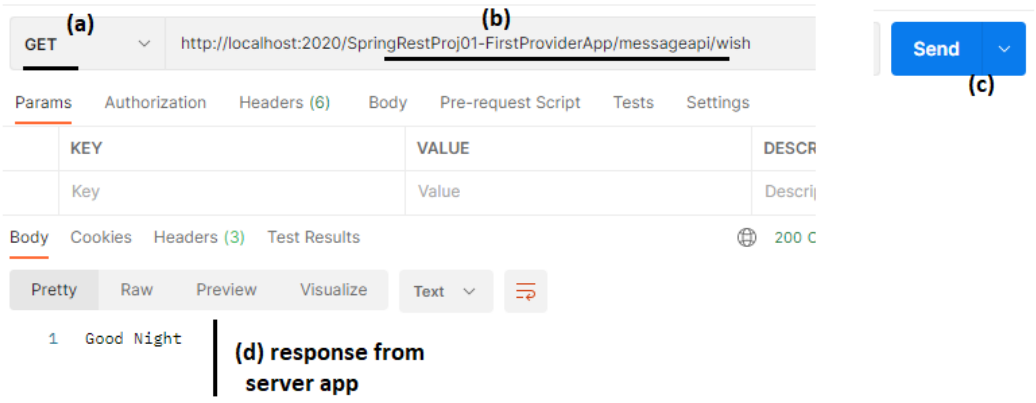
step2) launch postman tool (use icon on the desktop)

step3) create new collection



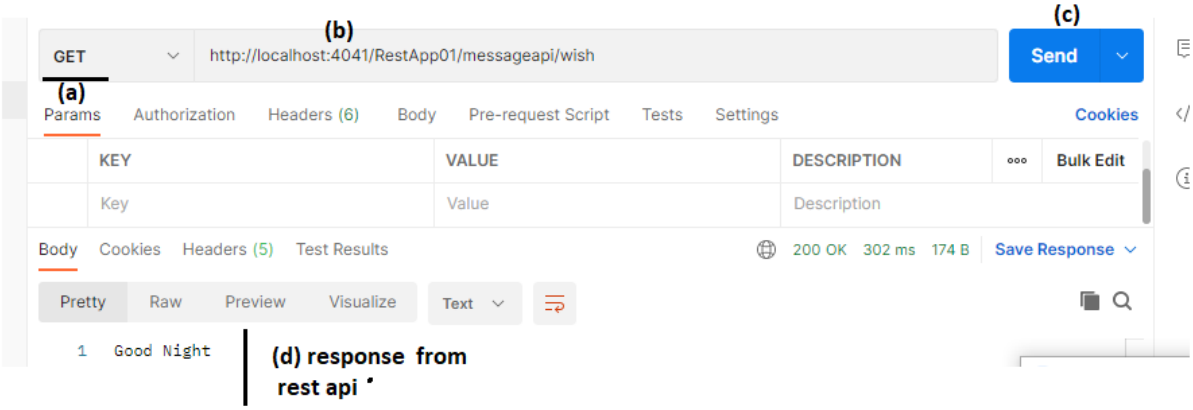
(Collection name)

step3) Add request to the collection
expand NTSPBMS615 Colletion ---> add a request
--> req1



note:: While performing step3 make sure that spring Rest api app (server/producer/ publusher App) is in running mode by deploying in external tomcat server..

if we run the same Spring Rest App in the eclipse IDE as the spring boot App (deployed in Embedded Tomcat server) then use the following in POSTMAN tool for testing



What is the difference b/w @Controller and @RestController ?

@Controller	@RestController
(a) It is enahacement of @Component	(a) It is enhacement of @Controller
(b) Given in spring 2.5 (Legacy Annotation)	(b) Given in spring 4.x (new annotation)
(c) makes the java class as spring bean cum web controller controller having ability to take http requests	(c) makes the java class as spring bean cum web controller + Rest api server/Producer/Publisher class having ability to http requests
(d) Useful in both spring mvc/Spring Rest and spring boot mvc/ spring boot Rest Apps	(d) Useful only in spring Rest /spring boot Rest Apps
(e) To use in Spring Rest/Spring Boot Apps we need to add @ResponseBody explicitly on the top of the class	(e) Here not required .. becoz @RestController = @Controller + @ResponseBody
(f) Every Handler method does not get @ResponseBody automatically	(f) Event Handler method also called b.method gets @ResponseBody automatically
g) By Default handler method looks to use ViewResolver and view comp to process the generated results .. then sends the results to client through DispatctheServlet. (becoz @ResponseBody will not be there on the top of method)	g) By default the handler method the sends the generated results/outputs to cosumer/Client App through FrontController(DispatcjerServlet) Servlet

note:: As technical guy.. better not to compare @Controller and @RestController rather use @RestController as a convienient annotation for @Controller in Restful webServices.

In http 1.1 the following http methods are avaialble
GET
POST
HEAD
PUT
DELETE
PATCH
TRACE
OPTIONS
CONNECT (Resevered for future)

In Rest Api Development (Server /Produder/Publisher) App we use the following http methods/modes to perform some business activities (related to CURD Operations)

=> GET ----> @GetMapping b.method (for select operation with out querying or with querying)
=> POST ----> @PostMapping b.method (for create operation --- insert/registration activaties)
=> PUT ----> @PutMapping b.method (for update operation --- full data modification activaties)
=> DELETE ----> @DeleteMapping b.method (for delete operation --- remove data activaties)
=> PATCH ----> @PatchMapping b.method (for partial update operation --- partial modification of data activaties)

```
@GetMapping("/all")
public ResponseEntity<T> getAllEmployees(){
  ...
  ...
}

@PostMapping("/save")
public ResponseEntity<T> registerEmployee(Employee emp){
  ...
  ..
}
```

```
@PutMapping("/update")
public  ResponseEntity<T>  modifyEmployee(Employee emp){
  ...
  ..
 }

@PatchMapping("/update")
public  ResponseEntity<T>  modifyEmployeeEmail(String email){
  ...
  ..
}

@DeleteMapping("/delete")
public  ResponseEntity<T>  fireEmployee(long empno){
  ...
  ..
}
```

The signatures of of these methods can be improvised ..So keep more focus on @XxxMapping annotations..

=> TRACE, OPTIONS,HEAD will not be used in RestApI development But they will be used in App's monitoring administrative activies..

we do not have @HeadMapping , @TraceMapping , @OptionsMapping annotations becoz there is no praticality with these request modes in rest api development.

=>HEAD mode is same as GET mode but HEAD request related reponse does not contain response body .. So we can not use it for select operations like GET mode .. The "HEAD" mode generally to useful to check wheather webcomp/ restapi method avaiable or not

=> TRACE mode request given to web comp or rest api method will provide all details that are involved in the execution either to get success or to get failure So it is veryful for Debugging operations.

=> if u want to get all the comps/processes the that are involved in requet-response generation process then we need to give TRACE mode request.

=> The OPTIONS mode request given to web comp or rest api method will give different request modes that can be used to generate the request that webcomp or rest api..

```
@Webservlet("/testurl")
publi  class TestServlet  extends HttpServlet

  public void doGet(req,res)throws SE,IOE{
    ...
    ..
  }
}
```

=>if we give "OPTIONS" mode request to this comp we get the reponse having the following response header "allow"

allow : GET,HEAD,TRACE,OPTIONS

```
@Webservlet("/testurl")
publi  class TestServlet  extends HttpServlet

  public void doPost(req,res)throws SE,IOE{
    ...
    ..
  }
}
```

=>if we give "OPTIONS" mode request to this comp we get the reponse having the fresponse header "allow"

allow : POST, TRACE, OPTIONS

```
@RestController
@RequstMapping("/messageapi")
public  class MessageRenderController{

  @GetMapping("/wish")
  public  ResponseEntity<String>  showMessage(){
    ...
    ..
    ..
  }
}
```

if we give OPTIONS mode request to http://localhost:2020/SpringRestProj01-ProviderApp/messageapi/wish url then we get the following content in the rresponse header "allow"

allow : GET,TRACE,HEAD,OPTIONS

HEAD, TRACE, OPTIONS modes will not used in web comps , Rest api methods development.. they will used only for web application or Rest API monitoring Apps or Admin Apps.

=> The HEAD,GET,TRACE,OPTIONS modes are idempotent modes (safe to repeat the request) becoz they perform read operations on the server either through web comp or through rest api methods.

=> The POST,PUT,DELETE,PATCH modes are non-idempotent modes (not safe to repeat the requests) becoz they perform update/write operations on the server either through web comp or through rest api methods.

A typical or complete Restful Application contains

a) Sever/Provider/Publisher App as Rest Api comp having b.methods mapped with request paths | This is nothing but RestAPI development having endpoints

(this provider app can be developed using different restfull webservice apis like jersy ,resteasy , jax-rs , spring rest, Restlet and etc.)

b) Client /Cosumer App | communicates with RestApi comp and invokes the b.methods using porotocol http

This App can be

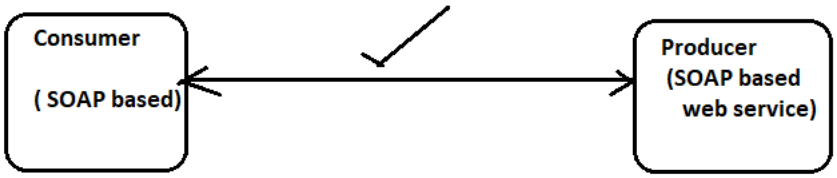
- POSTMAN, swagger | Tools for testing
- .net, php, python apps | Third party apps
- jquery, angular, angular JS, react JS, and etc.. | UI Apps

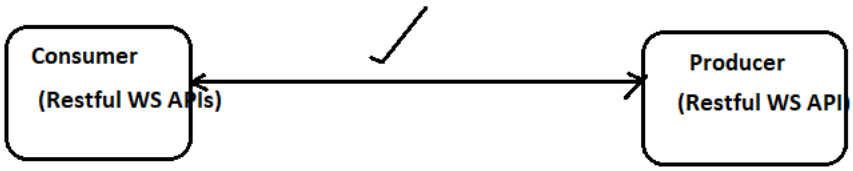
IOT apps
Embedded System Aps
IVR Apps

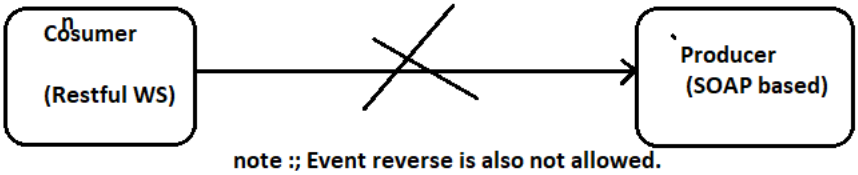
andriod Apps
IOS Apps

web application
Destktop App
enterpirse Apps

These Client apps can be developed using different Rest APIs like jersy ,spring Rest, Restlet , jax-RS and etc..

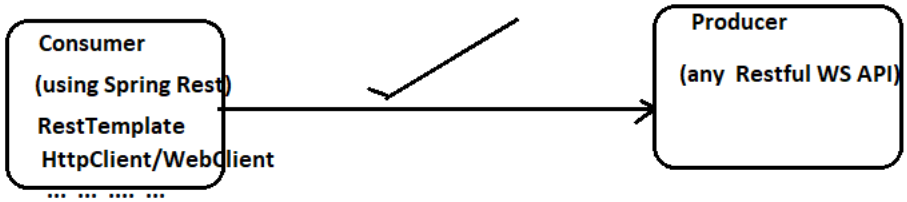






note :; Event reverse is also not allowed.

While working Spring Boot Rest we can develop Client App using RestTemplate , HttpClient/WebClient support...

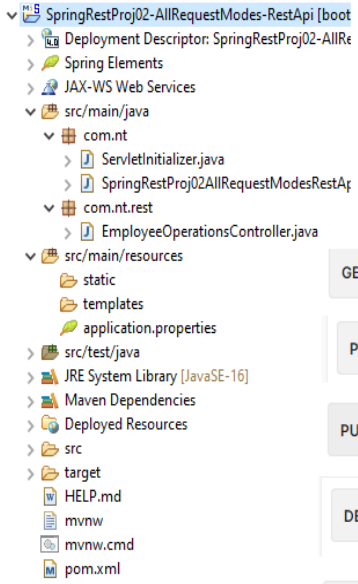


=> Second RestApi Development using Spring Rest covering different request methods of modes b.methods..

RestController
==========

```
package com.nt.rest;

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("/employee/api")
public class EmployeeOperationsController {

	@GetMapping("/all")
	public   ResponseEntity<String>  getAllEmployees(){
		System.out.println("EmployeeOperationsController.getAllEmployees()");
		return  new ResponseEntity<String>("GET Mapping -- Fetching all employees", HttpStatus.OK);
	}

	@PostMapping("/save")
	public ResponseEntity<String>  registerEmployee(){
		System.out.println("EmployeeOperationsController.registerEmployee()");
		return  new ResponseEntity<String>("POST Mapping -- saving the Employee", HttpStatus.CREATED);
	}

	@PutMapping("/update")
	public ResponseEntity<String>  updateEmployee(){
		System.out.println("EmployeeOperationsController.updateEmployee()");
		return  new ResponseEntity<String>("PUT Mapping -- updating the Employee", HttpStatus.OK);
	}

	@PatchMapping("/updateEmail")
	public ResponseEntity<String>  updateEmployeeEmail(){
		System.out.println("EmployeeOperationsController.updateEmployeeEmail()");
		return  new ResponseEntity<String>("PATCH Mapping -- updating the Employee partial", HttpStatus.OK);
	}

	@DeleteMapping("/delete")
	public ResponseEntity<String>  deleteEmployee(){
		System.out.println("EmployeeOperationsController.deleteEmployee()");
		return  new ResponseEntity<String>("DELETE Mapping -- deleting the Employee ", HttpStatus.OK);
	}

}
```

These methods signatures are not at their best .. can be improvosed ..and will be improvised in future.



7. NTSPBMS615-June6th-2022-Working with JSON from @RestController

Different popular error codes while working with protocol "http"

404 :: Requested resource is not found (no comp is mapped with request url)
405 :: Request resource is found.. but it is not ready to handle give http mode request .. For example for @PostMapping(-) method .. u have given GET,PUT,PATCH,DELETE mode of request .. that given 405 error.

401 :: Authentication is failed
403 :: Authorization is failed
| related to security.

500 :: Server/container had faced problems to load and instaiate the the requested resouce related class or some exeception is raised in the execution of requested resource.

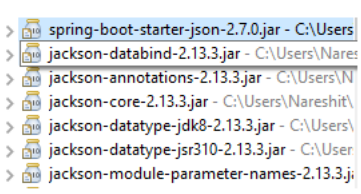
=> There are lots free/public or paid/commerical RestFull web services in market .. So most of times we need to know how to consume services from those free/paid rest api services by gathering rest api and its end points details (urls /paths of different b.methods of rest api comp)

eg :: paypal service (payment broker api)
Gpay API service
Rozar pay API service
weather Report API services
and etc..

Sending JSON format data as the response content/body from the b.methods of Rest API

=> if the b.method /handler method return type's (ResponseEntity<T>) Generic content is other than <String> (i.e any class object other than <String>) then Jackson api kept in classpath will be activated automatically to convert that object to into JSON Content

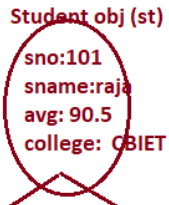
=> The spring boot gives built-in support for jackson api..



These jar files comes with basic spring boot support added to the Project.

JSON (Java Script Object Notation)
=====
=> It is one of the global formats to send data over the network between two incompitable or compitable systems
=> YML , MongoDB docs and etc.. are the Bi- Products JSON i.e they also maintain data in json format..
=> JSON global format is maintaining data in the form of key: value paris
=> The key must be String content kept in " " .. if the value is String content ..then the value should also be in "" otherwise not required.
=> { .... } in JSON represents main object or sub object or map collection
=> Arrays in JSON represented using [-,-,-]
=> List/Set collections data of java will be treated as arrays in JSON .so use [-,-,-] to represent their content
=> In JSON Arrays, List,set Collection data are called 1D arrays (One dimentional array)
=> In JSON Map Collection data is called 2D arrays (Two dimentional array)
=> In JSON , Map Collection data and HAS- A property data will be represented as sub object data of main object using { ... } ..

eg:: Student st=new Student(101,"raja",90.5, "CBIET");



JSON Global format

```
{
  "sno": 101,
  "sname": "raja",
   "avg": 90.5,
   "college": "CBIET"
}
```

xml global format

```
<student>
  <sno>101</sno>
  <sname>raja </sname>
  <avg> 90.5 </avg>
   <college> CBIET </college>
</student>
```

=> In Xml data is less .. the decoration is more..
=> JSON data is key=value pairs based data having more readability with less decoration.

note:: To convert xml content into JSON content , use JAXON API

What is the difference b/w XML and JSON API?

JSON	XML
(a) It is Java script Object Notation	(a) It is Extensible markup language
(b) It is way of represnting data using java script objs	(b) It is the way of representing data using tags
(c) maintains the data in the form of key: value paris { "key": value , "key": value }	(c) maintains data as xml tags body content and attribute vlaues.
(d) This format is given by Java Script Vendor (NetScape + Sun Ms/Oracle corp)	(d) This format is given by w3c (Wold wide web consortium)
(e) No schema or DTD rules are there to validate the data	(e) DTD or XSD rules can be imported to the xml files to validate the data..
(f) JSON content is easy to read and process	(f) Xml file content is bit complex to read and process
(g) To Process JSON data we take the support of Jackson api ,GSON API ,Simple Json api and etc..	(g) To process the xml data .. we take the support of xml apis like SAX API, DOM API, JDOM API, JaxB API , STAX API , DOM4J API and etc..
(h) The Process of coverting Object to JSON data is called Serialization and reverse is called DeSerilziation	(h) The Process of converting Object data to Xml is called marshalling and reverse called unmarshalling
(i) Does not allow comments in JSON content	(i) . Allows the comments
(j) No need placing/importing schema /namespace uri at the top of the document	(j) While developing the perfect xml document we import schema/namespace uris at top of the xml document
(k) Gives support for arrays and collections	(k) no direct support for arrays and colelctions
(l) Data is not hierarchal here and less structured becoz here data is key=value pairs data.	(l) Data is hierarchal and strong strucutred becoz xml tags and their hierarchy
(m) JSON data is less secured	(m) Xml data is more secured..

(n) Supports only UTF-8 chacters	(n) Supports multiple charsets including UTF-8
(o) useful only in Restfull web services	(o) So popular in data exchange while developing SOA Apps like SOAP webServices .can also be used in Restful webServices
(p) It is light weight becoz it needs less bandwith	(p) it is heavy weigth becoz it needs more bandwith to send over the network
(q) It is object oriented data ..	(q) It is document oriented data..
(r) Allows to keep data as string, numbers ,booleans	(r) every thing will be treated as strings.

JSON is more recomanded .. where ever it is possible JSON ..then prefer usng it.

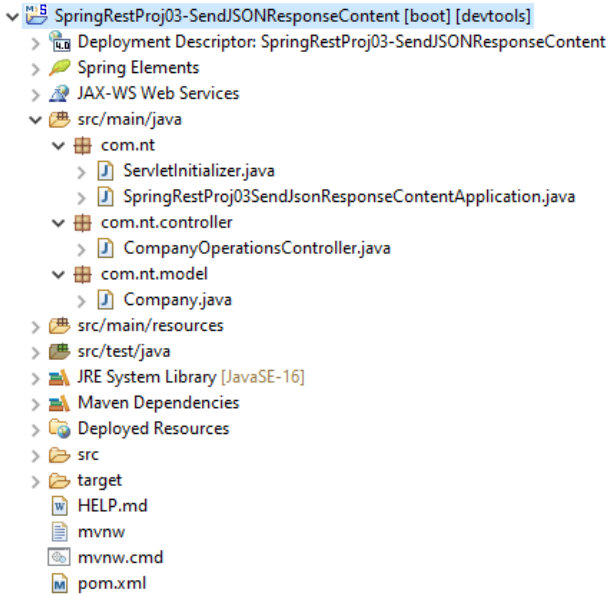
8. NTSPBMS615-June 7th-2022-Working with JSON format data in @RestController

=>HTML is given to display the data where XML and JSON is given to define/describle data..
=>Html displays the data by applying styles on it where as the XML or JSON defines the data by having structure supporting the global formats for data exchange.

=> if the generic in ResponseEntity<T> is other than String (any other object) in the method of @RestController then the method returned object data will be converted into JSON content automatically..

Example App
==========

=> create spring starter project adding
the following starters (web , devtools ,lombok api)



=> add entries in application.properties

```
#Embedded Server port number
server.port=4041

# Context path
server.servlet.context-path=/RestApp03
```

=> Develop Model class having lombok api annotations

```
//Company.java
package com.nt.model;
import lombok.AllArgsConstructor;
import lombok.Data;
@Data  // lombok api annotation
       //giving  setters, getters , toString,hashCode, equals(-) and etc..
@AllArgsConstructor
public class Company {
        private  String name;
        private  String location;
        private  Integer size;
        private  Double turnOver;
        private  String category;
}
```

=> Develop the RestController class

```
//CompanyOperationsController.java
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;
```

```
import com.nt.model.Company;

@RestController
@RequestMapping("/company/api")
public class CompanyOperationsController {

    @GetMapping("/show")
    public ResponseEntity<Company>  showCompanyDetails(){
        Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
        // create and return ResponseEntity obj having model class obj as the response body/content
        return new ResponseEntity<Company>(company, HttpStatus.OK);
    }
```

Allows us to take our choice response status code like HttpStatus.OK , HttpStatus.CREATED and etc.. becoz the ResponseEntity object can hold multiple details including response body/content and the status code.

(or)

```
    @GetMapping("/show")
    public Company  showCompanyDetails(){
        // create and return model class obj
        Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
        return company;
    }
```

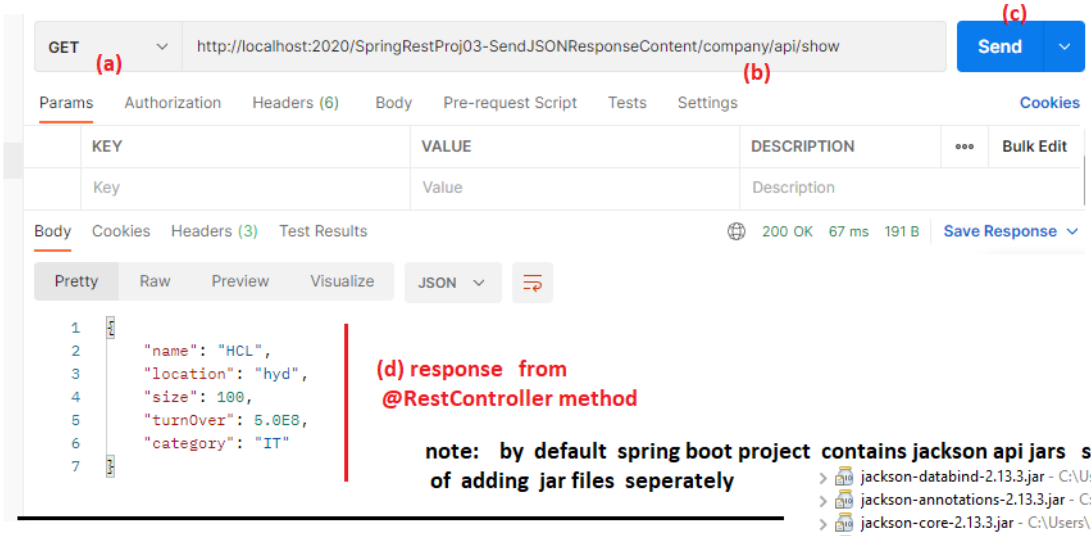
No Possibility of taking our choice response status code.. It always use the default response status code that is HttpStatus.OK (200)

```
}
```

=>In Rest api methods we can direct reeturn types like String , model class, collection and etc.. But recomanded to take becoz we loose ability to put our choice response status code in the http response. it is not

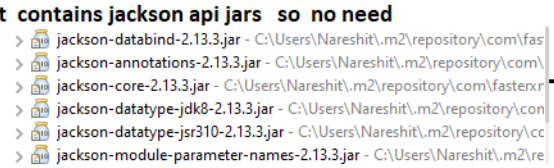
=> run the App either using extenal Tomcat or using embedded Tomcat

=>Test the application using POST MAN tool



(d) response from @RestController method

note: by default spring boot project contains jackson api jars so no need of adding jar files seperately



Making the Rest API method converting the complex Model class obj data to JSON content

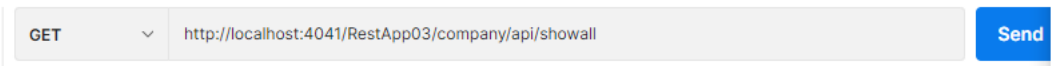
Model classes
==========

```
/Company.java
package com.nt.model;
import lombok.AllArgsConstructor;
import lombok.Data;
@Data  // lombok api annotation
//giving  setters, getters , toString,hashCode, equals(-) and etc..
@AllArgsConstructor
public class Company {
	private  String name;
	private  String location;
	private  Integer size;
	private  Double turnOver;
	private  String category;

}
```

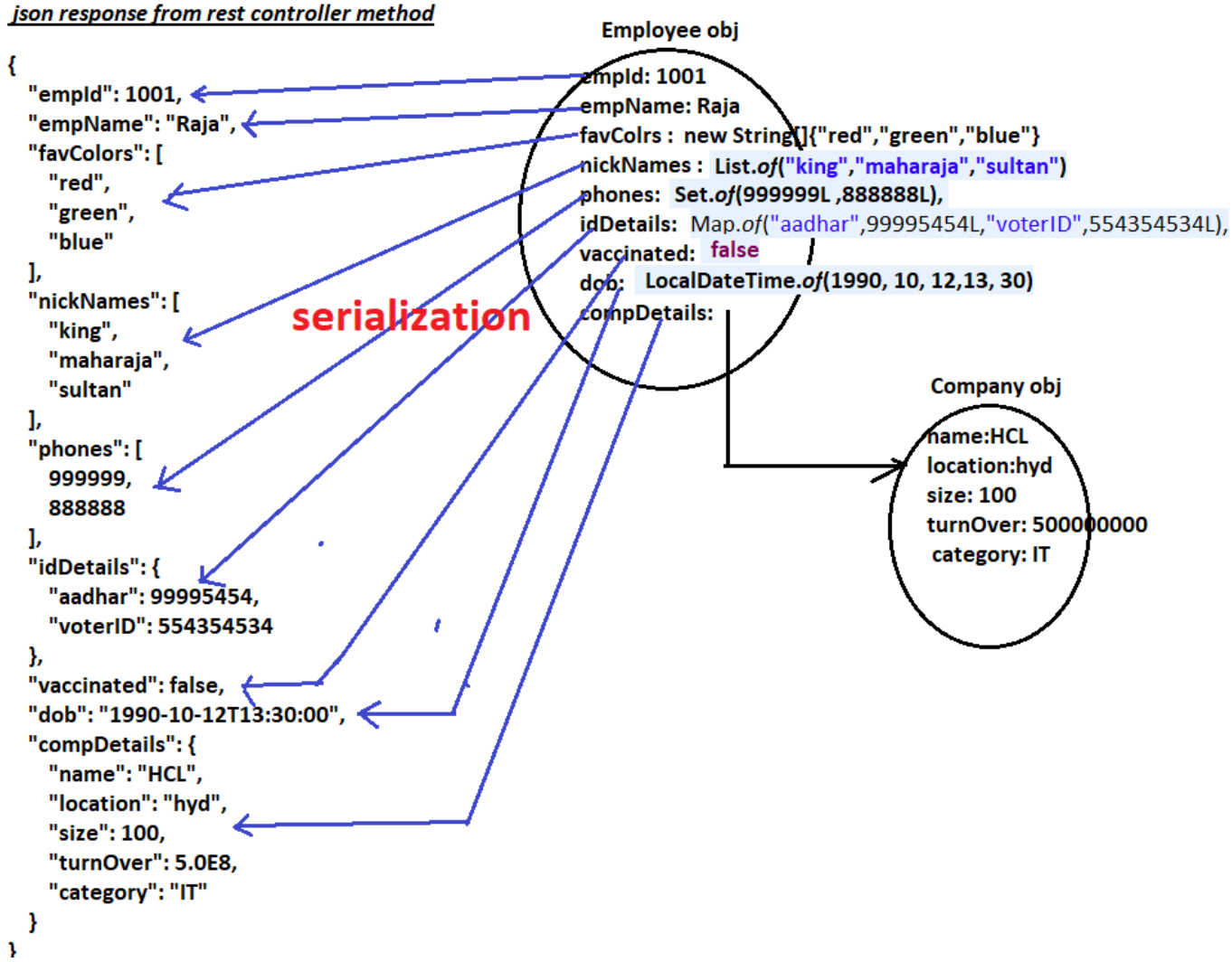
```
//Employee.java (model class)
package com.nt.model;

import java.time.LocalDateTime;
import java.util.List;
import java.util.Map;
import java.util.Set;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class Employee {
	private Integer empId;
	private String  empName;
	private String[] favColors;
	private List<String> nickNames;
	private  Set<Long> phones;
	private  Map<String,Long> idDetails;
	private  boolean  vaccinated;
	private  LocalDateTime dob;
	//HAS-A property
	private  Company    compDetails;
}
```

//Rest Controller

```
@RestController
@RequestMapping("/company/api")
public class CompanyOperationsController {

	@GetMapping("/showall")
	public  ResponseEntity<Employee>  showCompanyDetails(){
		Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
		Employee emp=new Employee(1001,
					"Raja",
					new String[] {"red","green","blue"}, //java 9 features
				List.of("king","maharaja","sultan"),//java 9 features
				Set.of(999999L ,888888L), //java 9 features
				Map.of("aadhar",99995454L,"voterID",554354534L),  //java 9 features
				 false,
				 LocalDateTime.of(1990, 10, 12,13, 30),
				 company);

		// create and return ResponseEntity obj having model class obj as the response body/content
		 return new ResponseEntity<Employee>(emp,HttpStatus.OK);
	}
}
```



json response from rest controller method



The http request header "Accept"

==========

=> This request header "Accept" tells the target server and web comp/Rest api that is expecting the response content (body of the response) having certain MIME Type.. The default value */* i.e the client accepts any MIME type content sent by the server or web comp or rest api

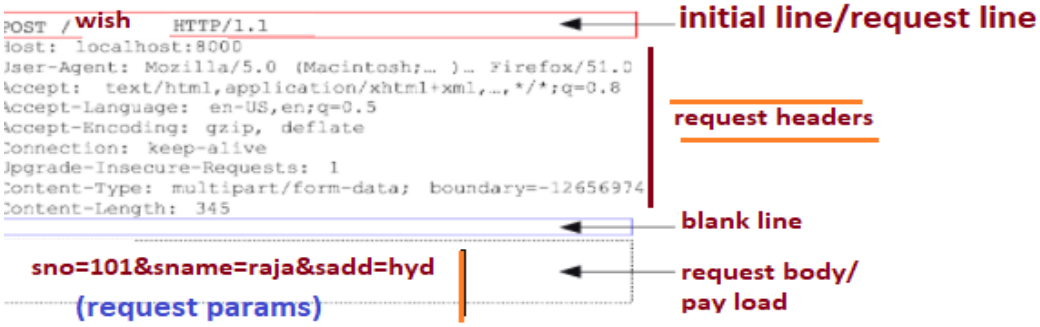
=> if we place the "Accept" rquest header value as the "accept/xml" then it makes the target servlet/web comp / rest api sending only xml content as the response content /body.

=> request headers are part http request structure carrying more info or addtional info about client to server.. The request header names are fixed but the values will be changed based client type and setup.

=> List of request headers are

"Accept" , "Accept-Language" , "referer" , "Accept-Encoding" , "Cookie" , " contentType", "contentLength", and etc..

Example Http request structure



SOAP over Http Request

What is the difference b/w request headers and request params

request headers	request params
a) part of every request structure carrying info about client app	a) carries enduser supplied input vlaues along with request (form data)
b) Every request contains request headers	b) request params are optional in the request
c) request header names are unique and fixed but the values will be changed based on the client app and its type we use	c) request param names are user-defined (programmer choice) and multiple request params can have same name
d) request header values stored in the request object can be read using req.getHeader(-) method	d) request param values stored in the request object can be read using req.getParameter(-) method
(e) request header names and values will be maintained in the request structure as key: value pairs	(e) request params can be there in any format i) as query string appended to the url (IN GET,HEAD mode requests) ii) as query String in request body (IN POST mode request) iii) as Xml content in request body (IN SOAP Over Http) v) as xml/json content in rquest body (IN Restfull webservices)

Proecure to make Spring Rest Based Rest API comp/ server or publisher or provider App sending the b.method provided java class obj as xml conent to client app

---

step1) make sure that Rest API comp developed as Restcontroller having b.method taking the return type as the ResponseEntity<T> having other thatn String as the generic type.

```
//CompanyOperationsController.java
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;

import com.nt.model.Company;

@RestController
@RequestMapping("/company/api")
public class CompanyOperationsController {

	@GetMapping("/show")
	public  ResponseEntity<Company>  showCompanyDetails(){
		Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
		//  create and return ResponseEntity obj having model class obj as the response
body/content
		return  new ResponseEntity<Company>(company, HttpStatus.PARTIAL_CONTENT);
	}

}
```

step2) make suere that following starters and maven dependecies are added to the Project

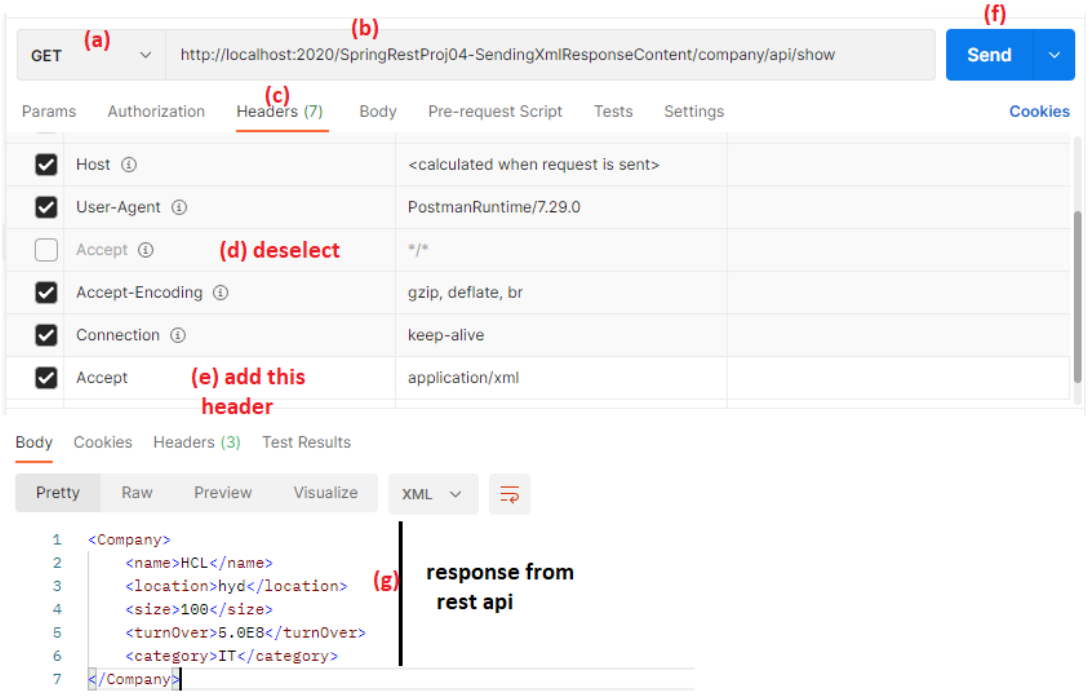
starters : spring web , dev tools , lombok api

seperate maven dependencies : jackson-dataformat- xml (search in mvnrepository.com) and place in pom.xml file

```
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml -->
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
  <version>2.13.3</version>
</dependency>
```

step3) Run the rest api or provider or server or publisher app

step4) Test the above Rest api from postman by sending request having "Accept" header with "application/xml " value



Q) In the above setup if we take Accept request header value as the */* then what happens

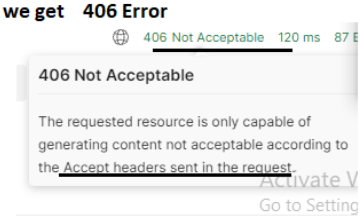
Ans) The default Object to JSON content convertion takes place and the JSON response comes to the Client App

Q) In the above setup if we take Accept request header value as the application/json then what happens?

Ans) application requested the object JSON convertion takes palce and JSON response comes to Client App

Q) In the above setup (example App) if we donot add "jackson-dataformat-xml" jar file or dependency then what happends?

Ans) we get 406 Error



Q) In the above setup (example App) if we donot add "jackson-dataformat-xml" jar file or dependency and Accept header vlaue is */* then what happends?

Ans) JSON response body goes to client App

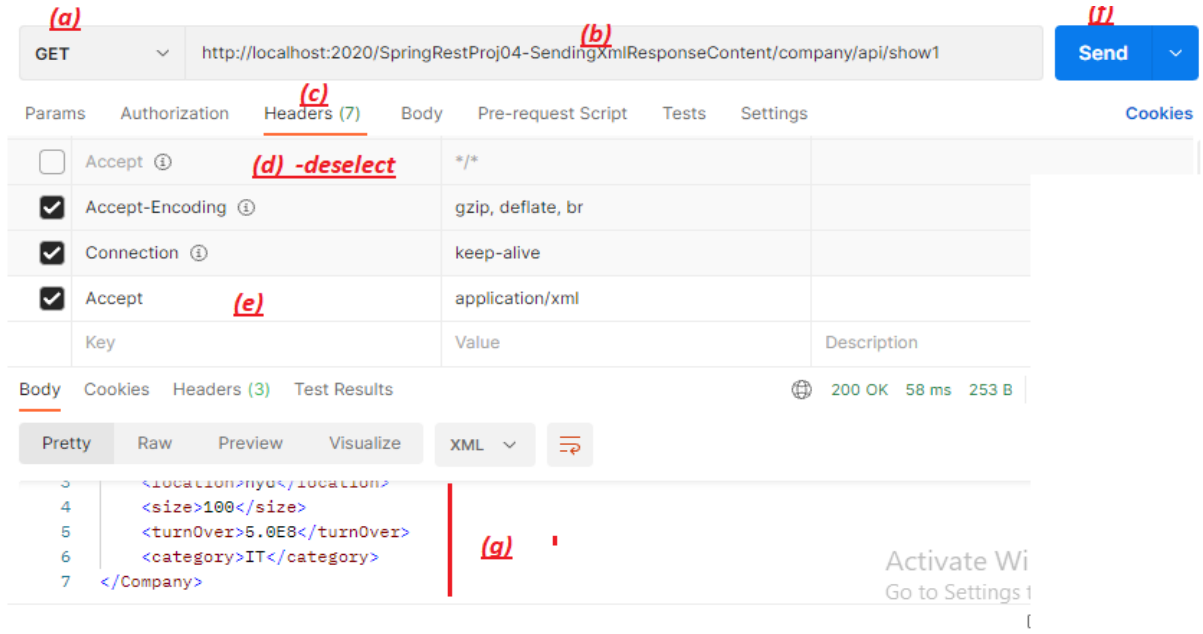
Q) can we send XML response and JSON reponse from one RestController /Rest API?

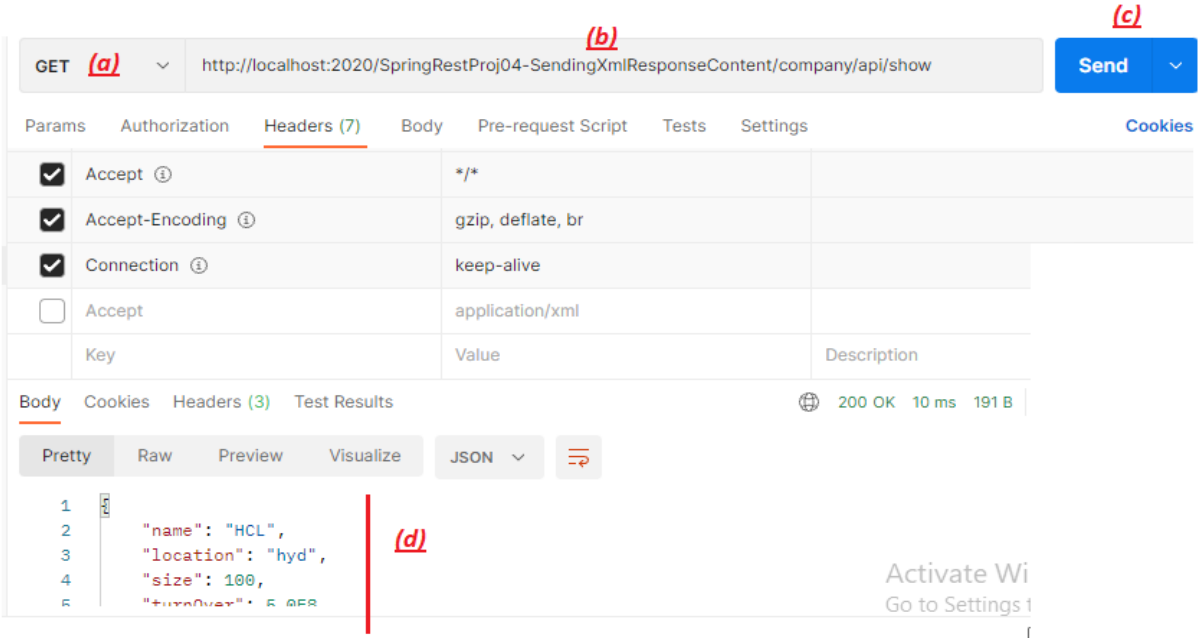
yes) yes possible take two differnet methods in one @RestController class sending two different formats of response

//RestController class

==========

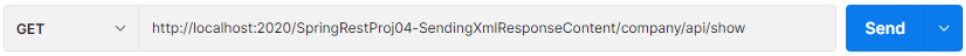
```
@RestController
@RequestMapping("/company/api")
public class CompanyOperationsController {

	@GetMapping("/show")
	public  ResponseEntity<Company>  showCompanyDetails(){
		Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
		// create and return ResponseEntity obj having model class obj as the response body/content
		 return  new ResponseEntity<Company>(company, HttpStatus.OK);
	}


	@GetMapping("/show1")
	public  ResponseEntity<Company>  showCompanyDetails1(){
		Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
		// create and return ResponseEntity obj having model class obj as the response body/content
		 return  new ResponseEntity<Company>(company, HttpStatus.OK);
	}

}
```



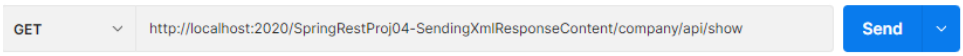


Using single method in @RestController and giving request to that method with or with out "Accept" (application/xml) header value

```
@RestController
@RequestMapping("/company/api")
public class CompanyOperationsController {

        @GetMapping("/show")
        public  ResponseEntity<Company>  showCompanyDetails(){
                Company company=new Company("HCL", "hyd", 100,500000000.0, "IT");
                //  create and return ResponseEntity obj having model class obj as the response body/content
                 return  new ResponseEntity<Company>(company, HttpStatus.OK);
        }
}
```



gives json response



Accept : application/xml

gives xml response

9. NTSPBMS615-June 10th-2022-Working @RequestBody-@ResponseBodyAnnotations

=> Sending the MIME type/content type along with the "Accept" request header and asking provider/ pubblisher /server app of RestApI comp to give cetain MIME type of content in the response body is called called "Content Negotiation" (we are actually negotiating with Provider/Server/Publusher comp being from Client app)

[what we have done in yesterday's app is called Content negotiation]

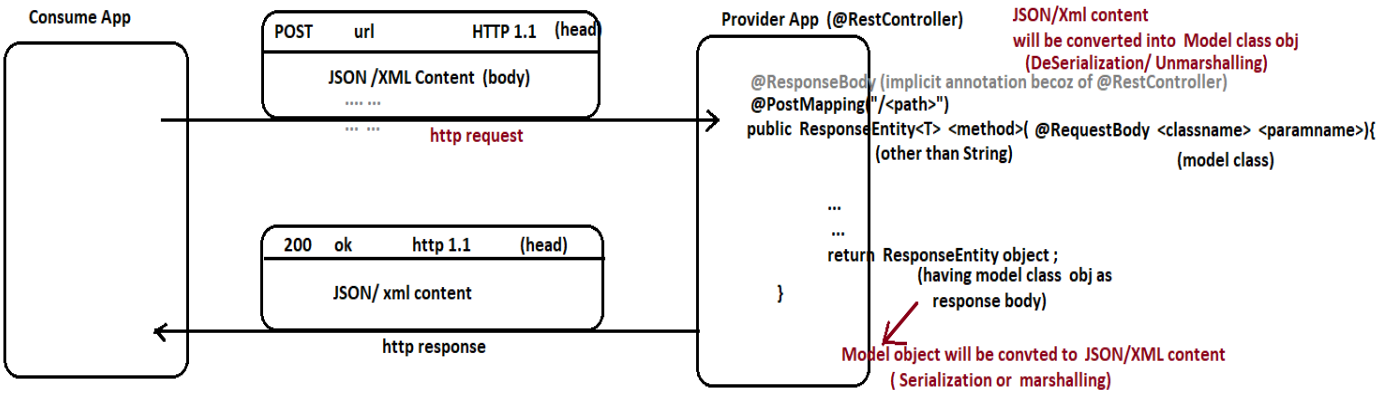
Media Type annotations

==========

=>@RequestBody and @ResponseBody annotation are called media type annotations.

@ResponseBody :: Capable of connverting @RestController method generated object data into JSON/XML content (@ResonseBody will be applied automatically on every method of @RestController class)

@RequestBody :: if the http request structure contains xml /json content , then that content can be converted into Model class obj (java bean class obj) using this @RequestBody Annotation



@RequestBody performs DeSerialization/unmarshalling activities while recieving content from body of the http request

@ResponseBody performs Serialization/Marshalling activities while putting the content into http response from ResponseEntity obj

note:: if xml is involved in the request body or response body then we need to add jackson-dataformat-xml jar file to the Project otherwise not required..

When @RequestBody perform Deserialization/Unmarshalling activity the following matchings should be there

for JSON to Model class obj converstion (DeSerialization)

==========

a) keys count in json content must match with properties count in model class
if not matched then the extra property or extra key will be ignored

b) the key names in json content and property names in model class must match
if not matched model class properties will hold default values .. if few are matched and other few are not matched then data binding takes place among the matched properties.

c) The values of keys must be compitable to store in the properties of model class object.

JSON Document

```
{
 "sno" : 101,
 "sname":"raja",
 "avg": 67.88
}
```

Model class

```
@Data
public class Student{
 private Integer sno;
 private String sname;
 private Double avg;
 }
```

Student class obj(stud)

sno:101
sname: raja
avg: 67.88

@RequestBody creates the Model class obj having param name

```
@PostMapping("/register")
public ResponseEntity<Student> registerStudent(@RequestBody Student stud){
 return new ResponseEntity(stud,HttpStatus.OK);
 }
```

Example App

==========

=> create starter Project adding spring web, devtools , lombok api and jackson-dataformat-xml (required only while dealing with xml as request body /response body)

=> Develope model class adding lombok api support

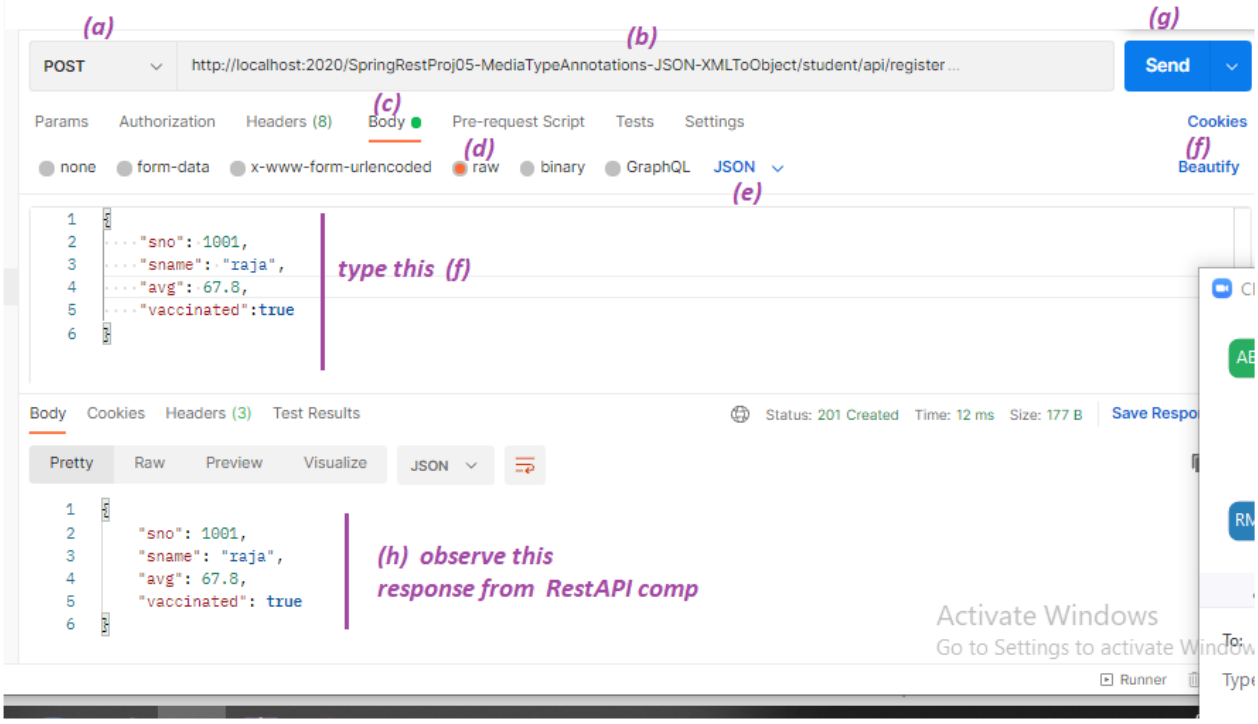
```
//Student.java
package com.nt.model;

import lombok.Data;

@Data
public class Student {
	private Integer sno;
	private String sname;
	private  Double avg;
	private boolean vaccinated;
}
```

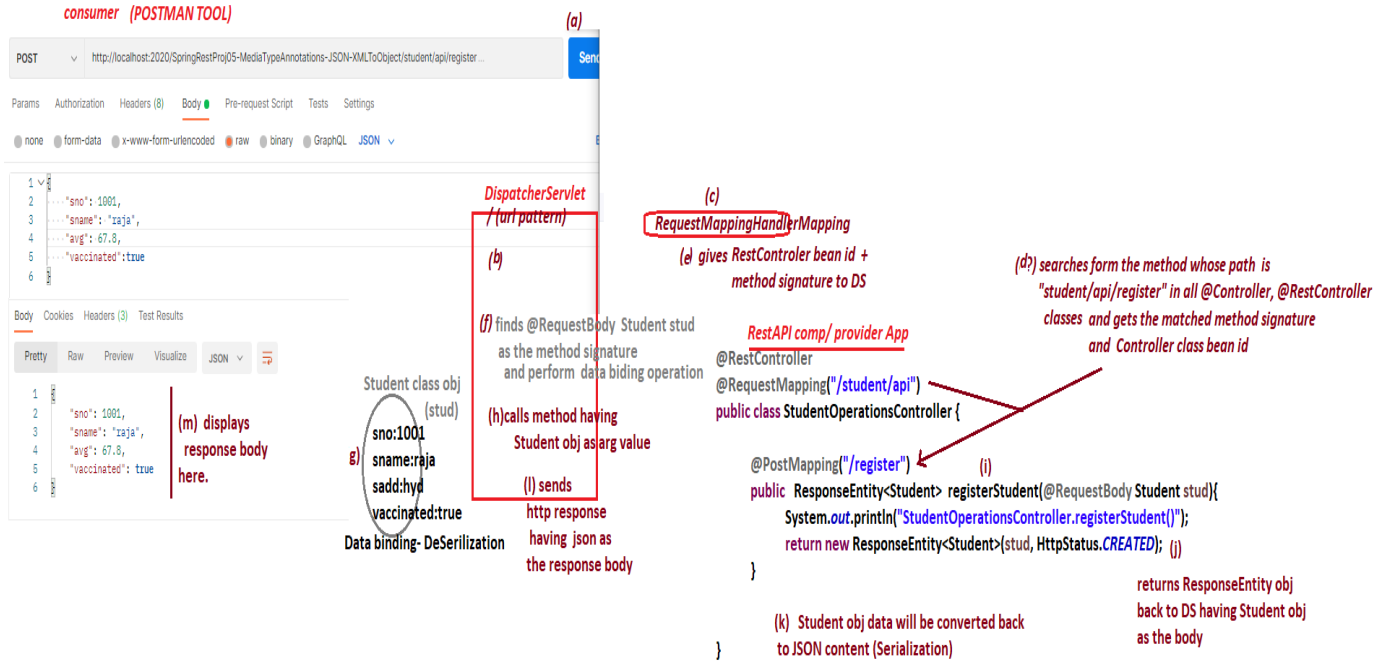
=> Develop the RestAPI component having @XxxMapping methods

```
//StudentOperationsCotroller.java
package com.nt.rest;

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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.nt.model.Student;

@RestController
@RequestMapping("/student/api")
public class StudentOperationsController {

	@PostMapping("/register")
	public  ResponseEntity<Student>  registerStudent(@RequestBody Student stud){
		System.out.println("StudentOperationsController.registerStudent()");
		return new ResponseEntity<Student>(stud, HttpStatus.CREATED);
	}
}
```

=> Run the App into external Tomcat server.

=> send request from POSTMAN





Converting the xml content of consumer supplied http request body to Model class objs

==========

(performing the unmarshalling activity)

=> Take the previous project and make sure that "jackson-dataformat-xml" dependency is added

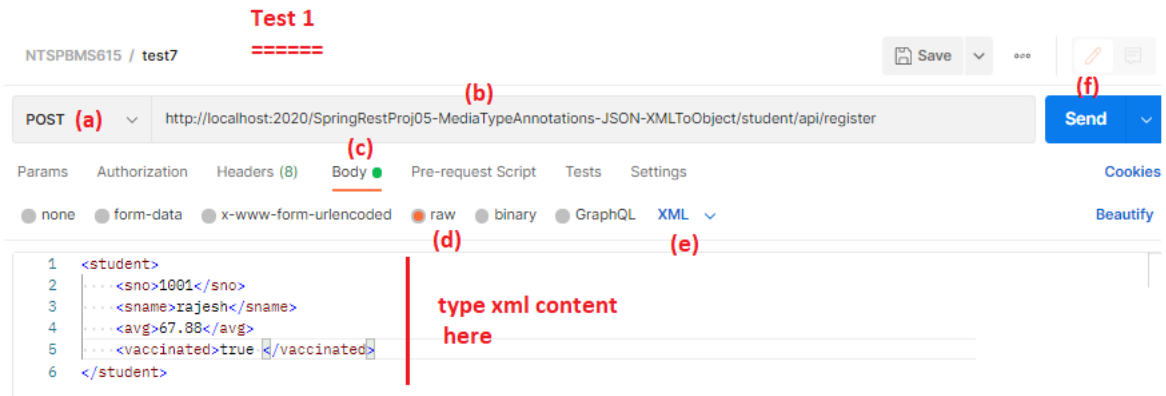
```
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml -->
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
  <version>2.13.3</version>   (placing version is optional)
</dependency>
```

=> Develop the @RestController class having b.method with @RequestBody annotation in the param

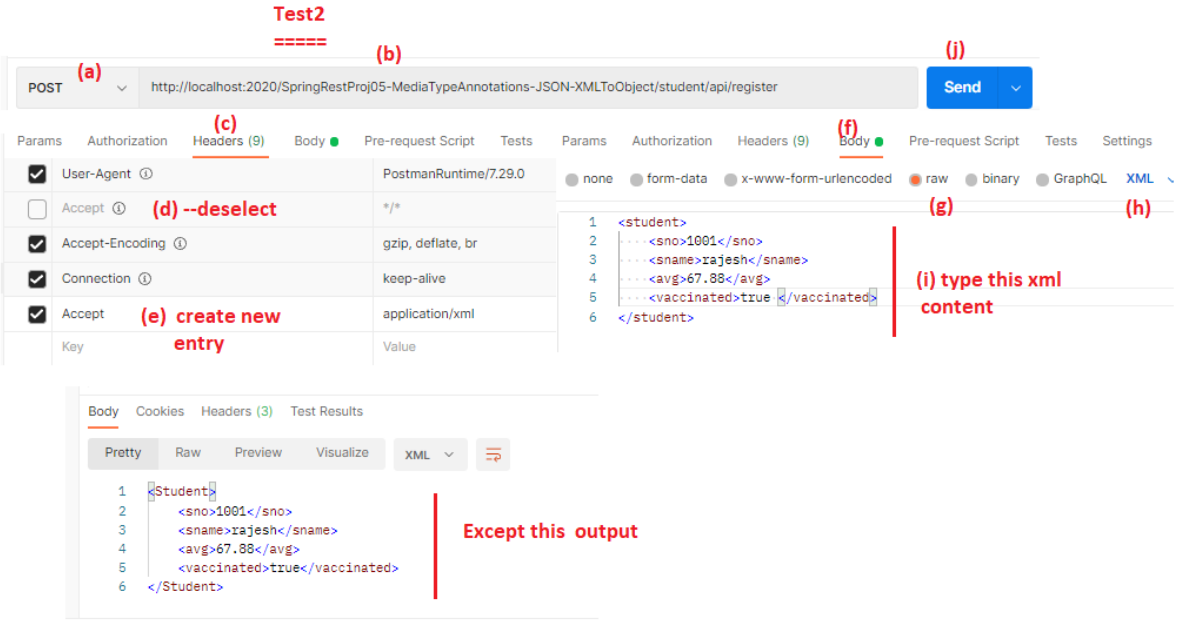
```
//StudentOperationsCotroller.java
package com.nt.rest;

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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.nt.model.Student;

@RestController
@RequestMapping("/student/api")
public class StudentOperationsController {

    @PostMapping("/register")
    public  ResponseEntity<Student>  registerStudent(@RequestBody Student stud){
        System.out.println("StudentOperationsController.registerStudent()");
        return new ResponseEntity<Student>(stud, HttpStatus.CREATED);
    }

}
```

=> Run the app to deploy into web server

=> Give request from POSTMAN having xml tags content as the request body /payload content



10.NTSPBMS615-June 14th-2022- Sending Complex Json Data and converting to objs using @ReponseBody



=>if request body is xml content and rest controller method wants to convert that content into Model java class obj using @RequestBody but "jackson-dataformat-xml" jar file is not added then we get 415 Error.
(unsupported mediatype)

=> if the request body is empty or other than xml content .. but the request header "Accept" is taken as application/xml and "jackson-dataformat-xml" jar file is not added then we get 406 error
(Not acceptable)

=> if @Requestbody fails to convert xml content of rquest body into JAva class object then we get 415 Error (unmashalling is failed)
=> if @Response fails to convert JAva class object data into xml content then we get 406 Error (marshalling is failed)

=>Problem in converting the input xml content to Java class obj using @RequestBody then 415 error
=>@ResponseBody is getting problem to content Java object data into xml content then we get 406 error.

=> while sending json data along with the request to convert into model class obj using @RequestBody .. if u send {} (empty json doc) then the Model class obj will be created using 0-param constructor having default vlaues (The JVM supplied default values like (0, 0.0 ,false, null ande etc..)

=> while sending xml data along with the request to convert into model class obj using @RequestBody .. if u send <student> </student> (empty parent tags with out sub tags) then the Model class obj will be created using 0-param constructor having default vlaues (The JVM supplied default values like (0, 0.0 ,false, null ande etc..)

Q) GET mode requeat can carry data .. POST mode request also can carry data what is difference b/w both data?

a) => GET mode req data is not for giving data to server i.e it is not for insert/updating in the server it is actually passing query inputs while gathering data from the server.

=> Asking web app/rest api comp to search results for "seven wonders"
(this query data to get results data from server)

=> GEt mode request always gets/fetches data from server ...In that process if u want given querying inputs then those inputs will be sent along with the GET request request as query String appended to the url

http://localhost:2525/RestApp04/student/api/showAll? type=doctor
http://localhost:2525/RestApp05/emp/api/all?salary=100000

=>POST mode request sends data to server and that data will be inserted or updated in the Server Application (web applicatiom/Rest api)
So we can say POST mode request posts data.. This data always go to server as the request body/payload i.e this data does not appear as query string data appened to the reuquest url

http://localhost:2525/RestApp04/student/register

(it carries sno=101&sname=raja&avg=35.6) as request body/payload

is

=> GET mode request carried data not secured/safe data and carray max of 2kb

=> POST mode reuest carried data is secured /safe data and can carry unlimited data.

=> GET mode request is idempotent.. i.e safe to repeat the request

=> POST mode request is not idempotent.. i.e not safe to repeat the request

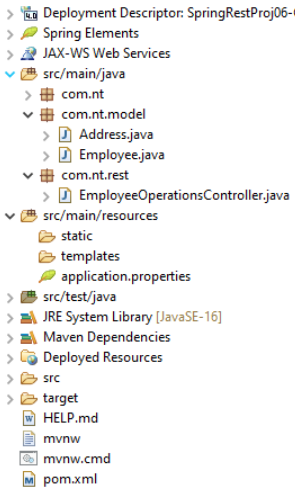
Converting Complex JSON data into Complext java objs using @Request Body

Complex data in JSON is 1D array and 2D Array Data

1D Array (array/list/set) :: variable : [val1, val2,val3 ...]
2D Array (Map Collection) : variable : {"key1": val1, "key2": val2,.... }
2D Array (Has-A Property) : variable : {"subproperty1": val1, "subproperty2": val2,.... }

=>we can use @RequestBody to convert given JSON/XML data to java objs ... (Model class objs)
=> we can use @ResponseBody(that comes implicitly with @RestController) to convert JAva objects data (Model class objs data) to JSON/XML data.

Example App



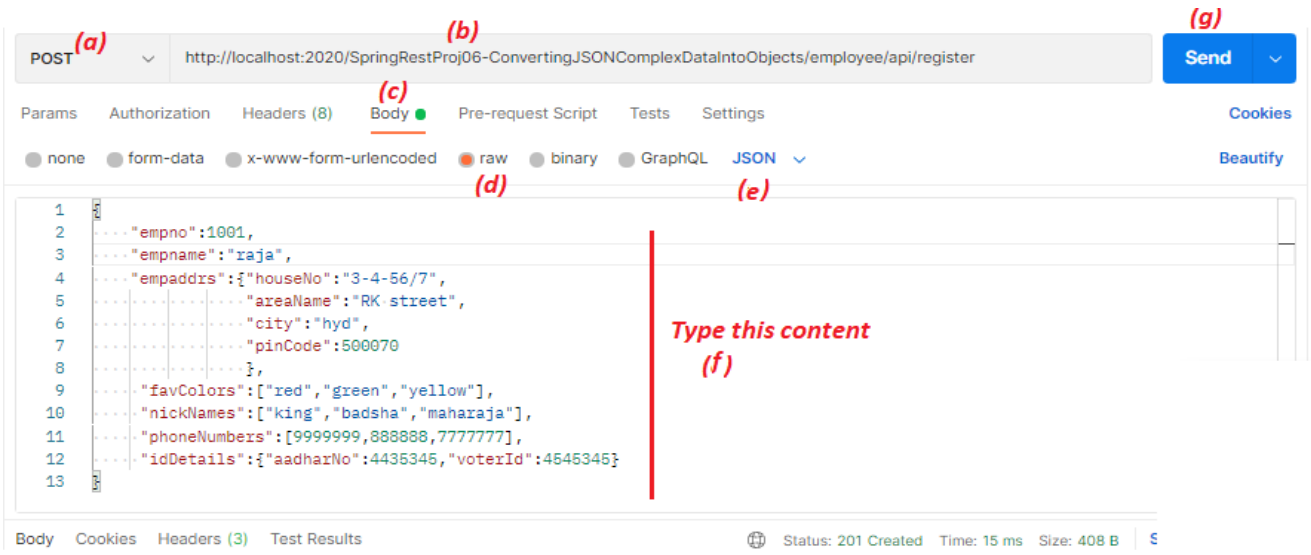
Model classes

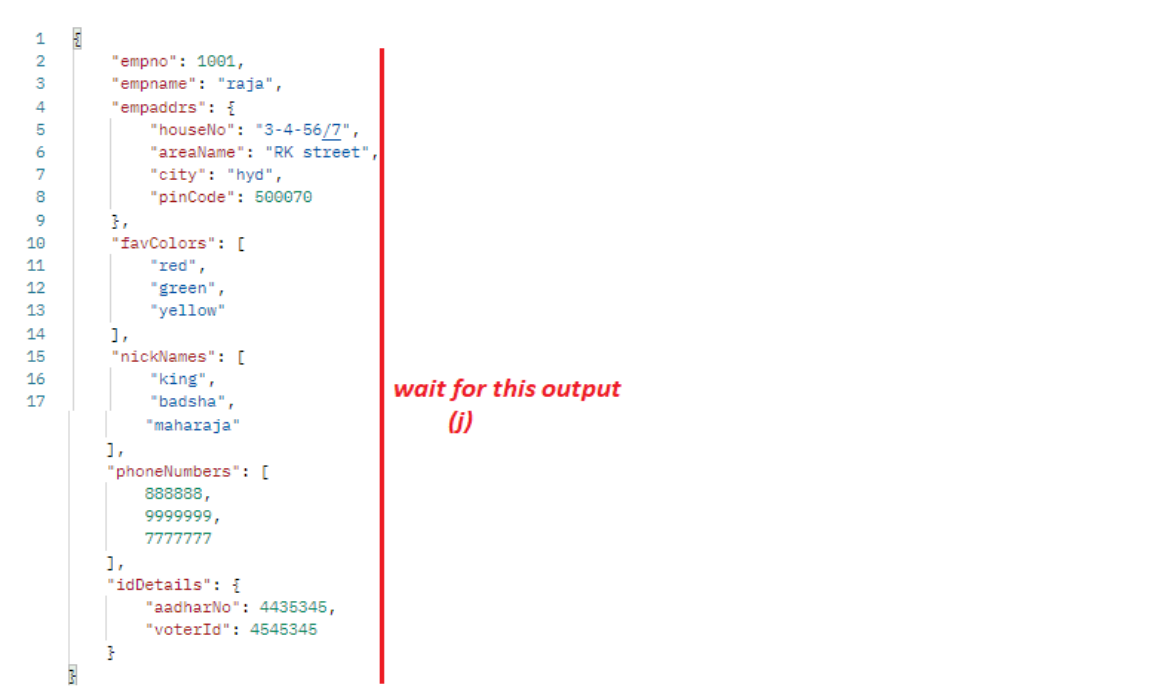
```
//Employee.java
package com.nt.model;

import java.util.List;
import java.util.Map;
import java.util.Set;

import lombok.Data;

@Data
public class Employee {
        private  Integer empno;
        private  String empname;
        private  Address  empaddrs;  // HAS- property 2D Array
        private  String[]  favColors;  //1D array
        private  List<String> nickNames;  // 1D array
        private  Set<Long> phoneNumbers;  // 1D Array
        private Map<String,Long> idDetails;  // 2D Array
}
```

```
//Address.java
package com.nt.model;

import lombok.Data;

@Data
public class Address {
        private String houseNo;
        private  String areaName;
        private  String city;
        private  long  pinCode;


}
```

RestController

```
//EmployeeOperationsController.java
package com.nt.rest;

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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.nt.model.Employee;

@RestController
@RequestMapping("/employee/api")
public class EmployeeOperationsController {

        @PostMapping("/register")
        public ResponseEntity<Employee>  registerEmployee(@RequestBody Employee emp){
                return new ResponseEntity<Employee>(emp,HttpStatus.CREATED);
        }
}
```





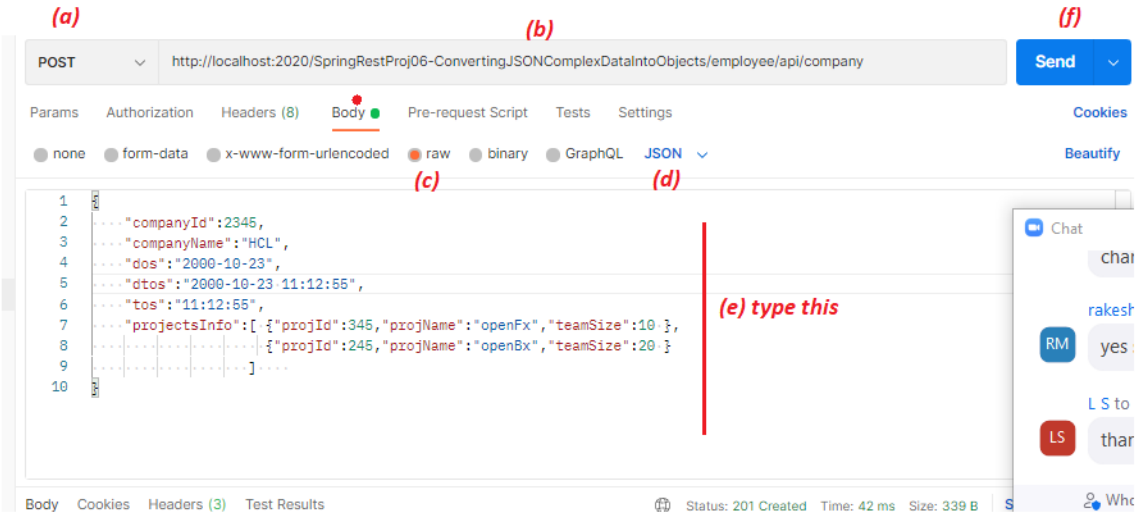
Converting List<model> of JSON data to Java model class obj using @RequestBody .. Also converting JSON Date, DateTime, Time values to LocalDate, LocalDateTme ,LocalTime object

Example App
==========

```
//Project.java
package com.nt.model;

import lombok.Data;

@Data
public class Project {
 private Integer projId;
 private String projName;
 private  Integer teamSize;
}
```

```
//Company.java
package com.nt.model;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

@Data
public class Company {
	private Long companyId;
	private String companyName;
	@JsonFormat(pattern = "yyyy-MM-dd")  //optional for this pattern
	private  LocalDate dos;
	@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")  //option for this
pattern
	private LocalDateTime dtos;
	private  List<Project> projectsInfo;
	@JsonFormat(pattern = "HH:mm:ss")  //option for this pattern
	private  LocalTime tos;

}
```

```java
//EmployeeOperationsController.java
package com.nt.rest;

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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.nt.model.Company;
import com.nt.model.Employee;

@RestController
@RequestMapping("/employee/api")
public class EmployeeOperationsController {

	@PostMapping("/company")
	public ResponseEntity<Company>  registerCompany(@RequestBody Company company){
		System.out.println("EmployeeOperationsController.registerCompany()::::"+company);
		return new ResponseEntity<Company>(company,HttpStatus.CREATED);
	}


}
```



(g) wait for this output

=>if we give wrong json content .. along with the request then the @RequestBody generates 400 error (Bad Request)

11.NTSPBMS615-June 15th-2022- Passing input values to RestAPI comp as request params

Passing Data as request params
==========

=> GET Mode request can not carry data as request body becoz the GET mode request structure does not contain body in the request. so we can pass data in GET Mode reuquest in two ways

(a) As request params placed in query String of the url
eg: url?param1=val1&param2=val2&param3=val3

(b) As path variables (Latest)
eg: url/<value1>/<value2>/<value3>/....

=>The {<key>} will be given in the @GetMapping along with the request path
@GetMapping(<path>/{key1}/{key2}/...)

=> both spring MVC and spring rest Supports request params where as only Spring rest supports path variables

=> All web tehnologies belogining to both java and non-java env.. support request params
eg:: servlet,jsp , sturts, spring mvc , spring boot mvc, jsf , php, asp.net and etc..

=> Only restfull services programming support the path variables .. So all java and non-java techologies and frameworks that re given form restfull web sevices support path variables..

Conclusion is :: web applications support only request params
restful apis/ comps support botht request params and path variables

Spring Rest= spring mvc ++
spring boot Rest = spring boot MVC ++

So we can use both requesst params and path variables in Spring Rest programming

---

Request params (?param1=val1&param2=val2&param3=val3 ....)
==========

In spring MVC controller class or In spring Rest api class we can take @RequestParam annotation in the methods before the parameters to assign request param values to method params.

syntax1:: @RequestParam("<param -name>") <param type> name

These two names need not match

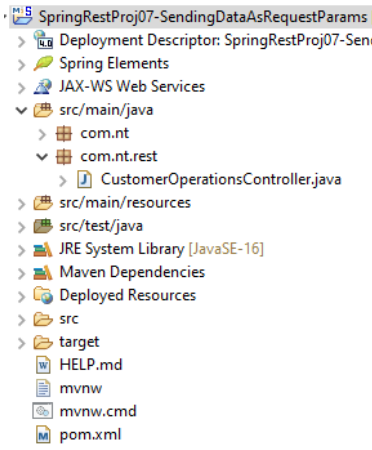
eg:: http://localhost:2525/RestApp01/report?sno=101&sname=raja

```
public <RT> <method>(@RequestParam("sno") Integer no,
                     @RequestParam("sname") String name){
    ,,,
    ,,,,
    }
```

syntax2 :: @RequestParam <param type> name
(the name must match requuest param name)

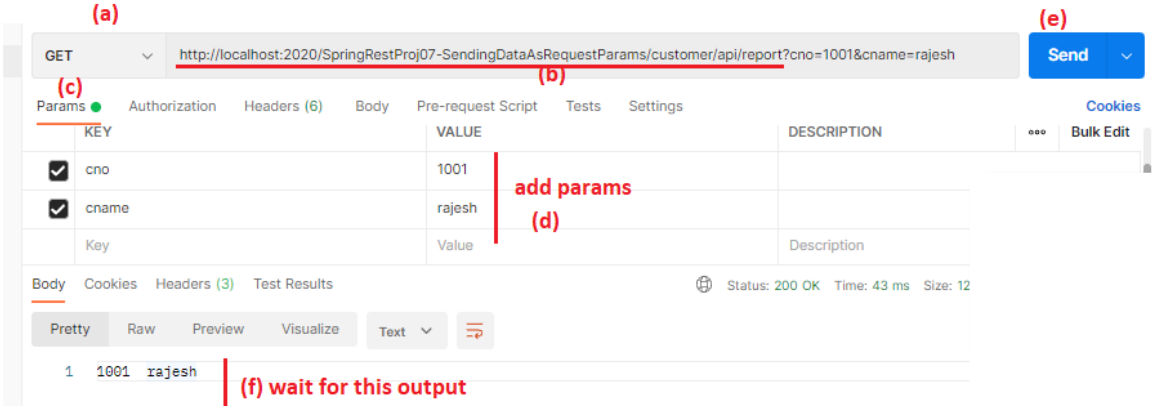
eg:: http://localhost:2525/RestApp01/report?sno=101&sname=raja

```
public <RT> <method>(@RequestParam Integer sno,
                     @RequestParam String sname){
    ,,,
    ,,,,
    }
```




```java
//CustomerOperationsController.java
package com.nt.rest;

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

@RestController
@RequestMapping("/customer/api")
public class CustomerOperationsController {

	@GetMapping("/report")
	public  ResponseEntity<String>  showReport(@RequestParam("cno") int no,
	                                           @RequestParam String cname){
		System.out.println(no+"   "+cname);
		return new ResponseEntity<String>(no+"  "+cname,HttpStatus.OK);
	}

}
```

Testing code from postman



=> if rest api method is designed for 2 request params and we passed more or less params then it will not be a problem (for less params we need to take required=false)

Case1:: http://localhost:2020/SpringRestProj07-SendingDataAsRequestParams/customer/api/report?cno=1001&cname=rajesh&sadd=hyd

extra param

Rest API method

```java
@GetMapping("/report")
	public  ResponseEntity<String>  showReport(@RequestParam("cno") int no,
	                                           @RequestParam String cname){
		System.out.println(no+"   "+cname);
		return new ResponseEntity<String>(no+"  "+cname,HttpStatus.OK);
	}
```

gives 1001 rajesh

Does not raise any error becoz it takes the matched req params

Case2:: http://localhost:2020/SpringRestProj07-SendingDataAsRequestParams/customer/api/report?cno=1001

less params are passed

```java
@GetMapping("/report")
	public  ResponseEntity<String>  showReport(@RequestParam("cno") int no,
	                                           @RequestParam(required = false) String cname){
		System.out.println(no+"  "+cname);
		return new ResponseEntity<String>(no+"  "+cname,HttpStatus.OK);
	}
```

The default vlaue of required is true

gives 1001 null

Does not raise error if we required=false for cname request param

The order of request params and the order of params in method recieving request params need not need to match but the names of types must be compitable

http://localhost:2020/SpringRestProj07-SendingDataAsRequestParams/customer/api/report?cname=rajesh&cno=1010

Rest API method

```
----------------------
@GetMapping("/report")
    public ResponseEntity<String> showReport(@RequestParam("cno") int no,
                                             @RequestParam(required = false) String cname){
        System.out.println(no+"  "+cname);
        return new ResponseEntity<String>(no+"  "+cname,HttpStatus.OK);
    }
```

gives 1010 raja

=>In the following situations we get 400 error (Bad Request)

(a) if given request param value is not compitable to store method param of RestAPI comp
(b) with out taking required=false in @RequestParam annotation .. if we ingore to pass that request param in request url
(c) we have @RequestParam with out specifying param name and method param is not matching with request param name (provided required=true)

url? cname=rajesh

(@RequestParam String name)

Error:400
1. Incompatible data types
2. Missing parameters
3. Missing name
4. Mismatch with the request and api

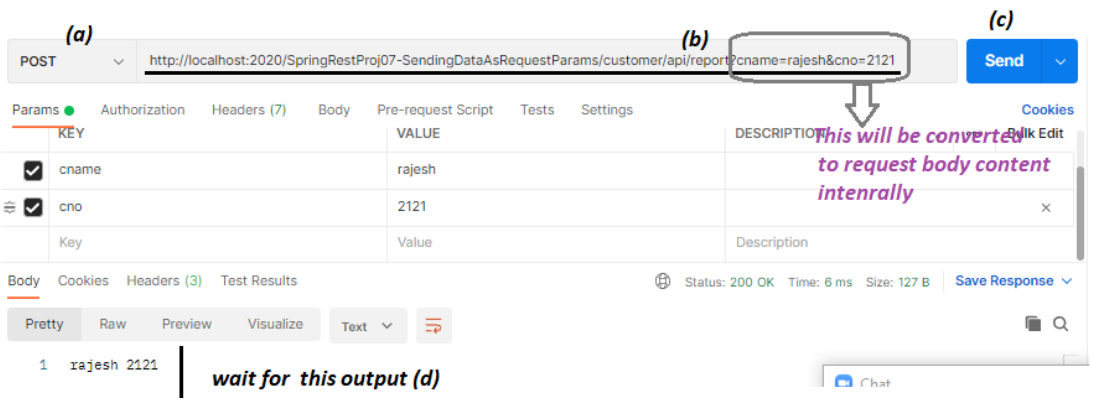
---

if POST mode request form Client App (like POSTMAN) sends data as query String having req params they will be converted into request body content internally .. In this situation the @RequeştParam annotations collects the param values from the Request Body..

Example
==========

```
@RestController
@RequestMapping("/customer/api")
public class CustomerOperationsController {

    @PostMapping("/report")
    public ResponseEntity<String> showReport( @RequestParam String cname,@RequestParam("cno") int no){
        System.out.println(cname+"  "+no);
        return new ResponseEntity<String>(cname+" "+no,HttpStatus.OK);
    }

}
```

These rquest pdrams will be gathered from request body



note:: Only GET , HEAD mode requests do not contain request body.. the remainign all requests (POST,PUT,DELETE,TRACE,OPTIONS,PATCH) contains request body ..

12.NTSPBMS615-June 16th-2022- Passing input values to RestAPI comp as Path variables

Limitations of request params
==========
(a) they increase the length of request url becoz we need to seperate query string
(b) query string needs the seperate syntax like ?param1=val1&param2=val2
(c) we can not pass "&","=" ,"%" symbols as the values of request params
(d) They are not part of basic request url .. so key kill the readability..
and etc..

note ::To overcome these problems take the support of PATH variables

PATH variables
==========
syntax ? <url> or <path>/<value1>/<value2>/....

=>The keys for the values will be defined in
@XxxMapping(-) methods like this
@XxxMapping("/<Path>/{<key1>}/{<key2>}/....)

=> Path variables are useful to minimize the request url /path length becoz path varaible values will be passed directly in url itself

http://localhost:2525/RestApp01/report/101/raja
path variable vlaues

=> we can pass "&","=" and "%" symbol part of data that goes to Restapi comp along with the request
=> No need of having seperate Syntax to pass path varable vlaues in request url
=> No need of specifying keys in request url .. we can directly pass values..
=> request url with query string is not clean url becoz having we need to append query string seperately to the request url to pass the data that to key=value pairs.

http://localhost:2525/RestApp01/report?sno=101&sname=raja (url is not clean becoz of query String)

=> request url with path variables is always clean url becoz the values of path variables will be placed directly in the url

http://localhost:2525/RestApp01/report/101/raja (clean url)
path variable vlaues

=> In web applications we can not use path variables to pass the data.. In Distributed Apps(RestFullAps) we can use both path variables and request params to pass the data..

The methods of RestAPI comp (@RestController class) should use @PathVariable annoation to read values of the path variables.

=>@PathVariable <DataType> paramname (Here the key of path vairable must match with param name)

=>@PathVariable("key") <DataType> paramname ((Here key of path vairable need not to match with param name)

=> The keys for path variables will be defined in @XxxMapping methods along with request path

=>Request path with parth variables contains two parts

a) Static path (fixed and does not change time to time)
b) Dynamic path (will change time to time -- not fixed)

@XxxMapping("/<path>/{<key>}/{<key>}")
static path Dynamic path

eg: @GetMapping("/report/{sno}/{sname}")
static path (fixed path) dynamic path (keys of path variables)
⇩
These key vlaues will change time to time

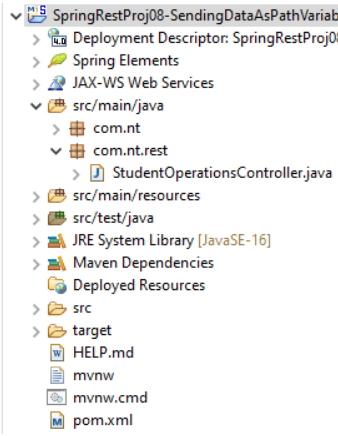
example request urls passing the values

---

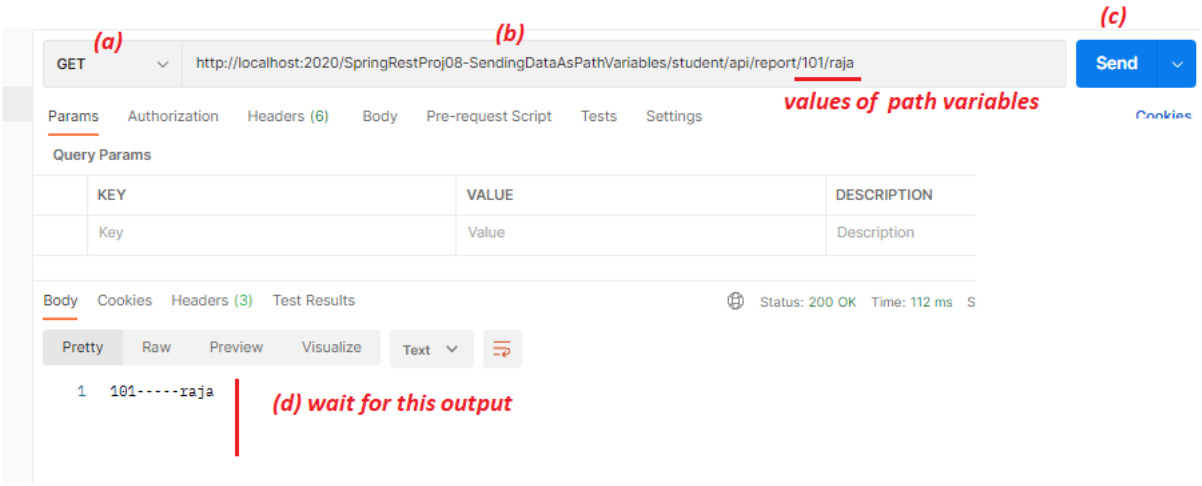
http://localhost:2525/SpringApp01/report/101/raja here sno , sname values are 101 raja
http://localhost:2525/SpringApp01/report/102/suresh here sno , sname values are 102 suresh

The complete RestAPI comp method

```
@GetMapping("/report/{sno}/{sname}")
public ResponseEntity<String> showReport(@PathVariable("sno") int no,
                                         @PathVariable("sname") String name);
```




```
//StudentOperationsController.java
package com.nt.rest;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/student/api")
public class StudentOperationsController {

                                    keys of path variables
    @GetMapping("/report/{sno}/{sname}")
     public  ResponseEntity<String>  showReport(@PathVariable("sno") int no,
                                                @PathVariable String sname){
            System.out.println(no+"<------->"+sname);
            return  new ResponseEntity<String>(no+"-----"+sname,HttpStatus.OK);
    }

}
```



While passing data as query String req params we can chnage the order of the request params that we are passing

http://localhost:2525/RestApp01/report?sno=101&sname=raja //valid
http://localhost:2525/RestApp01/report?sname=raja&sno=101 //valid

both are same

While working with path varibles we can not change the order of passing values to path variables/keys

```
@GetMapping("/report/{sno}/{sname}")
    public  ResponseEntity<String>  showReport(@PathVariable("sno") int no,
                                               @PathVariable String sname){
        System.out.println(no+"<------->"+sname);
        return  new ResponseEntity<String>(no+"-----"+sname,HttpStatus.OK);
    }
```

http://localhost:2525/RestApp01/report/101/raja //valid
http://localhost:2525/RestApp01/report/raja/101 //invalid (gives 400 - Bad request error)

This error has come becoz of "raja" needs store int no (incompitable type)

---

if above context if we take both param of rest api comp method linked with path variables as String params then error will not come .. but wrong values will be stored which bad effect in the b.logic execution

request url :: http://localhost:2020/RestApp01/report/raja/101

values are passed in wrong order

```
@GetMapping("/report/{sno}/{sname}")
    public  ResponseEntity<String>  showReport(@PathVariable("sno") String no,
                                               @PathVariable String sname){
        System.out.println(no+"<------->"+sname);
        return  new ResponseEntity<String>(no+"-----"+sname,HttpStatus.OK);
    }
```

Does not raise error but sno holds raja and sname holds 101 which is not good b.logic execution.

=>While working with query String we pass extra request params in query String they will be ignored

http://localhost:2525/RestApp01/report?sno=101&sname=raja&sadd=hyd

=>It is extra req param becoz no one read it in the web comp or rest api comp .. so it will be ignored

=> Path variables in requuest url does not support to pass extral levels or values in the request url .. if passed they will be taken extra leven in request path and generates 404- Requested resource not found error

Rest API method

```
@GetMapping("/report/{sno}/{sname}")
 public  ResponseEntity<String>  showReport(@PathVariable("sno") int no,
                                            @PathVariable String sname){
        System.out.println(no+"<------->"+sname);
        return  new ResponseEntity<String>(no+"-----"+sname,HttpStatus.OK);
}
```

two levels of dyna path

http://localhost:2525/RestApp01/report /101/raja/hyd

takes it extra level in request path
so 404 error will be generated

=>we can pass "&","=" symobls as the values of path variables as shown below .. but can not pass "/" "%" symobls as the data..

http://localhost:2020/RestApp01/report/101/raja1&= //valid
http://localhost:2020/RestApp01/report/101/raja=&1 //valid
http://localhost:2020/RestApp01/report/101/raja=1 //valid
http://localhost:2020/RestApp01/report/101/r&1 //valid
http://localhost:2020/RestApp01/report/101/r&1/ //valid
http://localhost:2020/RestApp01/report/101/r&1% --invalid (gives 400 -Bad request)
http://localhost:2020/RestApp01/report/101/r&1%/ -- invalid (gives 400 -Bad request)
http://localhost:2020/RestApp01/report/101/raja/// -- invalid (gives 404 -not found)

13.NTSPBMS615-June 17th-2022- More on Path Variables

if u r place more or less levels of data in the request url while working with path variables then we get 404 error (Requested resource not found)

method in @RestController

```
@GetMapping("/report/{sno}/{sname}")
 public  ResponseEntity<String>  showReport(@PathVariable("sno") int no,
                                            @PathVariable String sname){
        System.out.println(no+"<------->"+sname);
        return  new ResponseEntity<String>(no+"-----"+sname,HttpStatus.OK);
}
```

http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/
(gives 404 error -- less levels are given in the path)
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/
(gives 404 error -- less levels are given in the path)
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/raja
(gives 101 ---- raja)
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/raja/hyd
(gives 404 error -- more levels are given in the request path)

=>Though we have taken required=false in @PathVariable annotation ,we still get 404 error if we pass less or more levels in the request path of request urls.. becoz to map request with Rest api comp method the levels matching in request path is mandatory.

```
@GetMapping("/report/{sno}/{sname}")
public  ResponseEntity<String>  showReport(@PathVariable(name = "sno",required = false) int no,
                                           @PathVariable(required = false) String sname){
        System.out.println(no+"<------->"+sname);
        return  new ResponseEntity<String>(no+"-----"+sname,HttpStatus.OK);
}
```

http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/raja
(gives 101 ------- raja)

http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/
(we expect 101 ---- NULL --- but we get 404 error)

http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report
(we expect NULL ---- NULL -- but we get 404 error)

To map ruquest url generated request with rest api comp methods the path levels must match

---

=> if mutliple methods of Rest API comp is having similar path variables or matching path variables then the method that is having more static path matching will get more priority to execute.

Case1::

```
@RestController
@RequestMapping("/student/api")
public class StudentOperationsController {

    @GetMapping("/report/{sno}/{sname}")
     public  ResponseEntity<String>  showReport1(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport1 ",HttpStatus.OK);
    }

    @GetMapping("/report/sno/{sname}")
     public  ResponseEntity<String>  showReport2(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport2 ",HttpStatus.OK);
    }

@GetMapping("/report/{sno}/sname")
 public  ResponseEntity<String>  showReport3(@PathVariable(name = "sno",required = false) Integer no,
                                             @PathVariable(required = false) String sname){
        return  new ResponseEntity<String>("from ShowReport3 ",HttpStatus.OK);
}

@GetMapping("/report/sno/sname")
 public  ResponseEntity<String>  showReport4(@PathVariable(name = "sno",required = false) Integer no,
                                             @PathVariable(required = false) String sname){
        return  new ResponseEntity<String>("from ShowReport4 ",HttpStatus.OK);
}
```

http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/sno/sname
executes showReport4(-,-) method
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/sno/raja
executes showReport2(-,-) method
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/1010/sname
executes showReport3(-,-) method
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/sno/rajesh
executes showReport2(-,-) method
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/rajesh
executes showReport1(-,-) method

Case2::
======

```
@RestController
@RequestMapping("/student/api")
public class StudentOperationsController {

    @GetMapping("/report/{sno}/{sname}")
     public  ResponseEntity<String>  showReport1(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport1 ",HttpStatus.OK);
    }

    @GetMapping("/report/101/raja")
     public  ResponseEntity<String>  showReport2(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport2 ",HttpStatus.OK);
    }

    @GetMapping("/report/sno/sname")
     public  ResponseEntity<String>  showReport3(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport3 ",HttpStatus.OK);
    }


}
```

http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/raja
executes showReport2(-,-)
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/102/rajesh
executes showReport1(-,-) method
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/sno/sname
exeutes showReport3(-,-) method
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/sno/rajesh
400 -- Bad request
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/101/sname
executes showReport1(-,-)
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/3345/sname
executes showReport1(-,-)
http://localhost:2020/SpringRestProj08-SendingDataAsPathVariables/student/api/report/3347/raja
executes showReport1(-,-)

if multuple methods of RestAPI comp contains same request path and same mode including the same global path then we get IllegalState Exception during the application startup itself.

```
@RestController
@RequestMapping("/student/api")
public class StudentOperationsController {

    @GetMapping("/report/{sno}/{sname}")
     public  ResponseEntity<String>  showReport1(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport1 ",HttpStatus.OK);
    }

    @GetMapping("/report/101/raja")
     public  ResponseEntity<String>  showReport2(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport2 ",HttpStatus.OK);
    }

    @GetMapping("/report/101/raja")
     public  ResponseEntity<String>  showReport3(@PathVariable(name = "sno",required = false) Integer no,
                                                 @PathVariable(required = false) String sname){
            return  new ResponseEntity<String>("from ShowReport3 ",HttpStatus.OK);
    }
}
```

```
aused by: java.lang.IllegalStateException: Ambiguous mapping. Cannot map 'studentOperationsController' method
com.nt.rest.StudentOperationsController#showReport2(Integer, String)
to {GET [/student/api/report/101/raja]}: There is already 'studentOperationsController' bean method
com.nt.rest.StudentOperationsController#showReport3(Integer, String) mapped.
```

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

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

c) while passing request param values in the query string of request url the order need not to match

d) if pass more than required request params in query String then we will not get error

e) if we pass less than required request params in query Strung then we get error only when required=false is not taken @RequestParam

f) Does not allow to pass '&' as direct value (we vmust use %26 for that)

g) Supported by both spring MVC and Spring Rest

h) In all web technologies that are there to develop web applicaitons in different domains supports this feature as basic concept to pass data

i) URL is not clean URL (more characters required in the url to pass data)

j) Easy to read and interpet

k) Bit slow while sending data
(legth is bit more)

l) Recomaned to use in non RestFull Apps with

m) Generally used while working GET mode requests becoz data goes as query String

Path variables

a) Syntax to pass data along with request url is
url/<static path>/val1/val2/...
{key}/{key}/.. will be defined in @RestController class inside @XxxMapping(-) annotations

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

e) if we pass less than required path variable values in the request url then we get 404 error

f) Does not allows "/" as value

g) Supported only in Spring Rest

h) supported only in Restfull programming of all domains..

i) URL is clean URL (less characters required in the url to pass data)

j) complex to read to interpet.

k) Bit faster while sending the data
(Length is bit less)

l) very useful in Restful apps..

M) works with all modes of of requests becoz data goes directly from url itself

14.NTSPBMS615-June 18th-2022- MiniProject on SpringRest

==========

Mini Project performing CURD Operations using Spring Rest + Spring Data JPA + POSTMAN tool

==========

=> The methods in @RestController class are technically called as web operations or endpoints or API methods

=> we can write directly b.logic or persistence logic in @RestController web operation methods but not recomanded .. So place b.logic in b.methods of SErvice class and place persistence logics either in DAO class or Repository's In Memory Impl class which allows use to seperate logics in a layered app environment.

=> Always to use @RestController class methods to recieve http requests and to devlier http response and assign b.logics to service class and persistence logics to Repository or DAO class..

=> @RestController class methods are acts Restfull comp/Rest API comp methods provding distributed Access to Clients /consumer using http with xml/JSON content protocol. So always place logics in @RestController class methods to handle recieved requests , to delier generates response and also handle and propagate the exceptions..

=> service class is normal class , service class methods are ordinary java methods .. So they can not handle http request , http responses directly .. So we need one special class who do that work that is @Controller class in spring mvc /spring boot mvc apps and @RestController class in spring rest /spring boot rest Apps.

=> We link @RestController classes with Serivce , DAO/Repository classes for the following reasons

a) To develop the logics as Decoupled logics
b) To avoid fatty methods in @RestController class
c) To develop the apps as the Layered Apps
d) To seperate http requests , http response handling process from b.logics and persistence logics development.
e) Since service , DAO/Repository classes are normal java classes they can be moved other java projects supporting non-invasive behaviour

and etc...

=> Supplying Rest API details and end point details to cosumers is nothing but supplying base url + request paths of methods + path variables info of methods

=> Developing Rest API /API of Restfull webServices programming is nothing but developing @ RestController class as service/Producer/Publisher /Rest API comp talking Service, DAO/Repository classes.

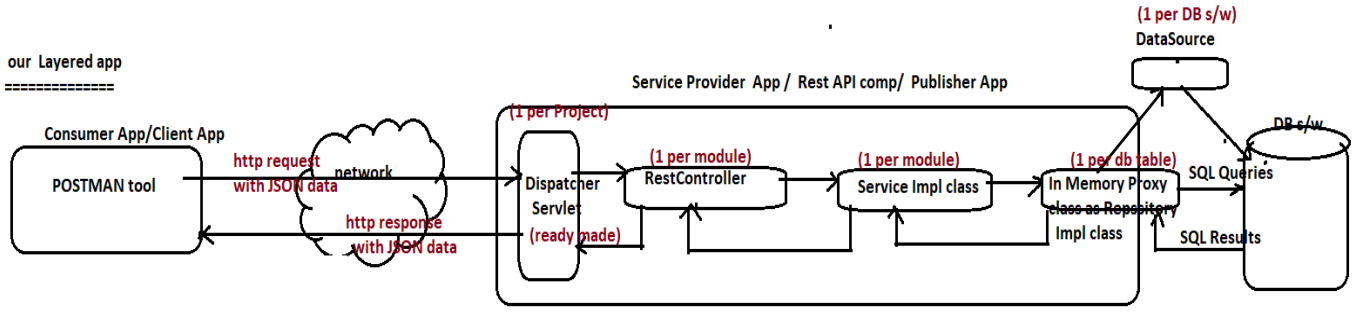
=> The Consumer /Client /Service Client needs API and endpoint details to consume/invoke Rest API/server/ producer/Publisher comp services in Restfull programming

example

baseurl :: http://localhsost:2525/RestApp01/student/api

end point1 details : /report -- GET
path variables of endpoint : {sno} ,{sname} | method1 details

end point2 details : /register - POST
path variables of endpoint : {sno}, {sname} ,{sadd}, ... | method details



Procedure to devleop Spring Rest based Layered Application having @Restcontroller, Serive, Repository interfaces

step1) creeate spring starter Project having the folowing spring boot starters and dependencies

a) spring web b) spring data jpa c) lombok api d) mysql e) devtools

step2) Add the following entries in application.properties for DataSource cfg..
(Here we are using spring boot's bult-in hikari cp DataSource)

application.properties

```
#DataSource cfg (by default it is hikaricp)
# common jdbc properties
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql:///ntspbms615db
spring.datasource.username=root
spring.datasource.password=root
# hikari cp specific properites
spring.datasource.hikari.minimum-idle=10
spring.datasource.hikari.maximum-pool-size=100
spring.datasource.hikari.connection-timeout=60000

# JPA properties
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
```

Q) Why the industry is prefering hikari cp kind of third party jdbc con pools over server managed jdbc con pools though the application is deployable web application or Restfull Apps?

Limitations with server managed jdbc con pool

a) Some server level and sever specific configurations required
b) if the underlying server is changed then we need to change cfgs to use the server managed jdbc con pool
c) Some web servers do not support server managed jdbc con pools then we are forced to use advacned web servers or application servers..

Advantages with third party jdbc con pools like apache dbcp, hikaricp and etc..

a) can be used in both standalone and web appliations , Restfull apps
b) No server specific entries are required
c) Spring boot framework gives hikaricp, apache dbcp2, tomcat cp , oracle UCP related Datasources through AutoConfiguration .. (hikari cp is the best)
d) Server to server to we need not modify the cfgs related to jdbc con pool

step3) Develop Entity class/ Model class having JPA Annotations

```
//Actor.java (model class)
package com.nt.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;

@Data
@Entity
@Table(name="REST_ACTOR")
public class Actor {
```

```
        @Id
        @GeneratedValue
        private  Integer actorid;
        @Column(length=20)
        private  String actorname;
        @Column(length=20)
        private  String category;
        private  Long mobileNo;
    }
```

step4) Create Repository interface

```
//IActorRepo.java
package com.nt.repository;
import org.springframework.data.repository.CrudRepository;
import com.nt.model.Actor;

public interface IActorRepo extends CrudRepository<Actor, Integer> {

}
```

No need of developing impl class for this custom Repository (I) .. Becoz the spring data jpa internally genertes InMemory Proxy class as impl of class this Repository interface and providing definitation for all CurdRepository<T,ID> method with hibernate persistence logic.

step5) Develop Service Interface and SErvice Impl class

```
//IActorMgmtService.java   (service interface)
package com.nt.service;

import com.nt.model.Actor;

public interface IActorMgmtService {
  public  String  registerActor(Actor actor);
}


//ActorMgmtServiceImpl.java   (Impl class)
package com.nt.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.nt.model.Actor;
import com.nt.repository.IActorRepo;

@Service("actorMgmtService")
public class ActorMgmtServiceImpl implements IActorMgmtService {
    @Autowired
    private IActorRepo  actorRepo;

    @Override
    public String registerActor(Actor actor) {
        Actor  actorS=actorRepo.save(actor);
        return "Actor is registered with id value ::"+actorS.getActorid();
    }

}
```

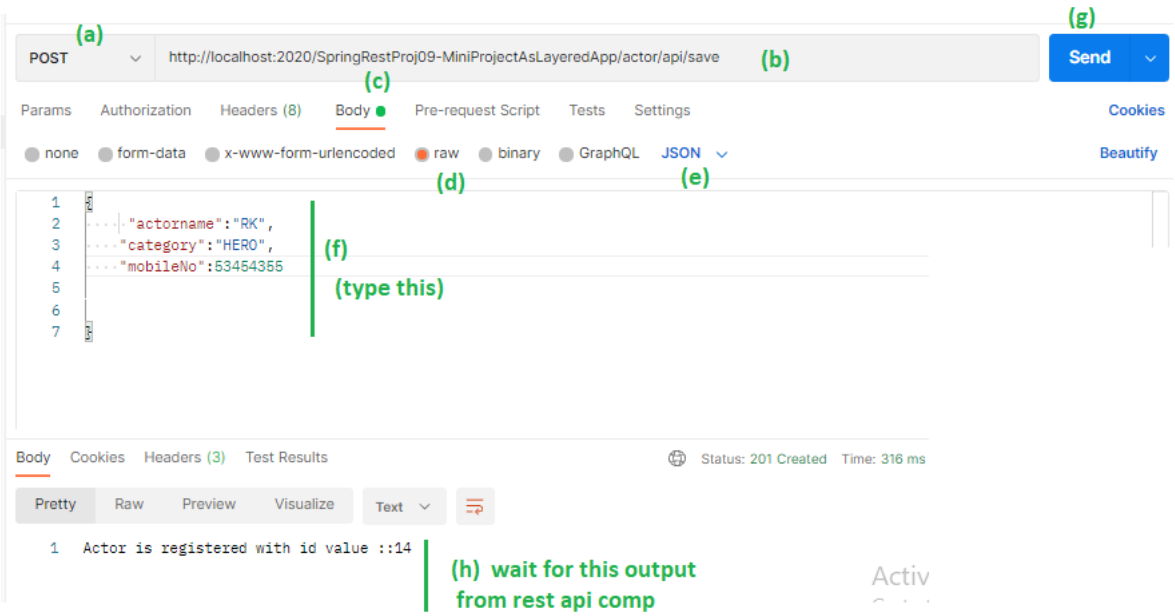
step6) Develop RestAPI comp

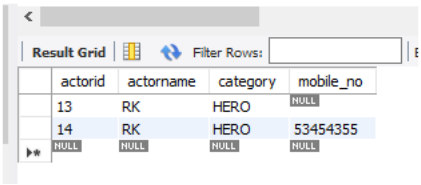
```
//ActorOperationsController.java
package com.nt.rest;

import org.springframework.beans.factory.annotation.Autowired;
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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.nt.model.Actor;
import com.nt.service.IActorMgmtService;

@RestController
@RequestMapping("/actor/api")
public class ActorOperationsController {
	@Autowired
	private  IActorMgmtService service;

	@PostMapping("/save")
	public  ResponseEntity<String>  saveActor(@RequestBody  Actor actor){
		try {
		 //use service
			String msg=service.registerActor(actor);
			return  new ResponseEntity<String>(msg,HttpStatus.CREATED);
		}
		catch(Exception e) {
			e.printStackTrace();
			return new ResponseEntity<String>(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
		}
	}
}
```

step7) Run the rest api comp

step8) Test the application from postman tool





NTSPBMS615-June 22nd-2022- MiniProject on SpringRest-CURD Operations

SpringRest Based Layered application

==========

Performing select all records operation

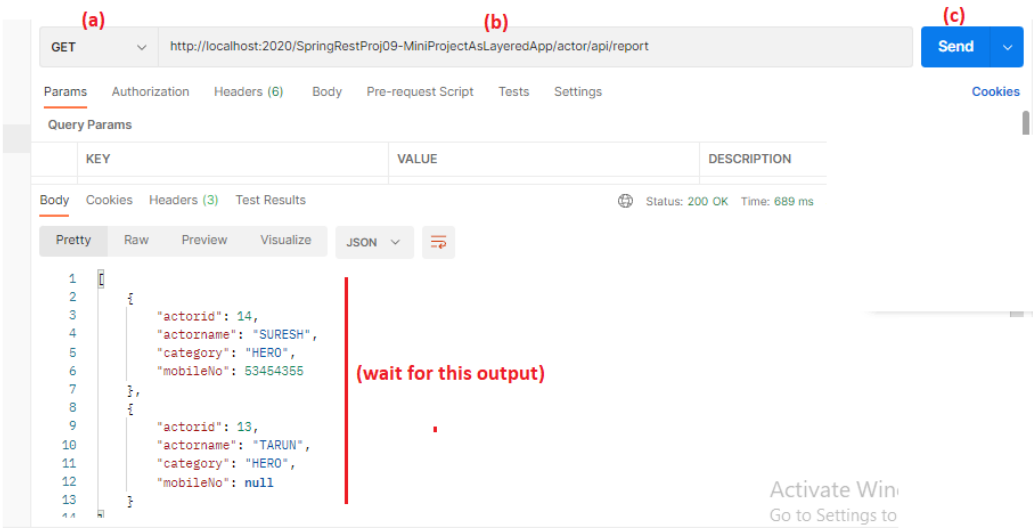
Code in Service interface

```
public  Iterable<Actor> getAllActors();
```

Code in Service Impl

```
@Override
public Iterable<Actor> getAllActors() {
        Iterable<Actor>  it=actorRepo.findAll();
        List<Actor> list1=(List<Actor>)it;
        list1.sort((t1,t2)->t1.getActorname().compareTo(t2.getActorname()));
        return  list1;
}
        (or)
@Override
public Iterable<Actor> getAllActors() {
        Iterable<Actor>  it=actorRepo.findAll();
        Collections.sort((List<Actor>) it, (t1,t2)->t1.getActorname().compareTo(t2.getActorname()));
        return  it;
}
        (or)
@Override
public Iterable<Actor> getAllActors() {
        Iterable<Actor>  it=actorRepo.findAll();
        List<Actor> list1 =
                StreamSupport.stream(it.spliterator(), false).sorted((t1,t2)->t1.getActorname().compareTo(t2.getActorname()))
                 .collect(Collectors.toList());
        return  list1;
}
```

request from POSTMAN



SEELCT Operation getting single record

Code in Service Interface

```
public  Actor  getActorById(int id);
```

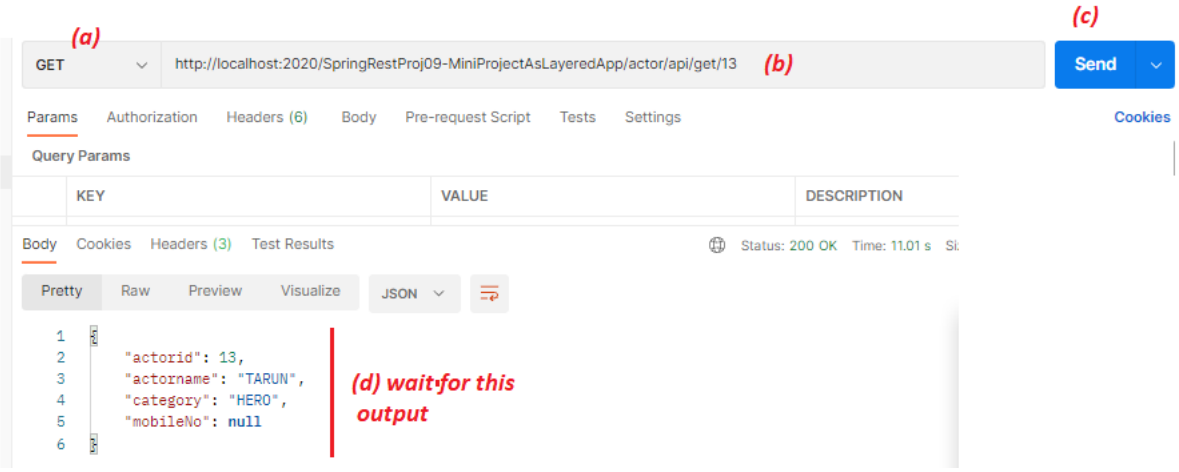
Code in Service Impl class

```
@Override
        public Actor getActorById(int id) {
                //Actor  actor=actorRepo.findById(id).get();
                //return actor;
                return actorRepo.findById(id).orElseThrow(()->new IllegalArgumentException());

        }
```

code in Rest API comp

```
@GetMapping("/get/{id}")
        public  ResponseEntity<?> fetchActorId(@PathVariable  int id){
      //use service
                try {
                        Actor actor=service.getActorById(id);
                        return  new ResponseEntity<Actor>(actor,HttpStatus.OK);
                }
                catch(Exception e) {
                        e.printStackTrace();
                        return  new ResponseEntity<String>
(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
                }
        }
```

Testing from POSTMAN TOOL



Performing Select operation with multiple multiple Conditions

==========

=> if readymade methods of any spring data jpa Repository(I) does not fullfill our requirements then we can add custom logics in the Repository interface either using findBy(-) methods or using custom methods linked with JPQL or Native SQL Queries..

=> getting actors details based on the given categories

In Repository Interface

```
public interface IActorRepo extends CrudRepository<Actor, Integer> {

    @Query("from Actor where category in(:c1,:c2) order by category")  //JPQL query
  public List<Actor> getActorsByCategories(@Param("c1")String category1,@Param("c2")String category2);
}                    custom method
```

In Service Interface

```
public   List<Actor>  fetchActorsByCategory(String category1,String category2);
```

In service Impl class

```
@Override
    public List<Actor> fetchActorsByCategory(String category1, String category2) {
            List<Actor>  list=actorRepo.getActorsByCategories(category1, category2);
            return list;
    }
```

In RestAPI comp

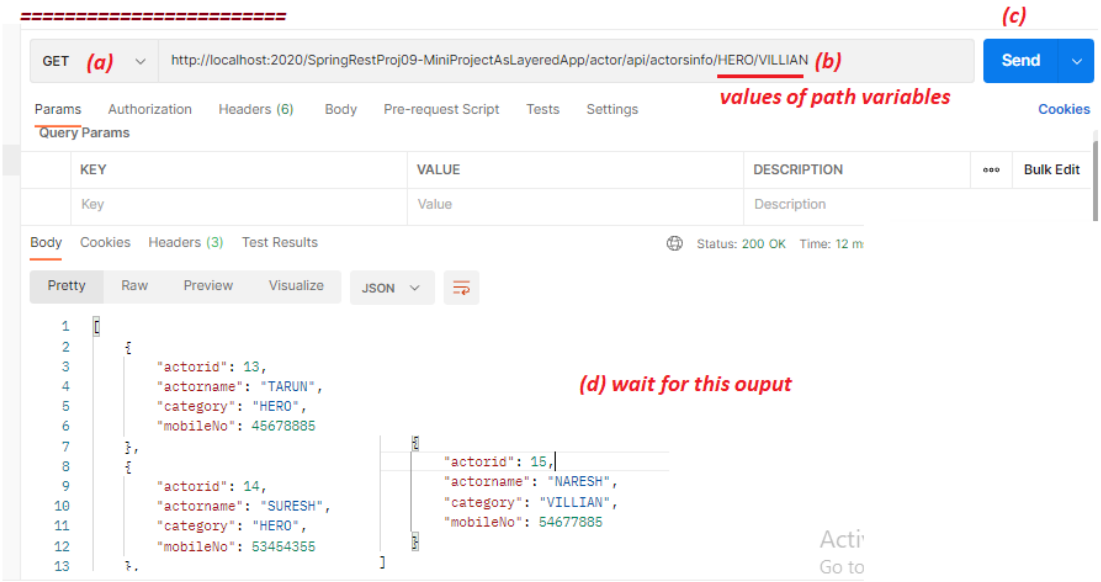
keys of path variables

```
@GetMapping("/actorsinfo/{category1}/{category2}")
	public ResponseEntity<?> showActorsByCategories(@PathVariable String category1,@PathVariable String category2){
		//use service
		try {
			List<Actor> list=service.fetchActorsByCategory(category1, category2);
			return new ResponseEntity<List<Actor>>(list,HttpStatus.OK);
		}
		catch(Exception e) {
			e.printStackTrace();
			return new ResponseEntity<String>(HttpStatus.INTERNAL_SERVER_ERROR);
		}
	}
```

==========

Testing from Postman tool

==========



==========

Update operation using spring REST API (Full record/object updation)

==========

Custom Exception class

```
//ActorNotFoundException.java
package com.nt.exception;

public class ActorNotFoundException extends RuntimeException {

	public ActorNotFoundException() {
		super();
	}

	public ActorNotFoundException(String msg) {
	  super(msg);
	}


}
```

code in serivce Interface

```
public String updateActor(Actor actor);
```

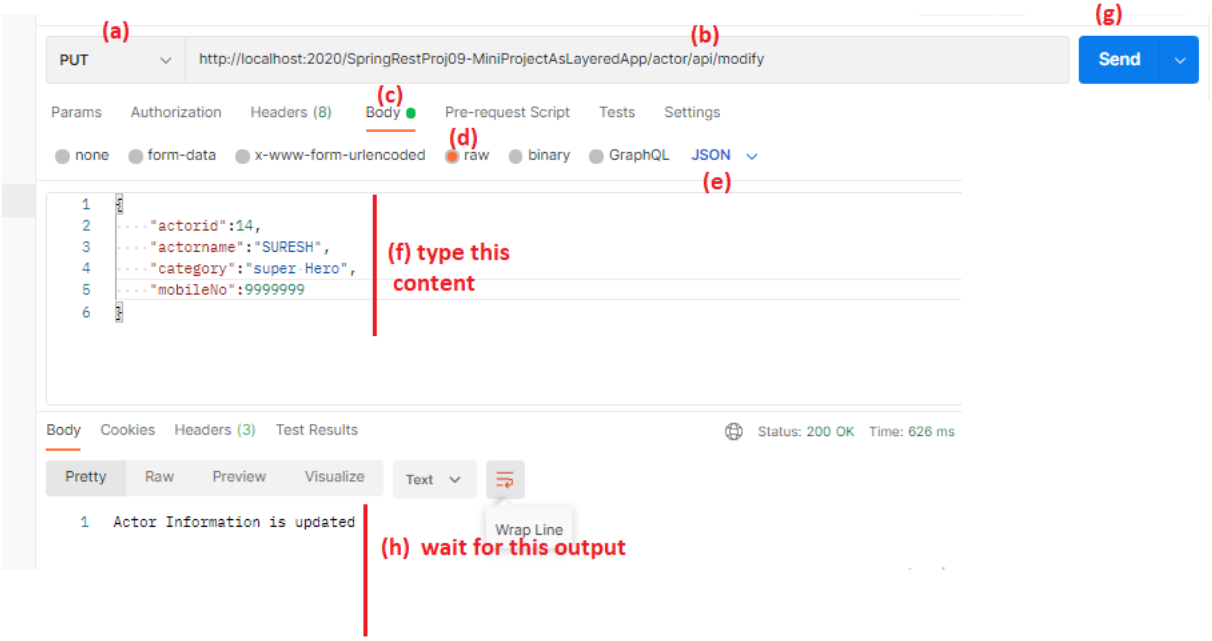
code in Service Impl class

```
@Override
	public String updateActor(Actor actor) {
		Optional<Actor> opt=actorRepo.findById(actor.getActorid());
		if(opt.isPresent()) {
			actorRepo.save(actor);  //update object
			return "Actor Information is updated";
		}
		else {
			throw new  ActorNotFoundException("Actor not found");
		}
	}
```

Code in Rest API comp

```
@PutMapping("/modify")
	public  ResponseEntity<String> modifyActor(@RequestBody Actor actor){

		try {
			//use service
	String msg=service.updateActor(actor);
	return  new ResponseEntity<String>(msg,HttpStatus.OK);
		}
		catch(Exception e) {
			e.printStackTrace();
			return new ResponseEntity<String>
(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
		}
	}//method
```

Testing from POSTMAN tool



Delete operation using spring Rest

==========

Code in Service Interface

```
public  String   deleteActorById(int id);
```

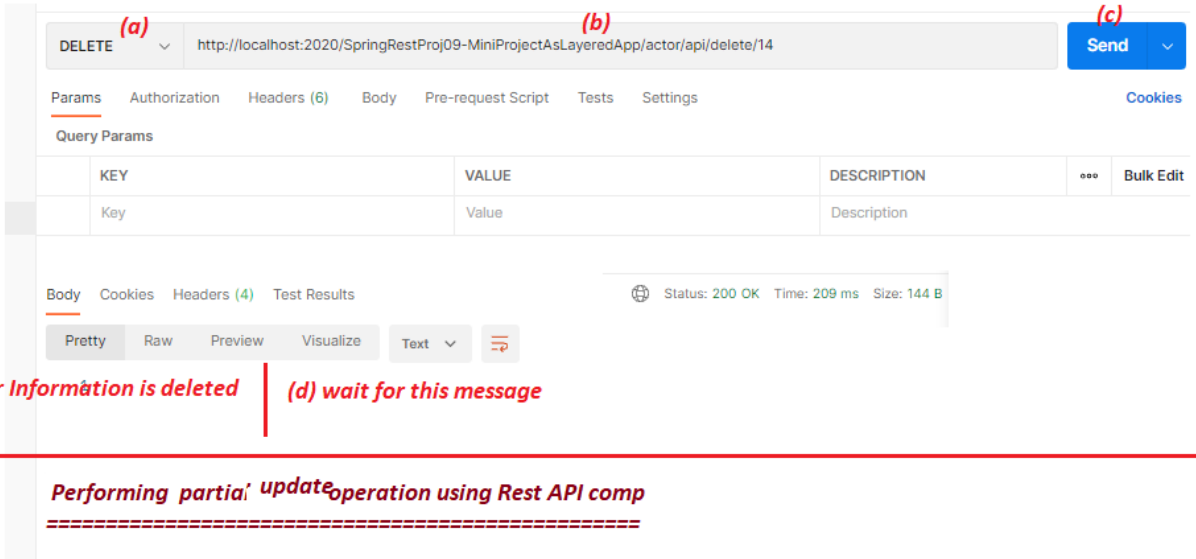
Code in Serivce Impl class

```
@Override
	public String deleteActorById(int id) {
		Optional<Actor> opt=actorRepo.findById(id);
		if(opt.isPresent()) {
			actorRepo.deleteById(id);
			return "Actor Information is deleted";
		}
		else {
			throw new  ActorNotFoundException("Actor not found");
		}
	}
```

Code in RestAPI comp

```
	@DeleteMapping("/delete/{id}")
	public  ResponseEntity<String>  deleteActor(@PathVariable("id") int id){

		try {
			//use service
		String msg=service.deleteActorById(id);
		return  new ResponseEntity<String>(msg,HttpStatus.OK);
		}
		catch(Exception e) {
			e.printStackTrace();
			return new ResponseEntity<String>(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
		}
	}
```

Testing from POSTMAN tool



Actor Information is deleted | *(d) wait for this message*

Performing partial update operation using Rest API comp

For complete record/object updation use PUT mode request
For partital record/object updation use PATCH mode request

Code in service interface

```
public String  updateActorMobileNo(int id , long  newMobileNo);
```

code in service Impl class

```
@Override
	public String updateActorMobileNo(int id, long newMobileNo) {
		Optional<Actor> opt=actorRepo.findById(id);
		if(opt.isPresent()) {
```

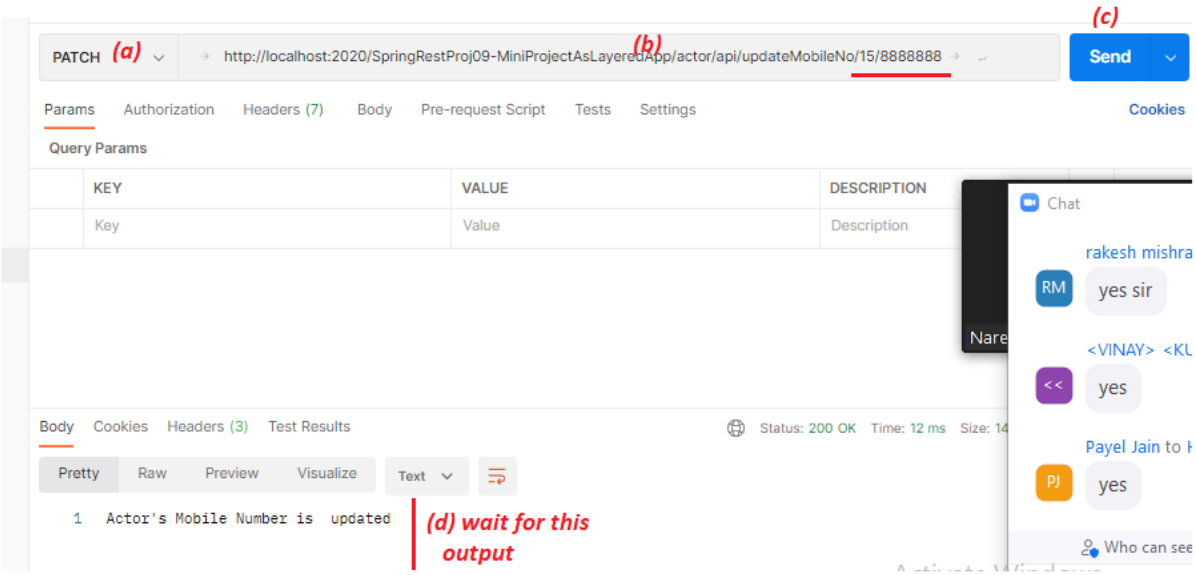
```
                Actor actor=opt.get();
                actor.setMobileNo(newMobileNo);
                actorRepo.save(actor);  //update object
                return "Actor's Mobile Number is  updated";
            }
            else {
                throw new  ActorNotFoundException("Actor not found");
            }

        }
```

Code in REST API comp

```
-------------------------------
    @PatchMapping("/updateMobileNo/{id}/{newNo}")  //partial update
    public  ResponseEntity<String>  changeActorMobileNo(@PathVariable("id") int id,
                                                        @PathVariable("newNo") long newMobileNo){
        try {
            //use service
      String msg=service.updateActorMobileNo(id, newMobileNo);
      return  new ResponseEntity<String>(msg,HttpStatus.OK);
        }
        catch(Exception e) {
            e.printStackTrace();
            return new ResponseEntity<String>(e.getMessage(),HttpStatus.INTERNAL_SERVER_ERROR);
        }

    }//method
```

Testing from POST MAN tool

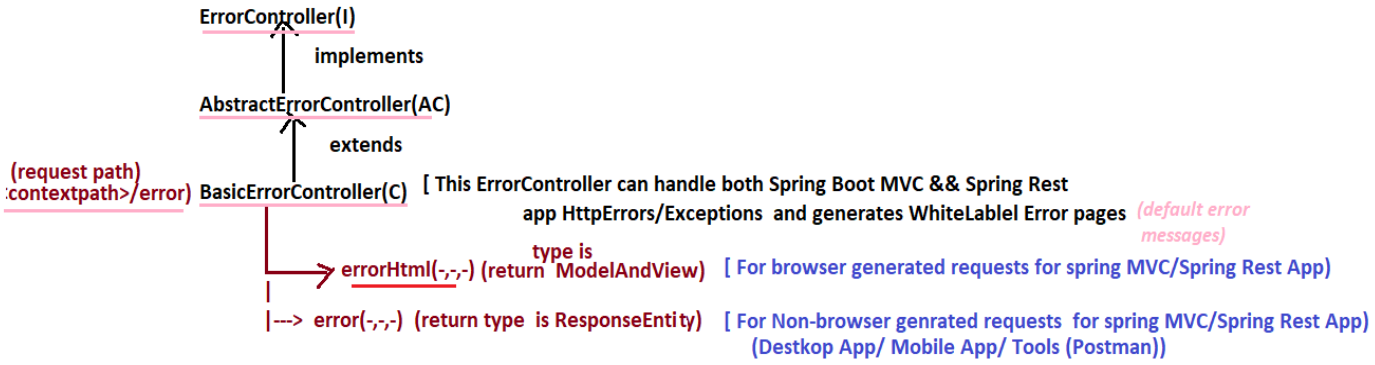
==========



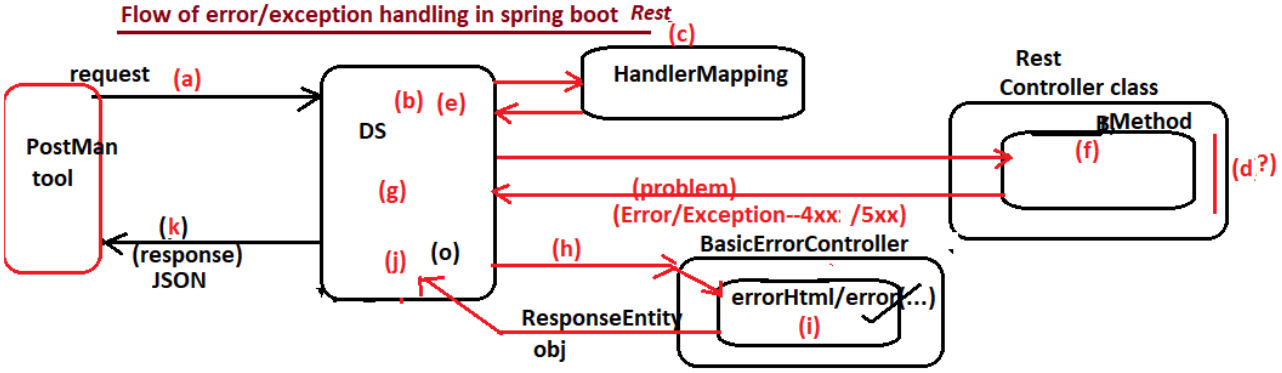
16.NTSPBMS615-June 25th-2022- @ControlAdvice- ExceptionHandling

Exception/Error Handling in SpringRest

Spring Rest/ => Spring Boot MVC is having pre-defined Controller to handle these HttpErrors/exception s and to give default whiteLabel Error pages



Flow of error/exception handling in spring boot *Rest*



=>if we do not catch and handle exception in the b.methods of RestController (API) class then the DS gets the Propagated Exception and gives to error(-,-,-) of pre-defined controller class BasicErrorController which is mapped with <requestpath>/error (like tourist/find/error) .

=>the error(-,-,-) of BasicErrorController class returns ResponseEntity<Map<String,Object>> obj having default error messages to DS and DS converts those messages to JSON Details to send to Client as error Json Response.

To feel this pratically

==========

a) open BasicErrorController class using ctrl+shit+T option

b) get of list of methods using ctrl+ o --> open error(-,-,-) source code

c) keep one breakpoint inside the error(-,-,-) method using ctrl+b or using double click in left margin

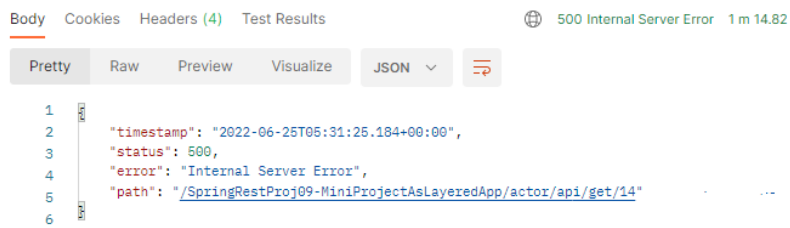
d) Run the App using Debug As server option

e) Given *GET* request from POSTMAN causing exception

not avaiable

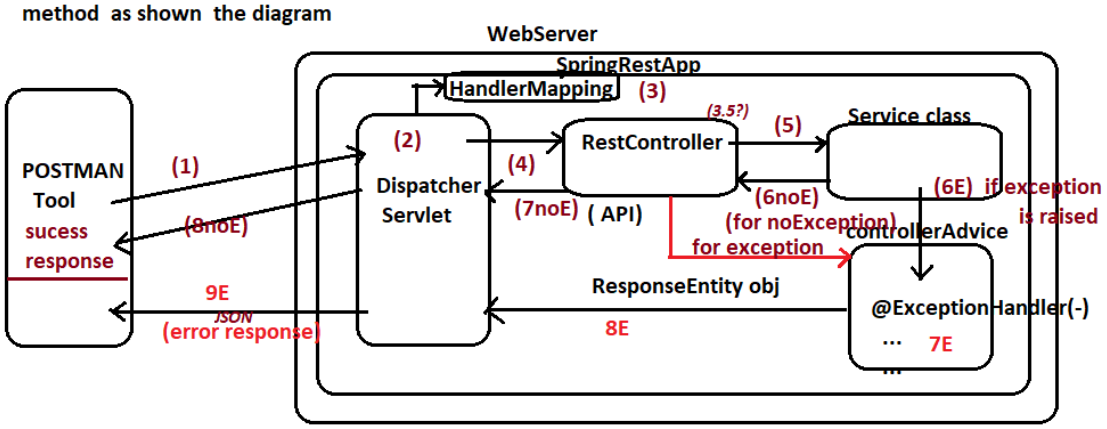
f) press F8 button to move control from default errorHtml(-,-,-) method to error(-,-,-) method then continuosly press f6/f7 button to go further...

g) ResponseEntity<Map<String,Object>> object based JSON Error response in POSTMAN tool as shown below



f5 is step into
f6 step over
f7 step return
f8 next break point

=>Instead of using BasicErrorController to handle the exceptions raised or propagated to RestController methods we can use custom throws advice class that is developed using @ControlAdvice(or) @RestControllerAdvice + @ExceptionHandler which can return ResponseEntity object to DispatcherSErvlet directly by catch the exception from Service class method as shown the diagram



note:: The @ControllerAdvice or @RestControllerAdvice class @ExceptionHandler methods respond to execute for exceptions raised in Repository, service, RestController classes ..

@RestControllerAdvice= @ControllerAdvice + @ResponseBody

Example App
==========

step1) Keep Mini Project ready

step2) Develop ErrorDetails class the model class to hold more details about exception

```
//ErrorDetails.java

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ErrorDetails {
      private LocalDateTime time;
      private String msg;
      private String status;
}
```

Custom property names..

step3) Develop ControllerAdvice class as shown below

```
//@ControllerAdvice
@RestControllerAdvice
public class ActorControllerAdvice {

    @ExceptionHandler(ActorNotFoundException.class)
    public ResponseEntity<ErrorDetails> handleActorNotFoundException(ActorNotFoundException anfe){
          System.out.println("ActorControllerAdvice.handleActorNotFoundException()");
          ErrorDetails details=new ErrorDetails(LocalDateTime.now() ,"500" , anfe.getMessage());
          return new ResponseEntity<ErrorDetails>(details, HttpStatus.INTERNAL_SERVER_ERROR);
    }
```

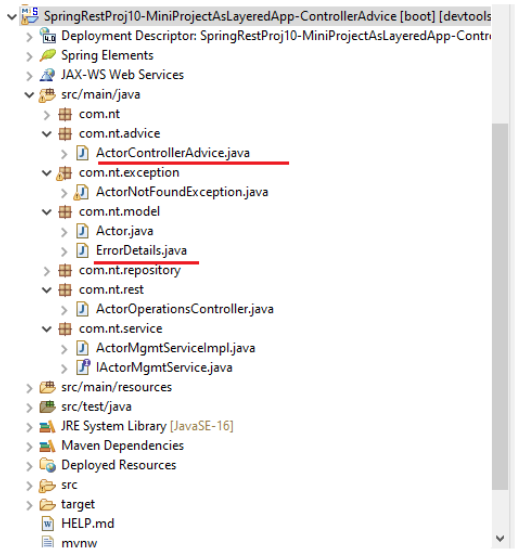
For specific Exception

```
    @ExceptionHandler(Exception.class)
    public ResponseEntity<ErrorDetails> handleAllExceptions(Exception e){
          System.out.println("ActorControllerAdvice.handleAllExceptions()");
          ErrorDetails details=new ErrorDetails(LocalDateTime.now() ,"500" , e.getMessage());
          return new ResponseEntity<ErrorDetails>(details, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

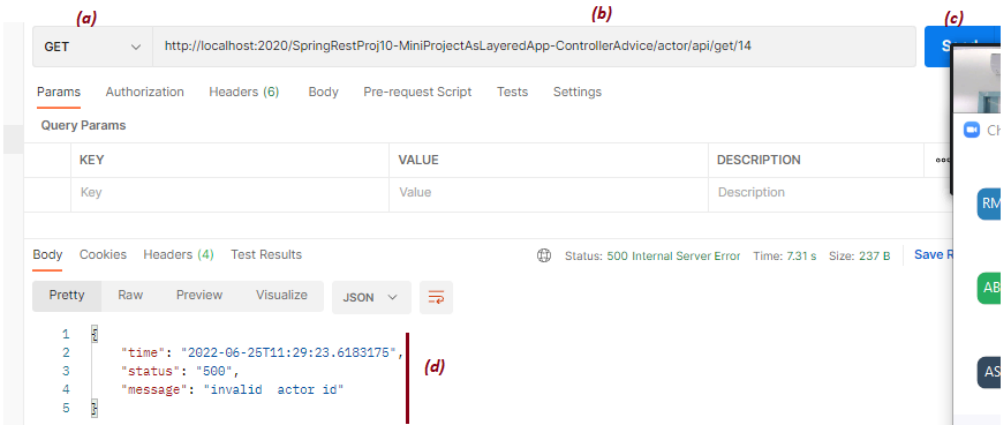
For All Exceptions

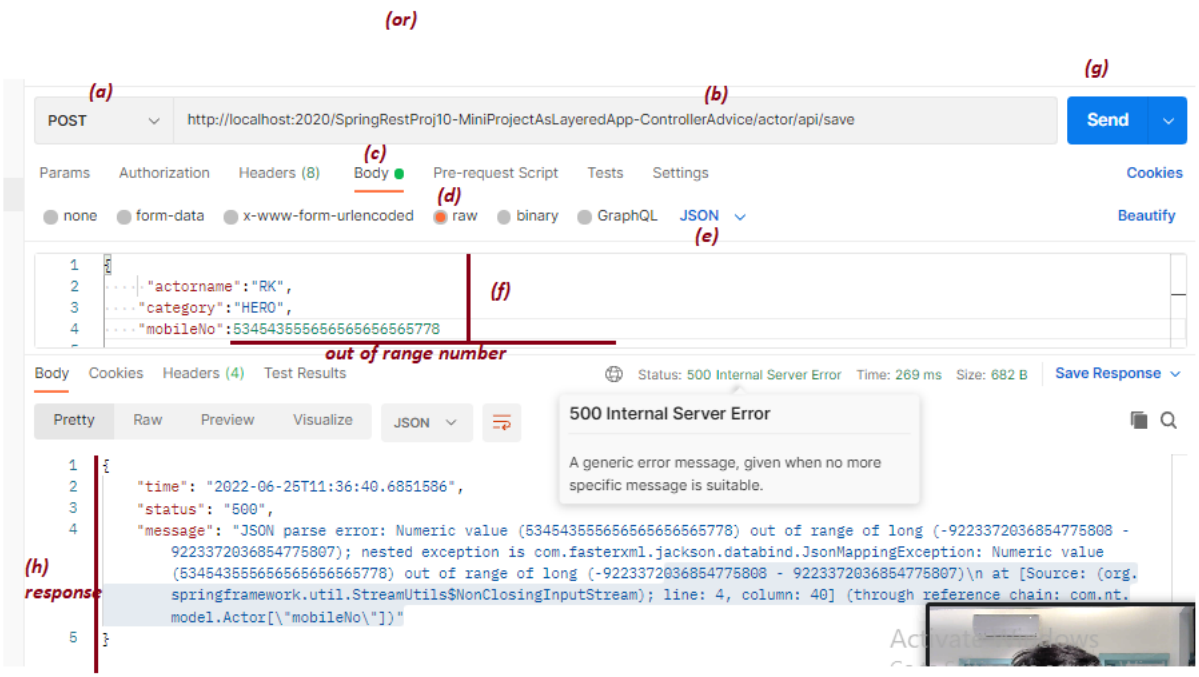
step4) Remove try catch blocks in the methods of RestController class

```java
@RestController
@RequestMapping("/actor/api")
public class ActorOperationsController {
	@Autowired
	private IActorMgmtService service;

	@PostMapping("/save")
	public ResponseEntity<String> saveActor(@RequestBody Actor actor){
		//use service
			String msg=service.registerActor(actor);
			return new ResponseEntity<String>(msg,HttpStatus.CREATED);
	}

	@GetMapping("/report")
	public ResponseEntity<?> fetchAllActors(){
		//use service
		Iterable<Actor> list=service.getAllActors();
		//return ResponseEntity object
		return new ResponseEntity<Iterable<Actor>>(list, HttpStatus.OK);
	}


	@GetMapping("/get/{id}")
	public ResponseEntity<?> fetchActorId(@PathVariable int id){
			Actor actor=service.getActorById(id);
			return new ResponseEntity<Actor>(actor,HttpStatus.OK);
	}


	@GetMapping("/actorsinfo/{category1}/{category2}")
	public ResponseEntity<?> showActorsByCategories(@PathVariable String category1,@PathVariable String category2){
			List<Actor> list=service.fetchActorsByCategory(category1, category2);
			return new ResponseEntity<List<Actor>>(list,HttpStatus.OK);
	}//method


	@PutMapping("/modify")
	public ResponseEntity<String> modifyActor(@RequestBody Actor actor){

	String msg=service.updateActor(actor);
	return new ResponseEntity<String>(msg,HttpStatus.OK);
	}//method


	@DeleteMapping("/delete/{id}")
	public ResponseEntity<String> deleteActor(@PathVariable("id") int id){

	String msg=service.deleteActorById(id);
	return new ResponseEntity<String>(msg,HttpStatus.OK);
	}//method


	@PatchMapping("/updateMobileNo/{id}/{newNo}") //partial update
	public ResponseEntity<String> changeActorMobileNo(@PathVariable("id") int id,
	                                                   @PathVariable("newNo") long newMobileNo){
	String msg=service.updateActorMobileNo(id, newMobileNo);
	return new ResponseEntity<String>(msg,HttpStatus.OK);
	}//method

}//class
```



step4) Run the Application causing exception..





Instead of writing try /catch blocks in each method of RestController class , it is better to use one @ControllerAdvice class with multiple @ExceptionHandler methods to handle all the exceptions raised in different Restcontroller, service, Repository classes..

17.NTSPBMS615- Swagger -Api documentation-june28th-2022

Swagger API

API Creation/Development :: developing RestController having different methods/opertions for various http method types like GET,POST,PATCH,PUT,DELETE and etc... is called API Creation/Development. (@RestController classes development)

=>We can generally take one @RestController per 1 module , So developing each RestController is called api creation../development

Accounts module ---> AccountsController (@RestController) is called Acccounts API creation
Daily Tx module ---> DialyTxController (@RestController) is called DailyTx API creation
and etc..

EndPoints :: Providing muliple details or collection of details that are required to call methods/opertions of @RestController from Client Apps or to send reqeusts from different tools like POSTMAN is called Providing end points..

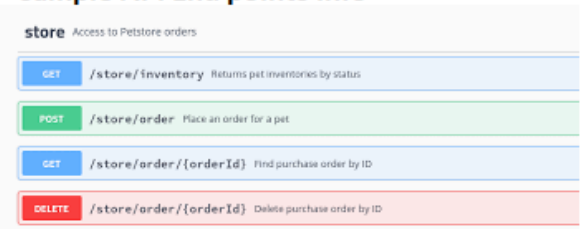
Eg:: For AccountController nothing but Accounts API, the end points are

BaseURL :: http://localhost:3030/RestProj1/tourist
register() :: /register ----> POST
findById(-) :: /find/{id} ----> GET
deleteById(-) : /delete/{id} --->DELETE
and etc..

API End points infomration (Base URL Info + methods info related consumning the webservices)

In End Points of any API we need to provide multiple details like Base URL ,method names, request paths , http method types ,content type and etc..

sample API End points info



API Documentation
==========

=>we can write documentation for java classes in multiple ways
a) using serperate Text docs
b) using API documentation comments and javadoc tool (/** .... */--- doc comments)
note:: Both these approaches non-responsive documentations i.e we can read about java classes and methods but we can not test them immediately

=> After developing Rest API we can provide API documentation for Rest API in the following ways
a) Using Seperate Text docs
b) Using API doc comments (/** ... */) | Creates Non-Responsive API documentation (Bad approach)
c) Using Swagger /Swagger API (Best)
(or) Open API | Creates Responsive API Documentation. (Good approach)

note:: Open API is alternate to Swagger API .. The indurstry standard is still Swagger API

Swagger API
==========

=> It is an open Source Third Party Library to provide Responsive API documentation for RestController and its methods
=> For All RestControllers of the Project we can create API documentation from single place while working Swagger API
=>Responsive Documnentation means not only we get docs about API and its methods (End points) we can test immediately by providing inputs and by getting outputs..
=> if this is used .. there is no need of using POSTMAN tool seperately.. and also very useful to provide Documentation based Testing env.. for Clients (or) consumers

=> Spring Fox +swagger togther released libraries that are required to use swagger api in spring boot Applications

=>Swagger API documentation provides the following details in the GUI Responsive docs
a) API Info (company, title, license url , and etc..)
b) End points Info
c) Model classes info
and etc..

=> While working with swagger documentation for reading and testing we need not to remember and give URL, Http method types , content type and etc.. we just need to give required inputs and get the outputs.

Procedure to work with swagger API (make sure that u r working with spring boot version : 2.5.7
==========
step1) keep RestController /Rest API Project ready.. java version : 1.8 and swagger version: 2.9.2)

step2) Add the following two jar file in pom.xml related to swagger API
a) springfox-swagger2
a) springfox-swagger-ui

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger-ui</artifactId>
   <version>2.9.2</version>
</dependency>
```

for swagger UI presentation

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger2</artifactId>
   <version>2.9.2</version>
</dependency>
```

For basic swagger API

note: change java version to 1.8
chnage spring boot parent version to 2.5.7

step3) Develope seperate Configuration class Enabling Swagger api

=>In configuration class create Docket object having
->Dcoumention type (screent type)
->specify base package of restControllers
->specify reqeusts paths info
->other details of API (ApiInfo obj having company name, licenseurl and etc.)

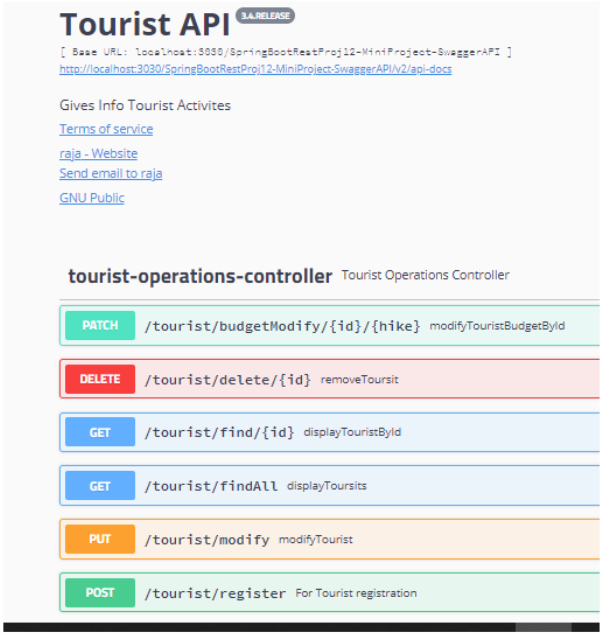
SwaggerDocConfig.java

```
package com.nt.config;

import java.util.Collections;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerDocsConfig {
	@Bean
	 public   Docket createDocket() {
		 return  new Docket(DocumentationType.SWAGGER_2)  //UI screen type
				 .select()  //to specify  RestControllers
				 .apis(RequestHandlerSelectors.basePackage("com.nt.controller")) //base pkg for RestContrllers
				 .paths(PathSelectors.regex("/tourist.*")) // to specify request paths
				 .build() // builds the Docket obj
				 .useDefaultResponseMessages(true)
				 .apiInfo(getApiInfo());
	}

	private  ApiInfo getApiInfo() {
		Contact contact=new Contact("raja","http://www.HCL.com/tourist","natarazjavaarena@gmail.com");
		return  new ApiInfo("Tourist API",
				"Gives Info Tourist Activites",
				"3.4.RELEASE",
				"http://www.hcl.com/license",
				contact,
				"GNU Public",
				"http://apache.org/license/gnu",
	}		Collections.emptyList());
}
```

step4) Run the Server app

step5) use the following url to get swagger api docs and to test the api

http://localhost:3030/SpringBootRestProj12-MiniProject-SwaggerAPI/swagger-ui.html



@ApiOperation("....") can be applied on the rest API/controller methods to provide our choice description and that reflects in swagger docs.

```
@PostMapping("/register")
    @ApiOperation("For Tourist registration")
    public ResponseEntity<String>  enrollTourist(@RequestBody Tourist tourist){
          try {
          //use service
          String resultMsg=service.registerTourist(tourist);
           return  new ResponseEntity<String>(resultMsg,
                                                    HttpStatus.CREATED); //201 content created successfully
          }
          catch(Exception e) {
                e.printStackTrace();
                return  new ResponseEntity<String>("problem in tourist enrollment",
           HttpStatus.INTERNAL_SERVER_ERROR); //500 error
          }

    }//method
```

18.NTSPBMS615- Rest Consumer using RestTemplate-June 30th-2022

Developing Consumer App using RestTemplate

========== of

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

=> 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

```
RestTemplate template=new RestTemplate();
            (or)
In @Configuration class
-----------------------------------
  @Bean("template")
 public   RestTemplate createTemplate(){
       return new   RestTemplate();
  }
```

(good)

=>This RestTemplate can injected to multiple other spring beans or runner classes.

Why RestTemplate is not coming through AutoConfiguration?

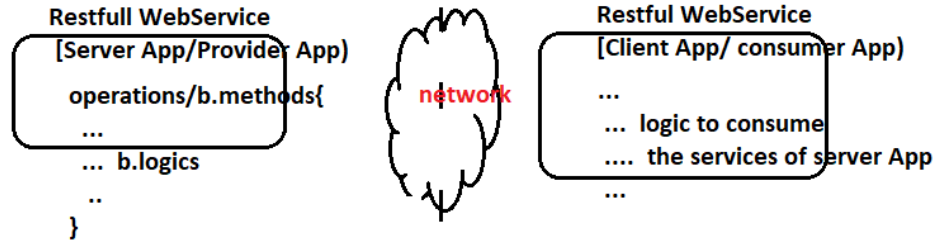
=>The cosumer App is another business app and there is not quarantee this business App should always cosume Rest API comp.. So RestTemplate is not coming through AutoConfiguration..

=> 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 /publisher app (@RestController class methods)

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

=> It provides xxxForEntity(....) methods like getForEntity(...) , postForEntity(...) and etc.. taking url, request obj(body,header) to send different modes http 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 in running same server or different servers belonging to same machine or different machines.



Server/Provider App	Client /Cosumer App
PaymentGateway	PayTM App
PayTM/GooglePay/PhonePay	Flipkart/Amazon
WeatherReport App	yahoo.com/Tourist.com
ICC Score Comp	CrickInfo.com, CrickBuzz.com and etc..

=>The RestFullWebService (Server/ provider App) must be the web application
=> The consumer App can be the standalone App or mobile App or IOT App or Web application or etc..

So far we have devleoped 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..

=>RestTemplate can be used only in Spring or Spring Boot env.. (The cosumer Apps for Rest webService comp/API can be developed outside the spring /spring boot env.. using jax-rs,resteasy and etc there RestTemplate can not be used)

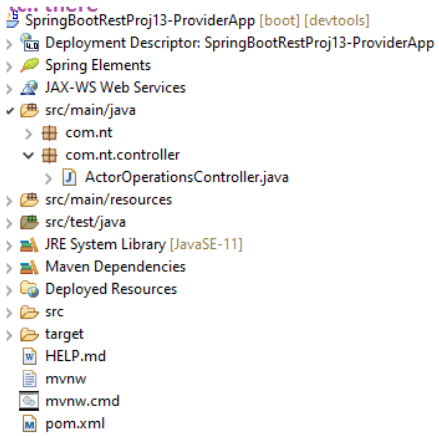
Example App
==========

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;
```



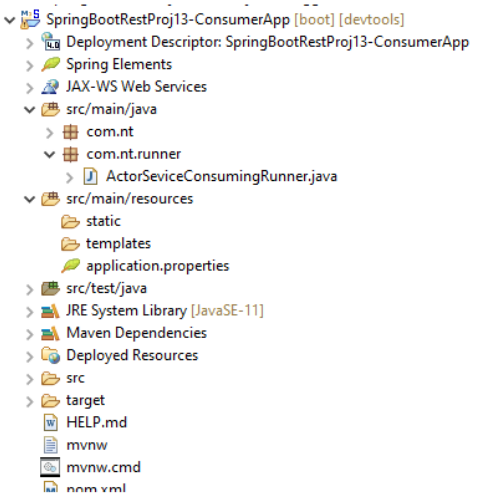

```
//WishMessageController.java
package com.nt.rest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/wish/api")
public class WishMessageController {

        @GetMapping("/message")
        public ResponseEntity<String>  showWishMessage(){
                return new ResponseEntity<String>("Good Night",HttpStatus.OK);
        }

}
```

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

---

step4) Develop the Consumer App as seperte Project
(starters :: web , dev tools , lombok)

step5) place the following entries in application. properties
server.port=4040



step6) Develop the Runner App

```
package com.nt.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class WishMessageConsumerRunner implements CommandLineRunner {

    @Override
    public void run(String... args) throws Exception {
        //create RestTemplate class obj
        RestTemplate template=new RestTemplate();
        //prepare  service url (or) base url + requestpath
        String  serviceUrl="http://localhost:2020/SpringRestProj12-ProviderApp/wish/api/message";
        //consume the service using xxxForEntity(-) method
        ResponseEntity<String> response=template.getForEntity(serviceUrl,String.class);
        // process response
        System.out.println(" response body/payload (output)::"+response.getBody());
        System.out.println("response status code Value::"+response.getStatusCodeValue());
        System.out.println("response Codecode ::"+response.getStatusCode());
        System.out.println("response headers::"+response.getHeaders());
    }
}
```

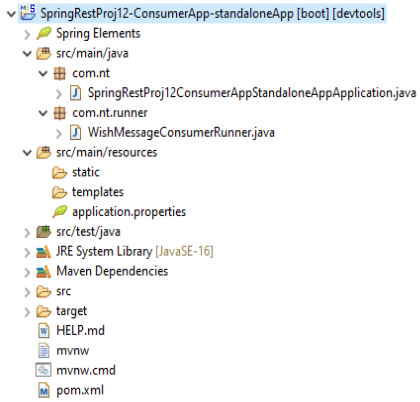
output::
response body/payload (output)::Good Night
response status code Value::200
response Codecode ::200 OK
response headers::[Content-Type:"text/plain;charset=UTF-8", Content-Length:"10", Date:"Thu, 30 Jun 2022 14:59:41 GMT"]

step4) Run Consumer App as spring boot App that uses Embedeed Tomcat server..

Run As ----> spring Boot App/ jav aApp
(Uses Embedded SErver)

Runners in spring boot app executes only when we run the app as the spring boot app or java app

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



Runner class code and application.propeeties content is same as the above Consumer App..

[Uses the Embeded Tomcat server]

application.properties

```
#Embedded tomcat port number
server.port=4042

#service url
service.url=http://localhost:2020/SpringRestProj12-ProviderApp/wish/api/message
```

```
//WishMessageConsumerRunner.java
package com.nt.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.CommandLineRunner;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class WishMessageConsumerRunner implements CommandLineRunner {
	@Autowired
	private  RestTemplate  template;
	@Value("${service.url}")
	private String serviceUrl;


	@Override
	public void run(String... args) throws Exception {
		//prepare  service url (or) base url + requestpath
		String  serviceUrl="http://localhost:2020/SpringRestProj12-ProviderApp/wish/api/message";
		//consume the service using xxxForEntity(-) method
		ResponseEntity<String> response=template.getForEntity(serviceUrl,String.class);
		// process response
		System.out.println(" response body/payload (output)::"+response.getBody());
		System.out.println("response status code Value::"+response.getStatusCodeValue());
		System.out.println("response Codecode ::"+response.getStatusCode());
		System.out.println("response headers::"+response.getHeaders());


	}
}
```

Run this app as the spring boot App/Java App

output:

```
======
 response body/payload (output)::Good Night
response status code Value::200
response Codecode ::200 OK
response headers::[Content-Type:"text/plain;charset=UTF-8", Content-Length:"10", Date:"Thu, 30 Jun 2022 15:09:21 GMT"]
```

main class and configuration class

```
package com.nt;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class SpringRestProj12ConsumerAppStandaloneAppApplication {

	@Bean(name="template")
	public RestTemplate  createTemplate() {
		return new RestTemplate();
	}

	public static void main(String[] args) {
		SpringApplication.run(SpringRestProj12ConsumerAppStandaloneAppApplication.class, args);
	}

}
```

=> for one provider app any no.of cosumer apps can be there at a time consuming the same or different services of the provider App

19.NTSPBMS615- Rest Consumer using RestTemplate-July 1st-2022

What is the difference b/w getforEntity(..) and getForObject(..) methods?

Ans) getForEntity(..) return type is ResponseEntity<T> which contains response body(output), headers, status code and etc..

getForObject(..) return type is Object which gives only response body(output) i.e it does not give response headers, status code and etc..

=>xxxForEntity(...) are better methods to use over xxxForObject(...) methods

note:: when we develop spring web mvc/spring rest app as war file(web application) we can use both extenal server and Embedded Server (like Embedded Tomcat) for deployment.

note:: when we develop spring web mvc/spring rest app as jar file(standalone web application) we can use only Embedded server (like Embedded Tomcat) for deployment.

sample code

```
//consume the service using xxxForEntity(-) method
        ResponseEntity<String> response=template.getForEntity(serviceUrl,String.class);
        // process response
        System.out.println(" response body/payload (output)::"+response.getBody());
        System.out.println("response status code Value::"+response.getStatusCodeValue());
        System.out.println("response Codecode ::"+response.getStatusCode());
        System.out.println("response headers::"+response.getHeaders());

        System.out.println("============================");
        String response1=template.getForObject(serviceUrl, String.class);
        System.out.println("response content is ::"+response1); //gives only response body
```

Passing Path variable values along with http request calls using RestTemplate

=>Both getForObject(...), getForEntity(...) are having multiple overloaded forms .. the form that is taking more params is maintaining last param as var args param which is given to pass any no.of path variable values along with http request call.

<T> T	getForObject(String url, Class<T> responseType, Map<String,?> uriVariables) Retrieve a representation by doing a GET on the URI template.	To pass path variable names and vlaues as map collection	Need not to fallow the order while passing values
<T> T	getForObject(String url, Class<T> responseType, Object... uriVariables) Retrieve a representation by doing a GET on the specified URL.	To pass path variable vlaues as var args..	Needs to fallow the order while passing values
<T> ResponseEntity<T>	getForEntity(String url, Class<T> responseType, Map<String,?> uriVariables) Retrieve a representation by doing a GET on the URI template.	To pass path variable names and vlaues as map collection	Need not to fallow the order while passing values
<T> ResponseEntity<T>	getForEntity(String url, Class<T> responseType, Object... uriVariables) Retrieve an entity by doing a GET on the specified URL.	To pass path variable vlaues as var args..	Needs to fallow the order while passing values

Example App

In server /producer / Provider App

RestControlller

```
@RestController
@RequestMapping("/wish/api")
public class WishMessageController {

    @GetMapping("/message/{id}/{name}")
    public ResponseEntity<String> showWishMessage(@PathVariable("id") int id,
                                                  @PathVariable("name") String name){
        System.out.println(id + "------"+name);
        return new ResponseEntity<String>("Good Night::"+id+"--- "+name,HttpStatus.OK);
    }
}
```

In Consumer App/Client App /

==========

runner class

```
@Component
public class WishMessageConsumerRunner implements CommandLineRunner {

    @Override
    public void run(String... args) throws Exception {
        //create RestTemplate class obj
        RestTemplate template=new RestTemplate();
        //prepare service url (or) base url + requestpath

          String serviceUrl="http://localhost:2020/SpringRestProj13-ProviderApp-PathVariables-JSONData/wish/api/message/{id}/{name}";
          //consume the service using xxxForEntity(-) method
          ResponseEntity<String> response=template.getForEntity(serviceUrl,String.class,
                                                  Map.of("name","raja","id",1001)); //Map obj to pass path variable values
          // process response                                   Here we need not follow the order
          System.out.println(" response body/payload (output)::"+response.getBody());
          System.out.println("response status code Value::"+response.getStatusCodeValue());
          System.out.println("response Codecode ::"+response.getStatusCode());
          System.out.println("response headers::"+response.getHeaders());
          System.out.println("============================");
          String response1=template.getForObject(serviceUrl, String.class,114,"rajesh");  //var args to pass path variable vlaues
          System.out.println("response content is ::"+response1); //gives only response body
                                                                  we need to follow the order here

                    output:
    }               response body/payload (output)::Good Night::1001--- raja
                    response status code Value::200
}                   response Codecode ::200 OK
                    response headers::[Content-Type:"text/plain;charset=UTF-8", Content-Length:"24", Date:"Fri, 01 Jul 2022 15:01:33 GMT"]
                    ============================
                    response content is ::Good Night::114--- rajesh
```

Sending JSON Data from Consumer app along with POST mode request using RestTemplate

=> POST mode request contains request body, request headers and initial line
where as GET mode request contains only request headers and initial line.

Every request sturcture contains 2 parts
- HEAD part (initial line + request headers)
- BODY/Payload part (request body)

HEAD, BODY of request structure will be seperate with blankline

=> Here we need to use postForEntity(..) or postForObject(...) methods of RestTemplate to get
to send post mode request and response completelty or partially..

`<T> ResponseEntity<T>`	`postForEntity(String url, Object request, Class<T> responseType, Map<String,?> uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the response as `HttpEntity`.
`<T> ResponseEntity<T>`	`postForEntity(String url, Object request, Class<T> responseType, Object... uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`postForEntity(URI url, Object request, Class<T> responseType)` Create a new resource by POSTing the given object to the URL, and returns the response as `ResponseEntity`.

Path variables

HttpEntity obj having head ,body of the request

`<T> T`	`postForObject(String url, Object request, Class<T> responseType, Map<String,?> uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the representation found in the response.
`<T> T`	`postForObject(String url, Object request, Class<T> responseType, Object... uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the representation found in the response.
`<T> T`	`postForObject(URI url, Object request, Class<T> responseType)` Create a new resource by POSTing the given object to the URL, and returns the representation found in the response.

path variables

In server/ Producer/Provider App

==========

RestController

==========

```
@RestController
@RequestMapping("/wish/api")
public class WishMessageController {


    @PostMapping("/register")
    public  ResponseEntity<String> registerTourist(@RequestBody Tourist tourist){
        System.out.println("WishMessageController.registerTourist() ::"+ tourist);
        return new ResponseEntity<String>("tourist Info::"+tourist.toString(),HttpStatus.CREATED);
    }

}
```

Model class

```
//Tourist.java
package com.nt.model;

import lombok.Data;

@Data
public class Tourist {
      private int tid;
      private String tname;
      private String startPlace;
      private String  destPlace;

}
```

In Consumer App/Client App

==========

Runner class

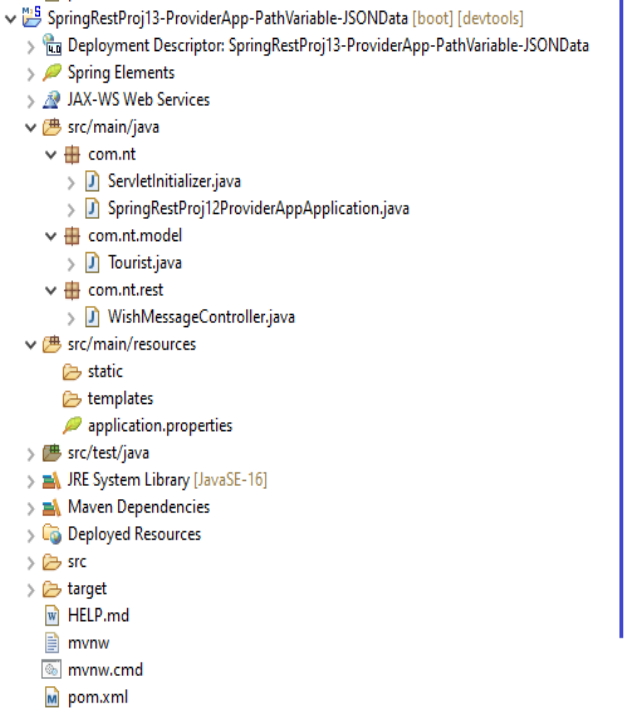
```
@Component
public class TouristInfoConsumer_Posting_JsonData implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		//create RestTemplate  object
		RestTemplate template=new RestTemplate();
		//prepare serivce url
		//prepare  service url (or) base url + requestpath
		String  serviceUrl="http://localhost:2020/SpringRestProj13-ProviderApp-PathVariables-JSONData/wish/api/register";
		//prepare  HttpHeaders
		  HttpHeaders headers=new HttpHeaders();
		  headers.setContentType(MediaType.APPLICATION_JSON); //optional to specify while sending JSON data
		//prepare HttpRequest as HttpEntity having headers,body
		  String json_body="{ \"tid\":1001,\"tname\":\"raja\",\"startPlace\":\"hyd\",\"destPlace\":\"goa\" }";
		  HttpEntity<String> request=new HttpEntity<String>(json_body, headers);
	// send  POST mode request to  consume rest api service
		ResponseEntity<String> response=template.postForEntity(serviceUrl, request, String.class);
		//process recived response
		System.out.println("response body::"+response.getBody());
		System.out.println("response status code code ::"+response.getStatusCodeValue());
		System.out.println("response status code  ::"+response.getStatusCode());
		System.out.println("response headers"+response.getHeaders());
		System.out.println("_____________________________________");

		 // send  POST mode request to  consume rest api service
		    String response1=template.postForObject(serviceUrl, request, String.class);
		    System.out.println("response content/body::"+response1);

	}
}
```

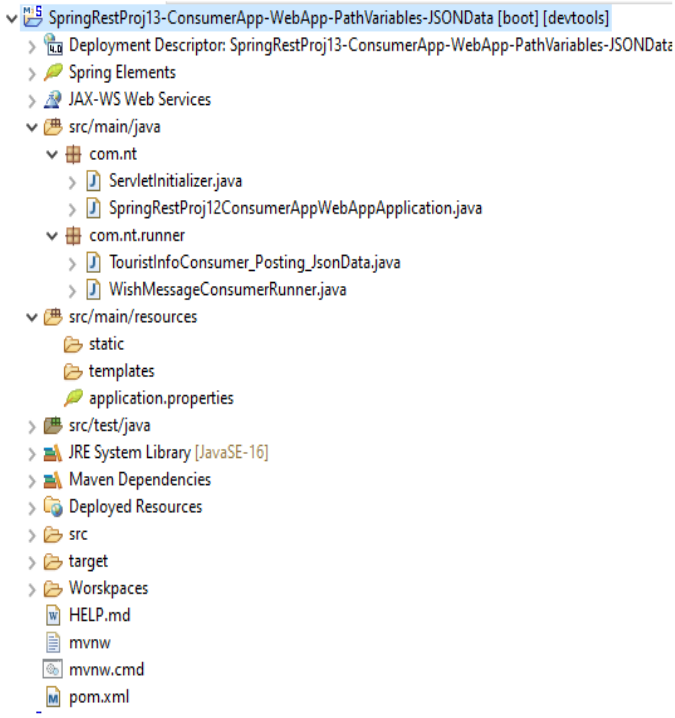
HttpHeaders obj contains request header info

HttpEnttity object contains request body and headers (in any form) (HttpHeaders obj)

represents one http request having body and headers

Producer App





20.NTSPBMS615- Rest Consumer using RestTemplate-July 2nd-2022-exchange(-) method

=> RestTemplate does not provide deleteForEntity(..) /deleteForObject(...) methods and putForEntity(..) and putForObject(..) methods .. For that we need to take delete(..) , put(..) methods of RestTemplate class whose return type is void.

`delete(String url, Map<String,?> uriVariables)` Delete the resources at the specified URI.	`void` `put(String url, Object request, Map<String,?> uriVariables)` Creates a new resource by PUTting the given object to URI template.
`delete(String url, Object... uriVariables)` Delete the resources at the specified URI.	`void` `put(String url, Object request, Object... uriVariables)` Create or update a resource by PUTting the given object to the URI.
`delete(URI url)` Delete the resources at the specified URL.	`void` `put(URI url, Object request)` Creates a new resource by PUTting the given object to URL.

Example delete mode request

In service provider (@RestController)

```
@DeleteMapping("/delete/{id}")
public ResponseEntity<String> deleteTourist(@PathVariable("id") int tid){
 return new ResponseEntity<String>(tid+" --Tourist is deleted",HttpStatus.OK);
}
```

In Service Consumer

```
@Component
public class DeleteTouristRunner implements CommandLineRunner {

        @Override
        public void run(String... args) throws Exception {
    //create  RestTemplate class obj
                RestTemplate template=new RestTemplate();
                        //prepare  service url (or) base url + requestpath
                        String  serviceUrl="http://localhost:2020/SpringRestProj13-ProviderApp-PathVariables-JSONData/wish/api/delete/{id}";
                        //send delete mode request
                        template.delete(serviceUrl, Map.of("id",1001));  //delete(-) method return type is void
                        System.out.println("delete operation is completed");
    }
```

Working with RestTemplate ***exchange(-)***

=>Instead of calling getForXxx() ,postForXxx() , put(..),delete() and etc.. as seperate methods to generate different modes of requests .. we can use single exchange(....) for all operations. (one method to generate all modes of requets)

exchange(....) method is all rounder method --It is one for sending all modes of requests

```
public <T> ResponseEntity<T> exchange(String url,
                                      HttpMethod method,
                                      @Nullable
                                      HttpEntity<?> requestEntity,
                                      Class<T> responseType,
                                      Object... uriVariables)
                               throws RestClientException
```

Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.

URI Template variables are expanded using the given URI variables, if any.

Specified by:

`exchange in interface RestOperations`

Parameters:

`url - the URL`

`method - the HTTP method (GET, POST, etc)`

`requestEntity - the entity (headers and/or body) to write to the request may be null` **(headers+ body)**

`responseType - the type to convert the response to, or Void.class for no body` **(required response type)**

`uriVariables - the variables to expand in the template` **(path variables values)**

Returns:

`the response as entity`

Throws:

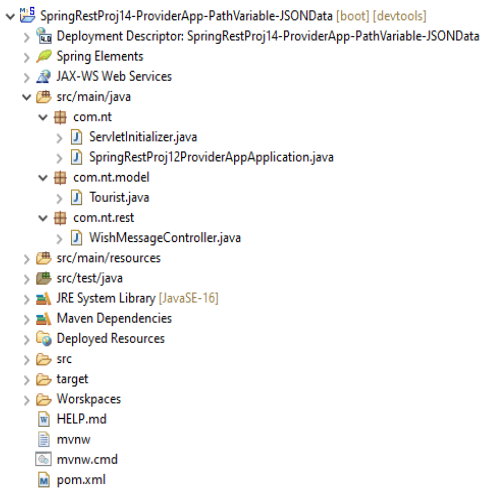
`RestClientException`

exchnage(.....) is alternate for the following methods

getForEntity(...) , postForEntity(...) , getForObject(...) , postForObject(...), delete(...) ,put(...) , patchForEntity(...) , patchForObject(...) and etc..

Example app

Provider/Producer/Server App

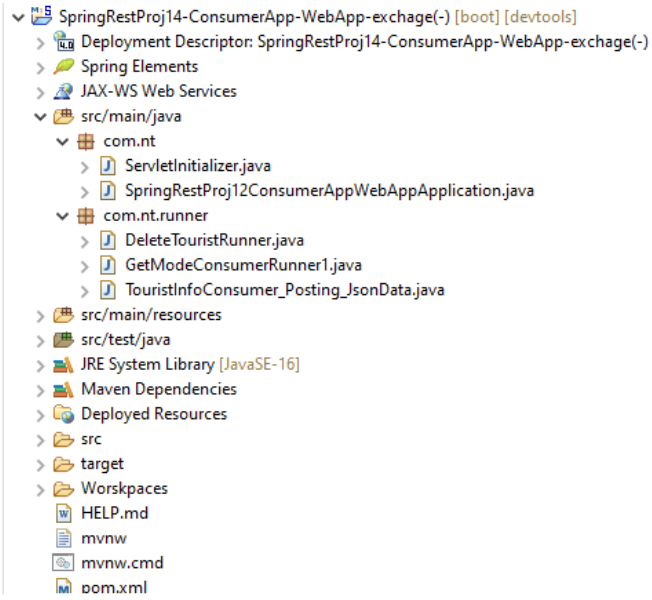



```
@RestController
@RequestMapping("/wish/api")
public class WishMessageController {

	@GetMapping("/report")
	public  ResponseEntity<String> showWishMessage(){
		return new ResponseEntity<String>("Good Night::",HttpStatus.OK);
	}

	@GetMapping("/message/{id}/{name}")
	public ResponseEntity<String>  showWishMessage(@PathVariable("id") int id,
	                                               @PathVariable("name") String name){
		System.out.println(id + "------"+name);
		return new ResponseEntity<String>("Good Night::"+id+"--- "+name,HttpStatus.OK);
	}

	@PostMapping("/register")
	public  ResponseEntity<String>  registerTourist(@RequestBody Tourist  tourist){
		System.out.println("WishMessageController.registerTourist() ::"+ tourist);
		return new ResponseEntity<String>("tourist Info::"+tourist.toString(),HttpStatus.CREATED);
	}

	@DeleteMapping("/delete/{id}")
	public  ResponseEntity<String>  deleteTourist(@PathVariable("id") int tid){
		return new ResponseEntity<String>(tid+" --Tourist is deleted",HttpStatus.OK);
	}
```

Client App/ consumer App

```
@Component
public class GetModeConsumerRunner1 implements CommandLineRunner {


	@Override
	public void run(String... args) throws Exception {
		//create RestTemplate class obj
		RestTemplate template=new RestTemplate();
		//prepare  service url (or) base url + requestpath
		String  serviceUrl="http://localhost:2020/SpringRestProj14-ProviderApp-PathVariable-JSONData/wish/api/report";
		//invoke the exchnage(...) method
		ResponseEntity<String> response1=template.exchange(serviceUrl,
		                                                   HttpMethod.GET,
		                                                   null, //no body+ headers
		                                                   String.class);
		System.out.println("response body content ::"+response1.getBody());
		System.out.println("response header ::"+response1.getHeaders());
		System.out.println("response status code::"+response1.getStatusCode()+" status code value:"+response1.getStatusCodeValue());
		System.out.println("-------------------------------------------------------------------------------");

		//prepare  service url (or) base url + requestpath
		 serviceUrl="http://localhost:2020/SpringRestProj14-ProviderApp-PathVariable-JSONData/wish/api/message/{id}/{name}";
		//invoke the exchnage(...) method
		ResponseEntity<String> response2=template.exchange(serviceUrl,
		                                                   HttpMethod.GET,
		                                                   null,  //no body+ headers
		                                                   String.class,
		                                                   Map.of("id",1001,"name","rajesh"));
		System.out.println("response body content ::"+response2.getBody());
		System.out.println("response header ::"+response2.getHeaders());
		System.out.println("response status code::"+response2.getStatusCode()+" status code value:"+response2.getStatusCodeValue());
	}
```

```
//TouristInfoConsumer_Posting_JsonData.java
package com.nt.runner;


import org.springframework.boot.CommandLineRunner;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class TouristInfoConsumer_Posting_JsonData implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
		//create RestTemplate  object
		RestTemplate template=new RestTemplate();
		//prepare serivce url
		//prepare  service url (or) base url + requestpath
		String  serviceUrl="http://localhost:2020/SpringRestProj14-ProviderApp-PathVariable-JSONData/wish/api/register";
		//prepare  HttpHeaders
		  HttpHeaders headers=new HttpHeaders();
		  headers.setContentType(MediaType.APPLICATION_JSON); //optional to specify while sending JSON data
		//prepare HttpRequest as HttpEntity having headers,body
		   String json_body="{ \"tid\":1001,\"tname\":\"raja\",\"startPlace\":\"hyd\",\"destPlace\":\"goa\" }";
		   HttpEntity<String> request=new HttpEntity<String>(json_body, headers);
 // send  POST mode request to  consume rest api service using  exchange(...)
		   ResponseEntity<String> response=template.exchange(serviceUrl,HttpMethod.POST,
		                                                     request, String.class);
		   System.out.println("_______________POST mode request----------");
		   //process recived response
		   System.out.println("response body::"+response.getBody());
		   System.out.println("response status code code ::"+response.getStatusCodeValue());
		   System.out.println("response status code  ::"+response.getStatusCode());
		   System.out.println("response headers"+response.getHeaders());
	    }  System.out.println("_____________________________________________");
 }

@Component
public class DeleteTouristRunner implements CommandLineRunner {

	@Override
	public void run(String... args) throws Exception {
	//create  RestTemplate class obj
		RestTemplate template=new RestTemplate();
		    //prepare  service url (or) base url + requestpath
		    String  serviceUrl="http://localhost:2020/SpringRestProj14-ProviderApp-PathVariable-JSONData/wish/api/delete/{id}";
		    //send delete mode request
		    ResponseEntity<String> response=template.exchange(serviceUrl,
		                                                      HttpMethod.DELETE,
		                                                      null,
		                                                      String.class,
		                                                      101);
		     System.out.println("_______________Delete mode request----------");
		  //process recived response
		  System.out.println("response body::"+response.getBody());
		  System.out.println("response status code code ::"+response.getStatusCodeValue());
		  System.out.println("response status code  ::"+response.getStatusCode());
		  System.out.println("response headers"+response.getHeaders());
		  System.out.println("_____________________________________________");
	}

}	=>put(-) ,delete(-) methods return type is  void .. So cosumer apps can not get any messages from
	proivider apps while working with put(-),delete(-) methods

	=>To solve the above problem use exchange(...) whose  return type ResponseEntity<T> i.e it can get
	provider generated results and  can  also be used to generate different modes of requests..
```



note: Instead of creating Multiple objects for RestTemplate class in different runner classes of Cosumer App .. create RestTemplate class obj in main class using @Bean method and inject the spring bean obj in runner classes with the support @Autowired annotation.

Recieving and processing diferent return type values of webService methods/operations using RestTemplate from

==========

=> if web service method /operation return type is ResponseEntity<Stirng> /String then we get plain text as the response..

if web service method /operation return type is othe than ResponseEntity<Stirng> /String then we get JSON response as text content.. by default .. this JSON text content can be used directly or can be converted to Java object using JACKSON API (built -in spring boot Rest Applications)

=>Once we add spring MVC starter to the Project ... we get JACSON API automatically it can be used to convert JSON data to Object (DeSerilization) and Object to JSON data (Serialization) using the support of ObjectMapper

=> Irrespective of the return type of web service method/operation (String or non-Stirng) we generally take the return value of exchnage(...) or getForXxx(...) or postForXxx(...) and etc.. methods as String based return value like ResponseEntity<String> object.. i.e recieve every thing from provider App as String content and convert it to other types if needed.

=>The ObjectMapper class of Jackson api gives methods for Serilziation and DeSerilizatation

a) mapper.readValue(-) :: JSON text to Object (DeSerialization)
b) mapper.writeValue(-) :: Object to JSON text (Serialization)

Sever App/proividerApp/Producer App
==========

```
@RestController
@RequestMapping("/politician/api")
public class PoliticianOperationsController {

    @GetMapping("/find/{id}")
    public  ResponseEntity<Politician>  findPoliticianById(@PathVariable("id") int id){
          return  new ResponseEntity<Politician>(new Politician(id, "modi", "bjp", "PM",65.0f),HttpStatus.OK);
    }

}
```

```
//Politician.java (model class)
@Dataaa
@NoArgsConstructor
@AllArgsConstructor
public class Politician {
   private  int pid;
   private String pname;
   private  String party;
   private  String position;
   private  float age;
}
```

Client App/Consumer App

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

```
@Component
public class JsonDataConvertorRunner implements CommandLineRunner {
        @Autowired
        private  RestTemplate template;

        @Override
        public void run(String... args) throws Exception {
                //prepare  service  url
                String serviceUrl="http://localhost:2020/SpringRestProj15-ProviderApp-JSONtoObject/politician/api/find/{id}";
          //invoke  rest api/provider  service
                ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.GET, null, String.class,1001);
                //analyze the response
                System.out.println("response content/body::"+response.getBody());
                System.out.println("response status code::"+response.getStatusCode()+"   "+response.getStatusCodeValue());
                System.out.println("response  headers ::"+response.getHeaders());
                System.out.println("_____________________________________________");
                //convert the recived json response content into   Java class object  usng  jackson api (DeSerialization)
                 ObjectMapper mapper=new ObjectMapper();
                 Politician politician=mapper.readValue(response.getBody(), Politician.class);
                 System.out.println(" json to object convertion data ::"+politician);

        }

}
```

=>if Consumer And provider apps are placed in two different machines and developed by two different developers the cosumer app developer gets the structure of   Model class through swagger api documentation.

Converting List of Json docs (1D array like of List of Policiticians) given Rest API/Provider App method to List<Politician> collection in the cosumer app

Converting Map json docs (2d Array) given by Rest API/provider App method to Map collection in cosumer app

==

RestController in Provider App

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

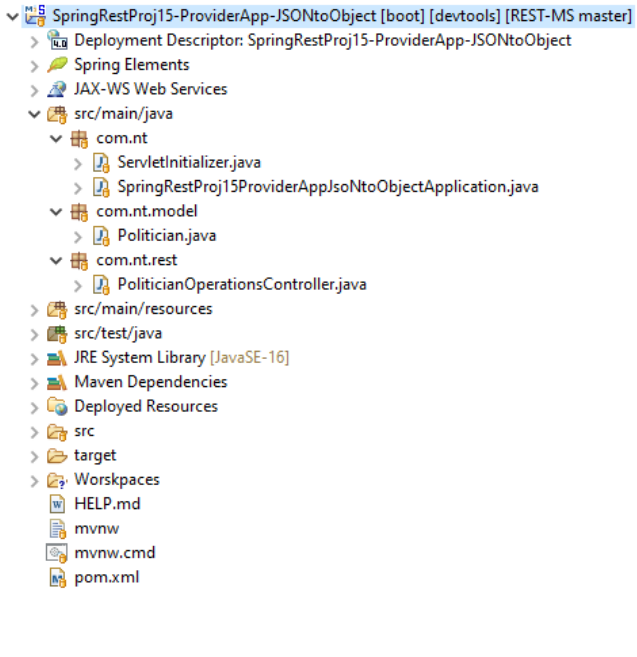
```
@RestController
@RequestMapping("/politician/api")
public class PoliticianOperationsController {

        @GetMapping("/find/{id}")
        public   ResponseEntity<Politician>  findPoliticianById(@PathVariable("id") int id){
                 return  new ResponseEntity<Politician>(new Politician(id, "modi", "bjp", "PM",65.0f),HttpStatus.OK);
        }

        @GetMapping("/report")
        public  ResponseEntity<List<Politician>>  showAllPoliticians(){
                List<Politician> list=List.of(new Politician(1001, "modi", "bjp", "pm", 67.8f),
                                              new Politician(1002, "amitsha", "bjp", "hm", 67.8f),
                                              new Politician(1003, "yogi", "bjp", "cm", 61.8f));
                return  new ResponseEntity<List<Politician>>(list,HttpStatus.OK);

        }

        @GetMapping("/report1")
        public  ResponseEntity<Map<String,Object>>  showIdCards(){
                Map<String,Object> map=Map.of("aadhar",54554353,"voterId",35435345,"passport",435435454);
                return  new ResponseEntity<Map<String,Object>>(map, HttpStatus.OK);
        }
}
```

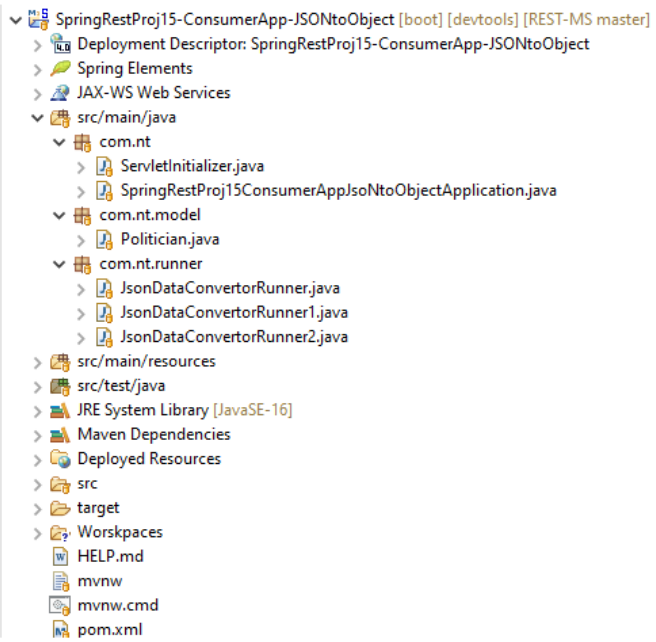
Runners in Consumer App

==========

```
@Component
public class JsonDataConvertorRunner implements CommandLineRunner {
	@Autowired
	private RestTemplate template;

	@Override
	public void run(String... args) throws Exception {
		//prepare service url
		String serviceUrl="http://localhost:2020/SpringRestProj15-ProviderApp-JSONtoObject/politician/api/find/{id}";
	  //invoke rest api/provider service
		ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.GET, null, String.class,1001);
		//analyze the response
		System.out.println("____________________<T>____________________");
		System.out.println("response content/body::"+response.getBody());
		System.out.println("response status code::"+response.getStatusCode()+"  "+response.getStatusCodeValue());
		System.out.println("response  headers ::"+response.getHeaders());
		System.out.println("_____________________________________________");
		//convert the recived json response content into  Java class object  usng  jackson api (DeSerialization)
		ObjectMapper mapper=new ObjectMapper();
		Politician politician=mapper.readValue(response.getBody(), Politician.class);
		System.out.println(" json to object convertion data ::"+politician);
	}
}

@Component
public class JsonDataConvertorRunner1 implements CommandLineRunner {
	@Autowired
	private RestTemplate template;

	@Override
	public void run(String... args) throws Exception {
		//prepare service url
		String serviceUrl="http://localhost:2020/SpringRestProj15-ProviderApp-JSONtoObject/politician/api/report";
	  //invoke rest api/provider service
		ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.GET, null, String.class);
		System.out.println("____________________List<T>------------");
		//analyze the response
		System.out.println("response content/body::"+response.getBody());
		System.out.println("response status code::"+response.getStatusCode()+"  "+response.getStatusCodeValue());
		System.out.println("response  headers ::"+response.getHeaders());
		System.out.println("____________JSON to Object convertion_List<T>____________________");
		ObjectMapper mapper=new ObjectMapper();
		Politician  politicians[]=mapper.readValue(response.getBody(), Politician[].class); //converts  1D array to  java array
		List<Politician> list=Arrays.asList(politicians);
		list.forEach(System.out::println);
		System.out.println("....................");
		List<Politician> list1=mapper.readValue(response.getBody(), new TypeReference<List<Politician>>() {});
		list1.forEach(System.out::println);
	}
}
```

```java
@Component
public class JsonDataConvertorRunner2 implements CommandLineRunner {
	@Autowired
	private  RestTemplate template;

	@Override
	public void run(String... args) throws Exception {
		//prepare  service  url
		String serviceUrl="http://localhost:2020/SpringRestProj15-ProviderApp-JSONtoObject/politician/api/report1";
	  //invoke  rest api/provider  service
		ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.GET, null, String.class);
		System.out.println("_____Map<String,Object>----------");
		//analyze the response
		System.out.println("response content/body::"+response.getBody());
		System.out.println("response status code::"+response.getStatusCode()+"   "+response.getStatusCodeValue());
		System.out.println("response  headers ::"+response.getHeaders());
		System.out.println("_____JSON to Object convertion_Map<String,Object>_____");
		ObjectMapper mapper=new ObjectMapper();
		Map<String,Object> map=mapper.readValue(response.getBody(), new TypeReference<Map<String,Object>>() {});
		System.out.println("id details map::"+map);

	}
}
```





21.NTSPBMS615- july 4th-Spring MVC and spring Rest Comminication

//Converting json content given rest api/provider App into Model with HAS -property pointing other model class

==========

Provider App's RestController class

==========

```
@GetMapping("/find/{id}")
public   ResponseEntity<Politician>  findPoliticianById(@PathVariable("id") int id){
        return  new ResponseEntity<Politician>(new Politician(id, "modi", "bjp", "PM",65.0f,
                                                    new Address("1-23","RK street",500070L)),
                                                HttpStatus.OK);
}
```

Sending model class obj having HAS-A property

Model classe in provider app

//Address.java

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Address {
 private String doorNo;
 private  String street;
 private  Long  pinCode;
}
```

//Politician.java

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Politician {
   private  int pid;
   private String pname;
   private  String party;
   private  String position;
   private  float age;
   //HAS -A property
   private Address addrs;
}
```

Runner class in Consumer app

```
@Component
public class JsonDataConvertorRunner implements CommandLineRunner {
      @Autowired
      private  RestTemplate template;

      @Override
      public void run(String... args) throws Exception {
            //prepare  service  url
            String serviceUrl="http://localhost:2020/SpringRestProj15-ProviderApp-JSONtoObject/politician/api/find/{id}";
        //invoke  rest api/provider  service
            ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.GET, null, String.class,1001);
            //analyze the response
            System.out.println("_____<T>_____");
            System.out.println("response content/body::"+response.getBody());
            System.out.println("response status code::"+response.getStatusCode()+"   "+response.getStatusCodeValue());
            System.out.println("response  headers ::"+response.getHeaders());
            System.out.println("_____");
            //convert the recived json response content into   Java class object  usng  jackson api (DeSerialization)
             ObjectMapper mapper=new ObjectMapper();
             Politician politician=mapper.readValue(response.getBody(), Politician.class);
             politician.setRetirementAge(politician.getAge()+20.0f);
             System.out.println(" json to object convertion data ::"+politician);

      }

}
```

model class in consumer app

//Address.java

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Address {
 private String doorNo;
 private  String street;
 private  Long  pinCode;
}
```

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Politician {
   private  int pid;
   private String pname;
   private  String party;
   private  String position;
   private  float age;
   private float  retirementAge;
   private Address addrs;
}
```

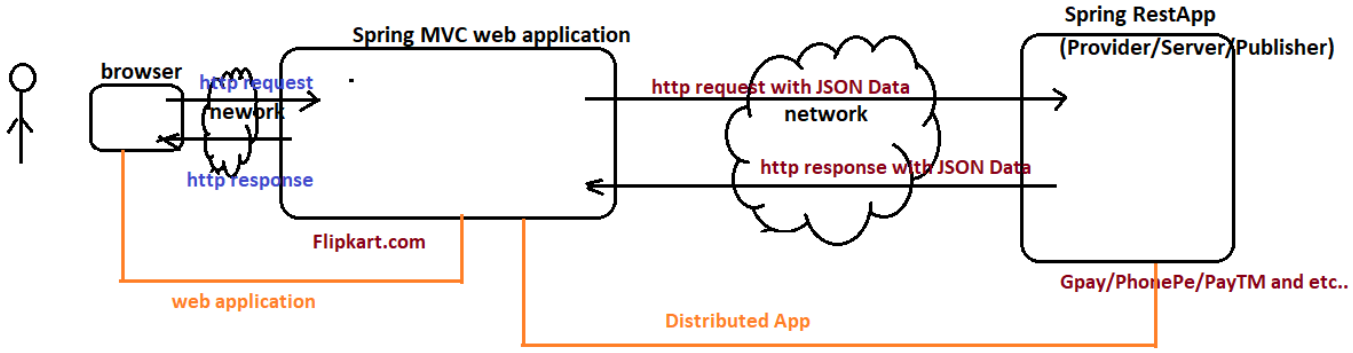
Extra property..

Mini Project using spring MVC and Spring Rest

=>Spring Rest App distributed app of type Restfull webServices .. The provider/server/ publisher app services of Spring Rest can be used from different types of Client apps either locally or remotely .. The different types of Client Apps are

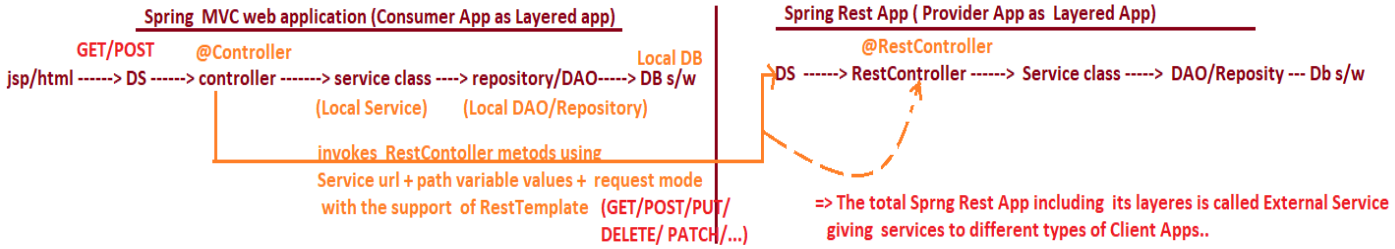
a) web applications
b) Andriod mobile apps
c) IOS mobile apps
d) IOT apps
e) Desktop Apps
f) IBM MainFrame machines
g) IVRS Apps

all these apps can be developed either in Java or not java..



=> Web serivces allows to develop the provider and cosumer apps in different languages using different apis i.e provider app can be in one language and consumer app can be in another language This web services comps as interoperable comps.

=>The spring MVC web application can have its own layers as consumer app
=>The spring Rest App can have its own layers as Provider App.



=> The Service cosumer side b.logics + persisitence logic like flipkark's bill calculations, inventory management , shopping card management takes place using Local serivce, DAO/Repository and Db s/w)

What is the difference b/w exchage(-) and delete(-) method of RestTemplate?

=>delete(-) method can send only delete method request to provider from Consumer App where as exchange(...) can send different modes of requests..

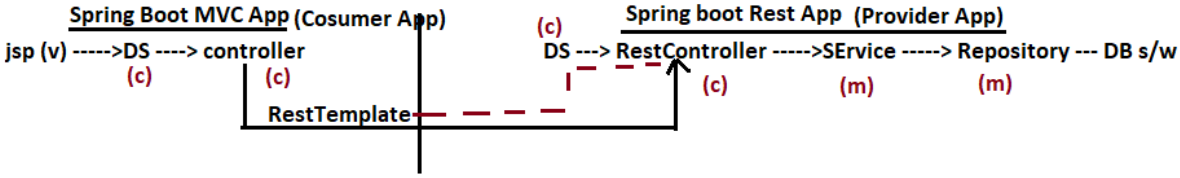
=>delete(-) method return type is void , So theConsumer App can not get any result/response from provider while sending delete mode reuquest where as exchange(...) method return type is ResponseEntity<String> .. so we can get result/response provider/server app.

What is the difference b/w exchage(-) and put(-) method of RestTemplate?

Ans) same as above

Spring Boot MVC ----- Spring Boot Rest based Mini Project development
(Cosumer App) (Provider App)
==========

=> we take already developed SpringBootRest based Mini Project as the Provider App
So we need to develop Spring boot MVC App as the Consumer App.



to
Steps devlope consumer app as the spring boot MVC App
==========

step1) create spring starter project adding the following starters
web , lombok , devtools ,*jstl*
|--collect from mvnrepository.com

step2) Develop the controller class having handler with @GetMapping("/") to launch the home.jsp as the home page

```
@Controller
public class ActorConsumerController {
    @Autowired
    private RestTemplate template;

    @GetMapping("/")
    public  String showHome() {
        //return LVN
        return  "home";
    }

}
```

step3) In main class configure RestTemplate class as the spring bean using @Bean method

```
@SpringBootApplication
public class SpringMvcProj16ConsumerAppApplication {

    @Bean(name="template")
    public  RestTemplate createTemplate() {
        return new RestTemplate();
    }

    public static void main(String[] args) {
        SpringApplication.run(SpringMvcProj16ConsumerAppApplication.class, args);
    }

}
```

step4) Configure viewResolver (default Internal ResourceViewResolver) by adding entries in application.properties file

```
application.properties
-------------------------------
#view Resolver cfg
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp
```

step5) add home.jsp in webapp\WEB-INF\pages folder

```
home.jsp
------------
<h1 style="text-align:center"><a href="actor_report">Show All Actors</a></h1>
```

in application.properties

```
fetchAllActors.serviceurl=http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/report
```

step6) add handler method in @Controller class that uses RestTemplate to invoke Provider App service/operation to get All the actors

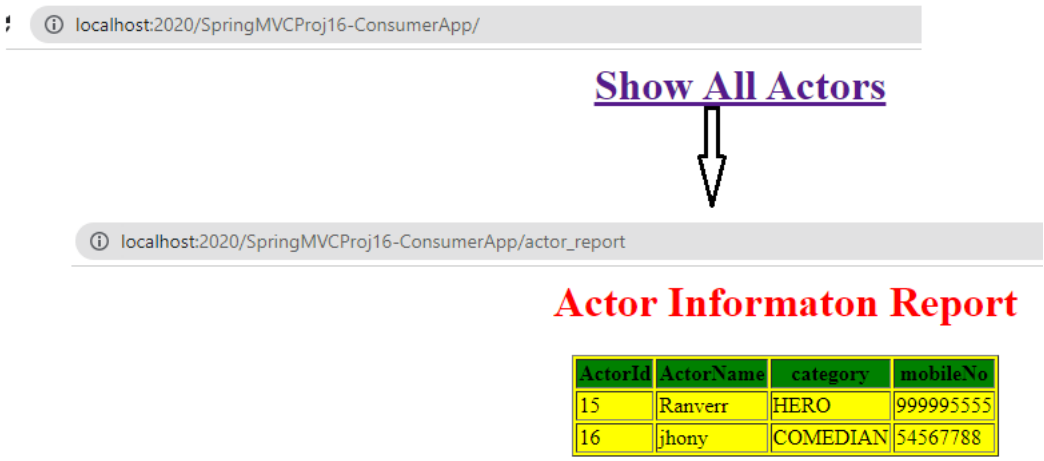
```
@Autowired
private Environment  env;

@GetMapping("/actor_report")
public  String   fetchAllActors(Map<String,Object> map)throws Exception{
        /*
         * provider url :: http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/report
         * req method/mode :GET
         * path variables :: no
         *  response content type ::  application/json (default)
         *  request haders : no
         *  request body type : no
         */
        String serviceUrl=env.getProperty("fetchAllActors.serviceurl");
        //invoke  Provider -Restcontroller operation/method using exchange(...) of RestTemplate
        ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.GET,null,String.class);
        // get json reponse from response obj
        String jsonBody=response.getBody();
        //convert  json body into List<Actor> object
        ObjectMapper mapper=new ObjectMapper();
        List<Actor> list=mapper.readValue(jsonBody,new TypeReference<List<Actor>>() { });
        //add result to  map object (SharedMemory b/w  Controller, DS and View comps)
        map.put("actorsInfo", list);
        //return LVN
        return "show_report";
}
```

step6) develop show_report.jsp page in WEB-INF/pages folder by using jstl tags

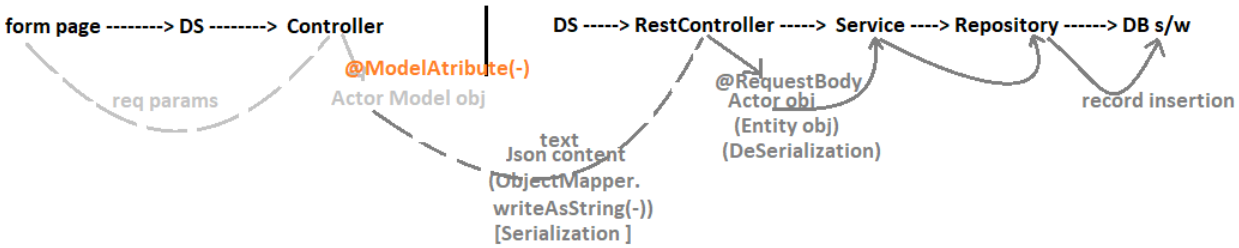
show_report.jsp

```
<%@page isELIgnored="false"  %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core"  prefix="c" %>

<h1 style="color:red;text-align:center"> Actor Informaton Report</h1>

<c:choose>
  <c:when test="${!empty actorsInfo}">
    <table border="1" align="center" bgcolor="yellow">
     <tr bgcolor="green">
       <th> ActorId </th>  <th> ActorName </th> <th> category </th> <th> mobileNo </th>
     </tr>

<c:forEach var="artist" items="${actorsInfo}">
  <tr>
   <td>${artist.actorid} </td>
   <td>${artist.actorname} </td>
   <td>${artist.category} </td>
   <td>${artist.mobileNo} </td>
  </tr>
</c:forEach>
</table>
 </c:when>
 <c:otherwise>
        <h1 style="color:red;text-align:center">Actors  not found </h1>
 </c:otherwise>
</c:choose>
```

=> **Run the Provider app first on extenal Tomcat server**
Run the consumer App next on same external tomcat server with removing previous App



Add operation using springboot MVC interaction with Spring Boot Rest Application



step1) add @GetMapping("/actor_add") in controller class of Consumer App to launch the form page of register actor

```
@GetMapping("/actor_add")
public String showRegisterActorFormPage(@ModelAttribute("actor") Actor actor) {
    //return LVN
    return "register_actor";
}
```

step2) Design register_actor.jsp as form page for actor registration in webapp\WEB-INF\pages folder

```
<%@page isELIgnored="false" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="frm" %>

<h1 style="color:blue;text-align:center"> Actor registration form page </h1>
<frm:form action="actor_add" modelAttribute="actor" method="POST">
  <table bgcolor="cyan" align="center">
    <tr>
      <td>actor name :: </td>
      <td><frm:input path="actorname"/> </td>
    </tr>
    <tr>
      <td>category :: </td>
      <td><frm:input path="category"/> </td>
    </tr>
    <tr>
      <td>MobileNo :: </td>
      <td><frm:input path="mobileNo"/> </td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="register"> </td>
    </tr>
  </table>
</frm:form>
```

must match with model attribute name

Must match with model class property names

application.properties

```
fetchAllActors.serviceurl=http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/report
registerActor.serviceurl=http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/save
```

In show_report.jsp

```
<h1 style="color:green;text-align:center">${resultMsg}</h1>
```

step3) add handler method in Controller class of Consumer app having @PostMapping("/actor_add") invoking Provider App operation/method to register the actor

```
@PostMapping("/actor_add")
    public String registerActor(@ModelAttribute("actor") Actor actor,
                                RedirectAttributes attrs)throws Exception {
        //convert model class obj data to json content using ObjectMapper
        ObjectMapper mapper=new ObjectMapper();
        String jsonContent=mapper.writeValueAsString(actor); //Serialization
        System.out.println("registerActor::"+jsonContent);
        //invoke proivder App's operation to save actor object

        /* provider url :: http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/save
         * req method/mode :POST
         * path variables :: no
         * response content type :: text/plain
         * request haders : contentType : application/json (default)
         * request body type : json content */

        String serviceUrl=env.getRequiredProperty("registerActor.serviceurl");
        // prepare Http Headers
        HttpHeaders headers=new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        // prepare HttpEntity object having headers + body
        HttpEntity<String> entity=new HttpEntity<String>(jsonContent,headers);
        ResponseEntity<String> response=template.exchange(serviceUrl,
                                                          HttpMethod.POST,
                                                          entity,
                                                          String.class);
        // get provider operation resuslt
        String msg=response.getBody();
        //keep result in model attribute(shared memory)
        attrs.addFlashAttribute("resultMsg", msg);

        //return "forward:actor_report";  POST mode request can not forwared to GET mode reuquest url (gives erorr)
        return "redirect:actor_report";  //we can use flash attirbutes in this method and also in redirected requested related methods/pages.
    }
```

update operation in spring boot mvc interaction with spring rest

==========

step1) place handler method in Spring boot mvc controller class to launch update form page having old values of Actor Information

```
@GetMapping("/actor_edit")
    public String showUpdateFormPage(@RequestParam("aid") int id,
                                     @ModelAttribute("artist") Actor actor)throws Exception {
        // invoke provider App (Spring RestApp) operation that gives actor infor based on actor id
        /* provider url :: http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/getactor/{id}
         * req method/mode :GET
         * path variables :: {id}
         * response content type :: application/json (default)
         * request haders : no
         * request body type : default */
        String serviceUrl=env.getRequiredProperty("updateActor.serviceurl1");
        ResponseEntity<String> response=template.exchange(serviceUrl,HttpMethod.GET,null,String.class,Map.of("id",id));
        //convert json response body to Model class obj
        ObjectMapper mapper=new ObjectMapper();
        Actor actor1=mapper.readValue(response.getBody(), Actor.class);
        //copyy actor1 object data to ModelAttribute Actor obj
        BeanUtils.copyProperties(actor1, actor);
        //return LVN
        return "update_actor";

    }
```

step2) develop update_actor.jsp page in WEB-INF/pages folder

```
<%@ page isELIgnored="false" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="frm"%>

<h1 style="color:blue;text-align:center"> Actor Information updation form page </h1>
<frm:form action="actor_edit" modelAttribute="artist" method="POST">
  <table bgcolor="pink" align="center">
  <tr>
     <td>actor id :: </td>
     <td><frm:input path="actorid" readonly="true"/> </td>
   </tr>
    <tr>
     <td>actor name :: </td>
     <td><frm:input path="actorname"/> </td>
   </tr>
   <tr>
     <td>category :: </td>
     <td><frm:input path="category"/> </td>
   </tr>
   <tr>
     <td>MobileNo :: </td>
     <td><frm:input path="mobileNo"/> </td>
   </tr>
   <tr>
     <td colspan="2"><input type="submit" value="update Actor"> </td>
   </tr>
  </table>
</frm:form>
```

step3) develop another handler method in spring boot mvc controller class having @PostMapping("/actor_edit") to update the actor information by taking the form submission request .. This method internally calls @PutMapping(-) of RestAPI comp

```
@PostMapping("/actor_edit")
public String updateActor(@ModelAttribute("actor") Actor actor,
                          RedirectAttributes attrs)throws Exception {
        //convert model class obj data to json content using ObjectMapper
        ObjectMapper mapper=new ObjectMapper();
        String jsonContent=mapper.writeValueAsString(actor); //Serialization
        //invoke proivder App's   operation to update the actor

        /* provider url :: http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/modify
         * req method/mode :PUT
         * path variables :: no
         *  request haders : contentType : application/json (default)
         *  request body type :  json content
         *   response content type ::  plain text
         *  */

    String serviceUrl=env.getRequiredProperty("updateActor.serviceurl2");
    // prepare Http Headers
    HttpHeaders headers=new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    // prepare HttpEntity object having headers + body
    HttpEntity<String> entity=new HttpEntity<String>(jsonContent,headers);
    ResponseEntity<String> response=template.exchange(serviceUrl,
                                                      HttpMethod.PUT,
                                                      entity,
                                                      String.class);
    // get provider operation resuslt
    String msg=response.getBody();
    //keep result in model attribute(shared memory)
    attrs.addFlashAttribute("resultMsg", msg);
    //return "forward:actor_report";  POST mode request can not forwared to GET mode reuquest url (gives erorr)
    return "redirect:actor_report";  //we can use flash attirbutes in this method and also in redirected requested related methods/pages.
}
```

DELETE Operation from Spring Boot MVC Cosumer App by invoking the operation of spring Rest App

step1) add handler method in MVC controller class invoking RestComp mehtod for actor deletion

```
@GetMapping("/actor_delete")
    public String deleteActor(@RequestParam("aid") int id,
                              RedirectAttributes attrs)throws Exception{
        //invoke provide'r RestAPI Comp operation
        /* provider url :: http://localhost:2020/SpringRestProj16-ProviderApp-Actor/actor/api/delete/{id}
         * req method/mode :DEELTE
         * path variables :: {id}
         * request haders : no
         * request body type : no
         *  response content type :: plain text
         */
        String serviceUrl=env.getRequiredProperty("deleteActor.serviceurl");
        ResponseEntity<String> response=template.exchange(serviceUrl, HttpMethod.DELETE, null,String.class,Map.of("id",id));
        // keep the response body (plain text) in flash attributes..
        String msg=response.getBody();
        // add result as flash attribute
        attrs.addFlashAttribute("resultMsg", msg);
        //redirect the request
        return "redirect:actor_report";
    }
```

step2) add confirm dialog box in the report page (show_report.jsp) for delete operation

In show_report.jsp

```
<a href="actor_delete?aid=${artist.actorid}"
    onclick="return confirm(' do u want to delete?')"><b>delete</b> <img src="images/delete.jpeg" width="30px" height="30px"></a>
```

Spring Cloud

Spring Cloud (working with MicroServices)

Need of MicroServices

Monolithic Apps Vs SOA Apps Vs MicrorSErvices Apps

(spring mvc apps) (web services) (spring Rest+ Spring cloud)

In spring Boot the Micro service archicture Apps will be implemented with support of spring Cloud Module and multiple other facilities

=> Generally we do not prefer using spring framework for MicroService implemenations .. we prefer spring boot a lot in the angel of spring boot cloud to implement MicroServices..

Monolithic Apps

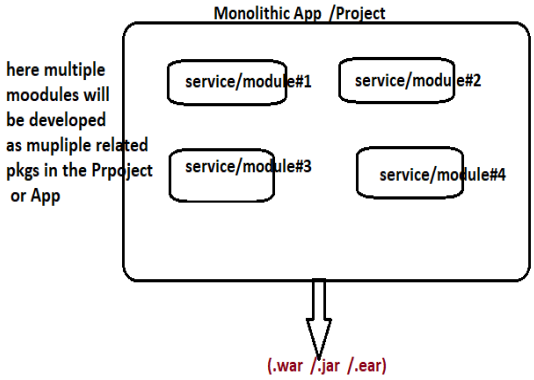
=>The Application the packs multiple services as multiple modules in a single unit (nothing but project) either as war file or jar file or ear file is called Monolithic Application/Project.

eg:spring web mvc apps ,spring boot mvc apps

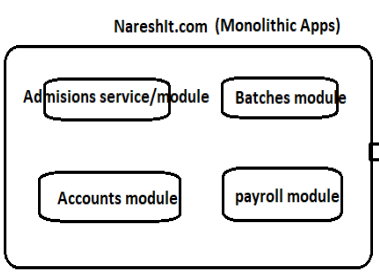
1 application = 1 Project contains multiple services as muitiple modules inside the project/Application

=>So far the spring MVC Apps, spring data jpa Apps , spring core Apps and etc.. we developed are called Monolithic Apps

=> Even spring MVC with sprign data JPA /spring ORM/spirng JDBC that we have developed so far (except Spring Rest Apps) are called Monolithic Apps..



eg:: Nareshlt.com , Flipkart.com, Banking Apps and etc..



=>we can not use each module/service of the project independently either directly or in other projects i.e monolithic apps maintains single code base.

.war ==>web application

=>All the apps we developed so far both web applications and standalone apps fall under monolithic apps .. except Spring Rest Apps becoz they are web services based on cosumer-producer architecture..

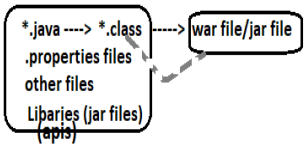
.jar ===> standalone App (Non-web applicaiton) / standalone web application (Using Embedded Server)

ear file = jar file + war file +..... (or)
ear file => jar file + jar file +jar file .... (or)
ear file => war file +war file +war file

ear file :: enterprise App archieve (now ear files not popular becoz web services comps also coming as war files .. When EJB was there ear files bit popluar to represent web applicaiton(war) + ejb comp (jar) togather as ear file

---

Build Process /Building the Application
==========
=>The process keeping the app ready for execution in certain env.. is called build process



note: we can use maven/gradle/ant and etc tools to make this build process happening in simple manner
So these tools are called build tools

=> In buid process , developmentn of source code , compilaation of source code , development operations helper resources like properties files and other files , adding api libaries and etc.. will be there .. At last the war file or jar file will be created reprsenting the whole project.

What is the difference App /Service API?

=>API --> application Programming Interface .. It is base for programmer to devleop certain language or technology or framework Apps. In Java api comes in the form of pkgs having classes,interfaces, enums and annotations.. These apis will be released /available in the form jar files (libaries/dependencies)

3 types of APIs :: a) pre-defifned apis (given by technology/language/framework vendor)
b) user-defined apis (given by developer)
c) third party apis (given by third party vendor)

=>Application/Service is the outcome of build process either in the form of jar file or war file having certain to task to perform.

note:: In the development App /Services in certain language/technology /framework we take the support APIs. App /service = APIs+ Build Process

note:: The existing APIS/libraries can be used to create new APIs/Libaries or projects/Apps/sevices

eg: we can use hibernate api directly to develop hibernate persinstece logic /hibernate App
the spring creators have used the same hibernate api to develop spring ORM apis and spring data jpa apis

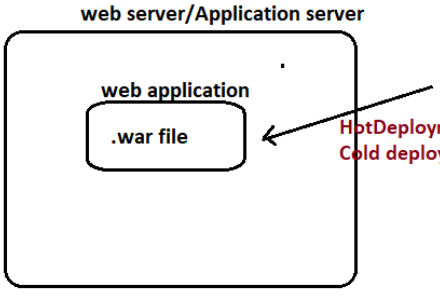
=>Rest Comps /provider Comps are also behaves likes APIs for consumers , So that consumers can use them in the development consumer apps as required for the Endusers .. Flikart uses PayTM , Paypall and etc.. RestAPIs/comps for payment.

=>Jar file purpose will change context to context..
jar repsents apis
jar represents jdbc driver
jar represents standalone web application
jar represents ejb comp
jar represents web application (war file)
jar represents standalone App/project
and etc...

=>Normal java apis avaiable with us locally and helps us to develope different types of apps
=>Rest apis avaiable in Remote places and helps us to develope different types of cosumer apps

Deployment /Installation
==========

=>The process of keeping web application (.war file) in the server (either that is running or later started) is called deployment..



HotDeployment :: Deploying the web application when the server is running mode
Cold deployment :: Deploying the web application when the server is in stopped mode

note:: In the devleopment /Testing env.. deployment takes place in Local servers of s/w companies or s/w company cloud account (aws/ gcp/azure and etc..)
note: In the UAT and Production the deployment takes place in the Client Org Servers or Client org's Cloud Account

How build Process takes place in Realtime companies?

Ans) using Maven/ gradle Tool In combination with Jenkins CI/CD configration + Docker

CI :: Continous intergration
CD :: Continous Deployment

How do we pack app/service/project in to single unit having entire env.. for execution?

=>Only coding packing -----> war /jar file
=> Envirormment packing ---> docker image
(code(war/jar) + server + DB + OS + ...)

IDE +maven/gradle ----->code ----> jenkinks +CI+CD ---> war file(code packing) ----> Env.. packing (Docket image)

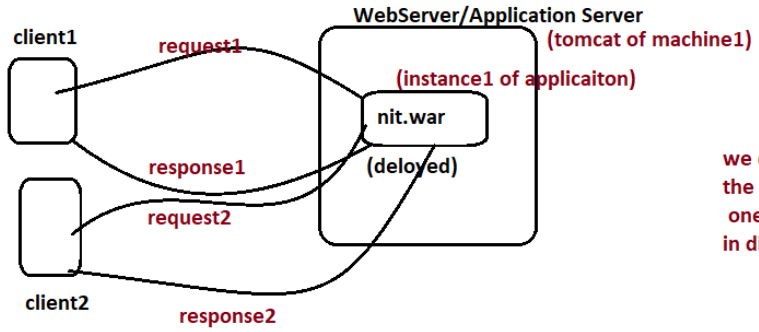
Service Instance / App Instance /instance
==========

watch this vedios::
===Tools used in Modern Build and Deployment | by Mr. Nataraj ====
https://youtu.be/5oemIoT3KEc

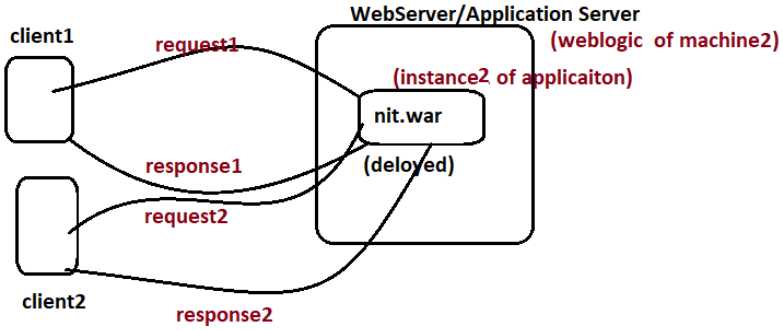
=>A runnig Application inside the server giving services to Clients (either for endusers or for other Apps) is callled Service Instance /instance /App instance..
=> Each copy of app(jar or war) that is deployed in server successfully to render services for clients is called is called service Instance..

App/Project ---> is like class
each deployed App/Project giving services to clients is like instance
note:: One class can have multiple instances.. simpliarly one app/project can deployed in multiple servers i.e can have multiple instances/Service instances

nareshIT web application (App /Project) => (complted devleopment/testing)
(nit.war)



we can deploy the
the multiple instances /sevice instances of
one App/Project either in same server or
in different servers of same machine or different meachiens



In Every server we can specify max no.f requests that each service can allow, In most of the server this max request count is 200 by default

In spring boot MVC or spring Rest Apps we can change /control this max request count using the following properties of application.properties

In application.properties
---
server.tomcat.threads.max= 150 (default is 200)
server.jetty.threads.max= 300 (default is 200)

Load Count /Current Load
==========

=> The no.of requests that are currently under proccessing by service instance is called Load Count /Current Load.
=> if the App1 instance/service instance is currently processing 10 requests then the Load count /Current Load is 10.

Load Factor
==========
=> it is current Load/Max Load

=> Load factor = courrent Load/ max Load

=> if App instance with respect to server max load is 150 and that app instance is processing only 100 requests currently then 100/150 is the Load Factor (0.666)

Load Factory is always >=0 to <=1

maxLoad = max requests that server can take for each instance of the Application at a time
we can specify this using server settings or using application.properties while working with embedded servers of spring boot.

Scaling
=======
=> Increasing the service capacity of App/SErvice /Project is called Scaling..

=> if the App is having facility to increase its service capacity as needed then it called scalable App.. (scalability feture)

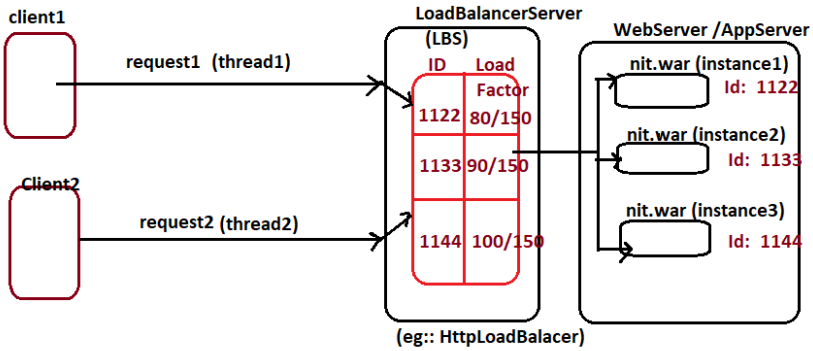
Examples :: eg1:: increasing shop capacity from 1000 square feet to 10000 square feet (vertical scaling)
eg2:: keeping same shop of 1000square feet in 10 places. (Horizontal scaling)

eg3:: increasing no.of ATM machies in different locations (Horizontal scaling)

Two types of Scaling
==========
a) Horizontal Scaling (good)
b) Vertical Scaling

a) Horizontal Scaling

In Horizontal scaling we take the multiple copies same App i.e multiple service Instances to give services to Clients.. and we control these multiple instances with the support of Load Balancer Service (LBS)



These instances
can be there either
in same server different domains
or different copies of same server or different servers
residing in the same machine or different machines,

=> The given request goes to that instance/service instance of the App whose LoadFactor is less (whose value is nearer 0) .. In our example the both req1,req2 will go to instance1 beoz it is having less load factor (80/150). if all the instances are having same load factor then the LBS will pick up the instance randomly..

Examples for different LoadBalancer
==========

- HAProxy – A TCP load balancer.
- NGINX – A http load balancer with SSL termination support. ...
- mod_athena – Apache based http load balancer.
- Varnish – A reverse proxy based load balancer.
- Balance – Open source TCP load balancer.
- LVS – Linux virtual server offering layer 4 load balancing.

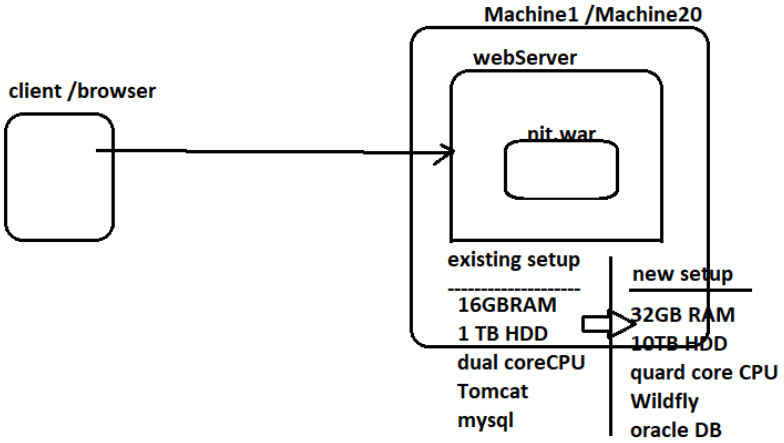
and etc..

Keeping gmail App in different zones of world map falls under horizontal scaling.

Either DevOPs team or Infrastucture team like Linux admin and etc.. will take care of this kind of Load Balancer..

b) Vertical Scaling

=> Here we add more software or hardware infrastructure for the existing App env.. to make it ready for taking more requests from more clients.



In Cloud env.. like AWS
we get max of 720 GB RAM
and unlimited HDD suport
can be taken..

---

of

Pros and cons Monolithic architecture based Applicaiton Development

==========

pros (advantages)

==========

=> Provides simple and easy env.. to develop Projects/Apps as the Layered Apps having layers like presnetation layer , controller layer , service Layer , Persistence Layer(DAO), Integration layer and etc..

note: Most of the current maintainance projects are Monolithic projects

=> Easy to deploy (all togather single jar/war file) , easy to manage (single war/jar file) , easy to scale (increasting or decresing instances)

=> Performing unit Testing is very easy becoz all layers and all services are avaiable togaher

=> Less possiblity of getting network related issues becoz all services /layers mostly there togather.

=> Performing logging and dubegging operations is very easy.. (Even Adutiing activities are easy)

d

cons (DisAvatages)

====

=> For small or tiny changes in one or two services /modules /layers we need redeploy the whole application by alterting the endusers when App in the production. (App will not is work for these many hours)

=> one bug or issue in one module/service of Project/App may effect other services /modules ..this indicates these Apps are not reliable. (Sometimes Apps will shutdown providing no services to clients)

=> Adding new technologies /concepts/ frameworks in the existing project as part of enhacement is very complex sometime we will be forced to redesign the project..

=> if no.of modules/services are incresed in the Project then their maintainace towards bug fixing and redeployment also takes lots of time while working with websErver and application server .. (This may cause increasing App/Project downtime)

=> we can not sell certain services /modules of one Project to clients as they needed

=> Since we can create service instances only for entire App .. not for certain services/modules of the Project lots of memory and CPU time wastage will be there.

Fatty horinzontal scaling is not good

and etc..

Problem::

=========

=>Application means Project

=>service means module

Monolithic Project/App (Flipkart)

search service (20MB)

cart service (20MB)

Order Service (20MB)

Tracking (20MB)

Return &Replace mentService (20MB)

Rewards Service (20MB)

war file

Total Project/App size is :: 120 MB

SErvice Instances created for the whole Project/Application (Horizontal scaling)

request capacity of each instance 250

Flipkart (120MB) (London)

50 req :: cartService
50 req :: search Service

Flipkart (120MB) (Dubai)

100 req :: search
100 req :: orders

Flipkart (120MB) (singapore)

50 req : cartService
50 req :: search service

Flipkart (120MB) (Blore)

100 req request cart service
50 req tracking service
40 req search service

Total Project Info
(All instance togather)

Size :: 480 MB

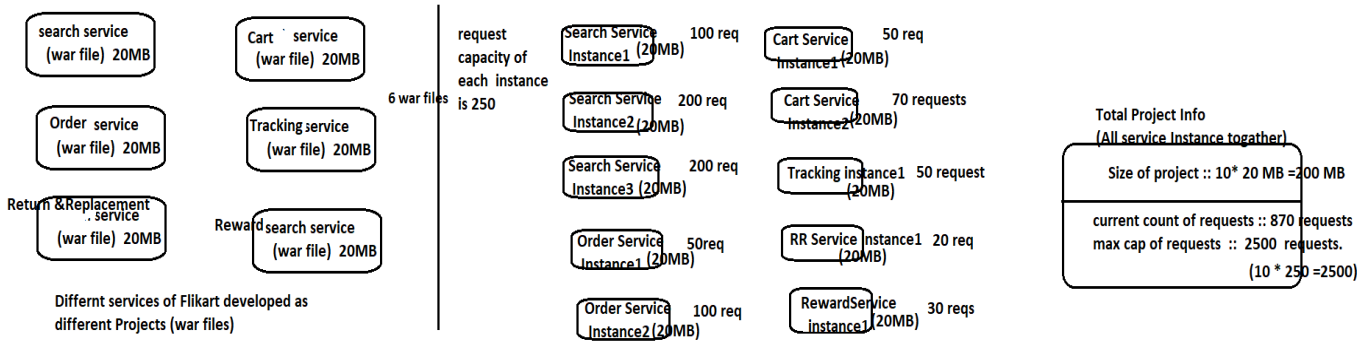
currently serving : 590 requests
max capacity :: 1000 request s
(4 * 250 =1000)

Solution::
=======

Let us assume we are able to create seperate project/App for every service/Module (Horizontal scaling by taking modules/sericces as the Projects/Apps)

==========

(Indirectly MicroServices architeture)



Differnt services of Flikart developed as different Projects (war files)

=>This kind of Application development in Monolithic architecture is possible but very complex to implement and manage .. To overcome this problem we have got MicroServices Archicture..

SOA (Service Oriented Architecture)
==========

SOAP based web service fall under SOA

Restfull webservices fall under Producer- Consumer mechanism

note:: MicrorServices Archicture is enhacment Producer-consumer mechanism where we develop each microservice as restfull App

What is the link between webServices and MicroServices?

Microservices are extension of webServices (Restfull webservices)
In fact each micro service will be developed as one Restfull webservice adding other facilities.

Monothilic vs SOA vs MicroServices archictures

=>SOAP based web services is based on SOA
=>Microservices App are based on RestFull webSErvices (Consumer and producer architecture)

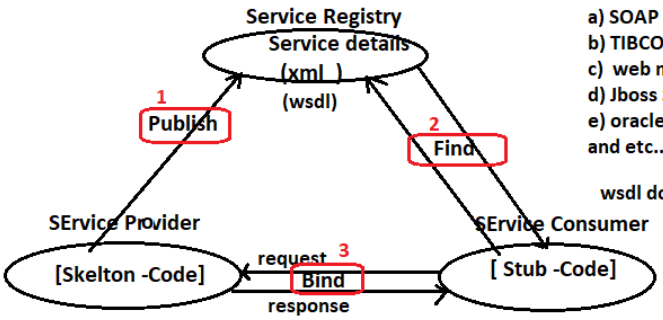
SOA (Service Oriented Architecture)

=> This architecture given to get communication between two applications the developed in two different technologies and running from two different machines using Register and Discovery Concepts..

=>SOAP based webServices (like Jax-ws, axis ,apche cfx and etc..), CGI-Links are given as the implmenentation models of SOA Design./ architecture

=>SOA is design that provides certain archicture with principles to get communication between two incompitable Apps..

3 comps SOA
- a) Service Provider
- b) Service Consumer
- c) Service Registry



There are multiple SOA implementations
- a) SOAP webservices
- b) TIBCO
- c) web methods
- d) Jboss SOA
- e) oracle SOA
- and etc..

wsdl doc :: WebService description language

1.Service Provide develops the prodiver App and exposes it details to serviceRegistry in the form of xml docs (Publish Operation)
(wsdl doc)

2. TheService consumer contacts the ServiceRegistry and get the details about provider or service (Find Operation)
(gets wsdl doc)

3. The service consumer prepare stub code to consum services of the Provider using request-response model (Bind Operation)

Advatntages of SOA

1) Interoperability :: we can develop the Provier and consumer Apps either in same technology or in different technologies

2) Easy Maintainace :: Any change in provider App , we just need to update to Service registry and need not inform to all consumers..becoz Service cosumer can collect the changes from SErvice Registry

3) Quality Code is guaranteed :: Since we can develop provider App in our choice technology So we can give quality provider App..

4) Scalable :: we can add more parellel servers or instanstaces for provider App as service consumers and their no.of requests are incresing

4) realiable :: Since the SOA machanism is involving the Service Registry .. we can maintain our Distributed App in reliable and stable state..

DisAdvantages ::

HighCost :: It needs lots of man power and technology /infrastrucure support for developing SOA applications

Xml utilization :: Supports only XML based Data exchange which is fading out day to day (understanding wsdl docs is very complex)

HighBandwidth of internet required :: Since service registry ,Service Providers will be placed in different locations we definelty need good bandwith of internet connections.

Service registry maintainace is higth : :: we generally use UDDI registries as service registry which quite costly to manage

Complext to learn and use :: Since SOAP based webservices use the high end xml schema and wsdl docs ro use ,, we can say implementing SOA apps is quite complext

MicroServices Architecture (Extesion Restufull web service model)

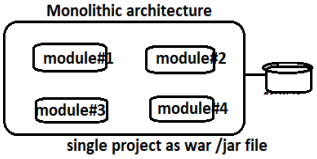
=>In this architecture every service /module will be developed as seperate project/App having connection with other service/module..

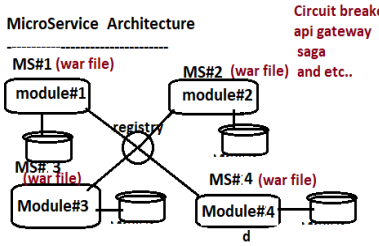
=> It is Decoupled Archicture of a single Project becoz multiple services/modules will be devleoped as multiple projects and they will be registered/Integrated in a common place. (That means we use spring rest Apps aling with common registry)

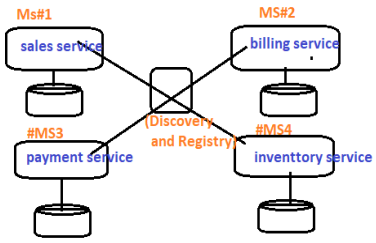
=> Micro SErvice is small service or small application
micro = small
service = project/ Application..

MicroSerivces App = Restful app ++
Spring cloud uses spring Rest + spring boot actuators other modules + Design Patterns
Circuit breaker
api gateway
saga
and etc..

MicroServices architecture is designed around special design patterns







Every Micro service will be developed as Restfull App /RestController and will be place in common registry/server to make it avaiable for other micro services or Client Apps.
(Eureka server)

Advatnatages of MicroSErvice Architecture
=========
=> Need not stop other services /modules (MicroServices) if problem comes in certain module/service
[if Rewards or Feebback sevrice failed it does not effect other services like sales , orders, cart and etc..]
=> Need not stop all other the services .. while updating /patching certain service for betterment

=>Since Each MS is restful app .. it will be deployed in the regular web server and application server (like tomcat, wildfly and etc..) .. and it also be registered to special Discovery and registry server like Eureka server..

=> We can use different Technologies to develop different services of the Project/application..

=> We can do horizontal scaling for each service as needed i.e if certain service like searching, cart and etc.. are having more demand then we can create more instances for them by enabling Load Balancing (Spring cloud provides ready made load balancers)

=> Updgrading/Migrating to new tehchnologies in each service is not complex .. Quite easy compare to monolithic architecture

=> Service Down time (while performing reloading or restart or redployment activities) is less becoz we need to perform these actitvies on one service at time.

=> Allows to place common things of multiple Services (MicorServices) in a common place like GIT hub env.. instead of placing in every service/ module (Look at GIT as central configuration place)

note:: Developing MicroServices using spring boot is very easy becoz most of the common things like DB setup , Schema initializations and etc.. can be cfg to occur through auto Configuration of spring boot
=> Though multiple services are developed as multipe Restfull Apps having differnet urls .. we can created common entry point for all the micro Services using sepcial Cfgs called API Gateways.

Limitations of MicroServices
==========
=>Maintainance of the Project is very complex becoz it generates multiple log files , contains multiple communication chains and debugging the code across multiple services is too difficult..

=> Needs high end intrastruture to maintain multiple micro services , if going for Cloud we need to purchase /rent muliple things to deploy and manage these projects.

=> Knowlege on More Tools , technlogies and processes is required to work with MicroServices Projects

=> Migrating Monothic Project into MicroServices project is very complex

=> Testing is bit difficult if one microservice is having dependency with other micro service.

note:: we generally uses Cloud env.. to deploy and execute the modren monolithic , SOA and Micro services projects.. Working with with Cloud is nothinb but taking things on Rental basis and using them for our requirements..

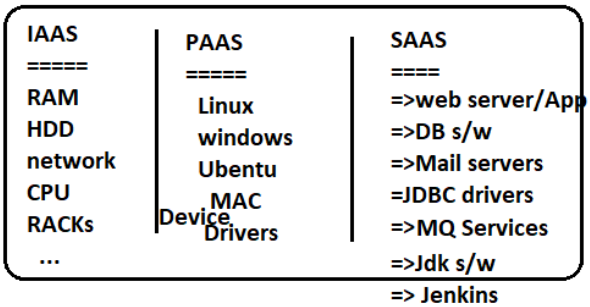
eg:: AWS , Azure , Goolge clould, PCF (Pivotal Cloud Foundary) , RackSpace and etc..

(no more further support)

Cloud services

a) PAAS :: Platform As A service
b) IAAS :: Infrastrutur As A service
c) SAAS : Software As A service

Clould env.. (AWS/Azure ...)



=> MicroServices architeture is Design and whose Apps/Projects can be implemented/ developed in spring env.. with the support of spring boot + spring clould + tools

=> Green field project : Project started from scratch level
=> Brown field Project :: adding new features to existing project

reallife examples :: naya raipur ---> greenfield capital
chandigadh ---> greenfield capital
hyderabed , new delhi , mumbai ---> brown field capitals

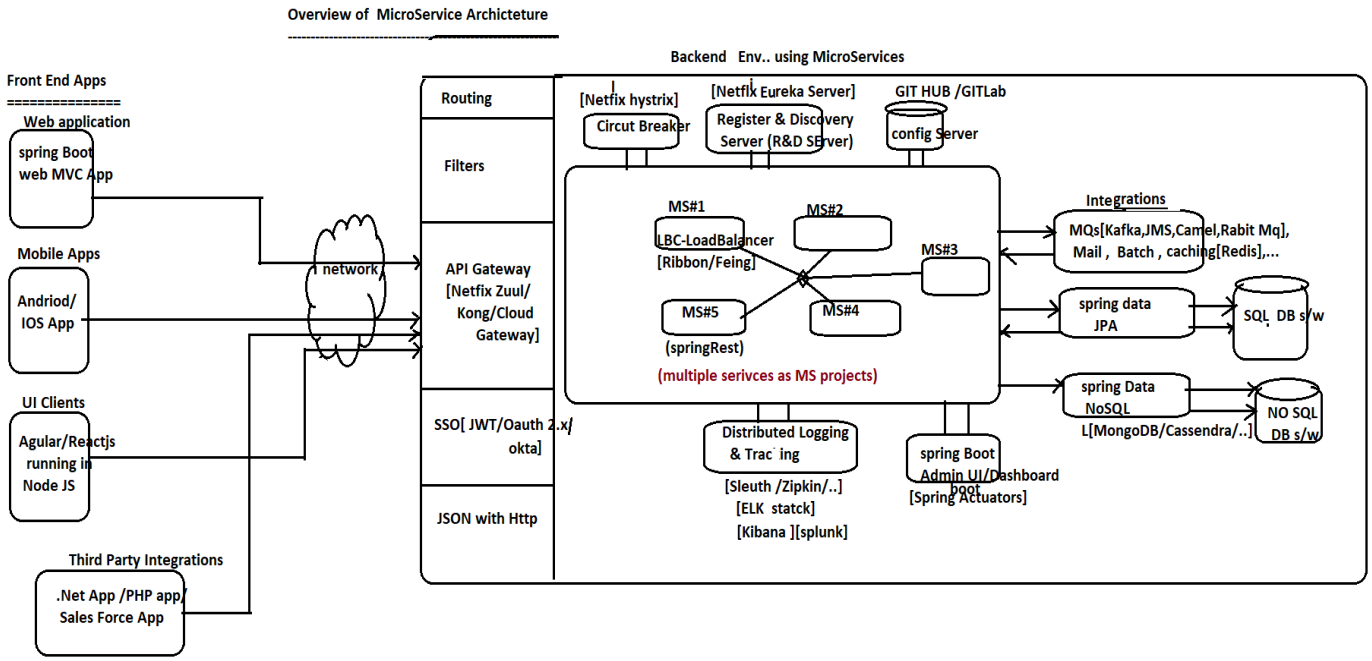
=>What is the difference WebService and AWS?

Ans) AWS -- Amazon web service :: it is cloud business platfor provding software,hardware services to the companies on rental /lease basis . It purely Infrastructure offerning organization in 3 forms PAAS, IAAS , SAAS

'

=>webSErvices is actually as specificc to develop frameworks to get App to app interaction by having data exchanging in global formats like XML , JSON and etc.. It is no way related to AWS and any other cloud accounts proivder/platforms,

23.NTSPBMS615- July 16th-2022- Spring Cloud-MicroServiceARchitecture



Micro service Architecture is Desinging or specification which provides set of rules and guidelines to develop Project as set of loosely coupled /De coupled Services .. and this can implemented using spring boot +spring cloud + Netflix and lots of other tools.

=> Every module in the project will be develop ed as seperate microservices in the form of spring RestController with the support of spring boot env..

R & D servers

=> Once the Microservic architecture project is ready .. It will be deployed in cloud env..like AWS/AZure/Google Cloud and etc.. with DevOPs tools Jenkins with CI/CD + Docker + Ansible + Kubernate s +....

=>After developing each Micro SErvice as seperate Spring Rest App/Project it must be published in a common place called R & D SErver (Register And Discovery Server) like Netflix Eureka server /apache zoo keepr / cosul and etc..

=> One MicroSErvice can find another MicroService in R & D SErver and can be used for Communication Through the same old R & D Server..

=> One MicroService can find and communicate with another microservice only when it is published in R & D server ..

Config Server

=> The common properties with same values of Multiple Micro services can be placed outside the Micro services in place called Config server like GIT Hub , GITLab and etc..

Circuit Breaker

=> if any exception is raised in the execution of one microService then it has to be informed to Admin UI /Dashboard with the support of Circuit Breker like Netflix hystrix ,Resilience4J and etc..

Load Balancer Client

=> if any MicroService is having more demand then we allows to create multiple instances dynamically.. In that situation to pick up right istance with less Load factor from other MicroServices we take the support of Load Balancer Clients (LBC) like Ribbon ,Feign ,Http LoadBalancer and etc..

ntegrations

=> since we are devleoping every MicrService as Spring Rest App in spring boot env.. So we can make these Micro services Apps integrating with lots of other facilities like MQs (Message Queues), Mail, Caching, Batch PRocessing and etc..

DB interaction

=> MicroServices can interact with SQL Db s/ws using spring data jpa

=> MicroServices can interact with No SQL Db s/ws using spring data Nosql modules like spring data mongoDB ,spring data cassendra and etc..

use

spring boot admin UI/ Dash Board

=> To Monitor and Manage all the MircorSErvices of the Project .. we try to spring boot Admin UI/Dashboard that is created using support of spring boot Actuators which are also useful providign non-funcational features like Health metrics on the projects..

Distributed Logging/Tracing

=> Since Project contains multiple microSErvices interacting with each other..So we need perform logging and tracing activity across the multiple micro services as needed with the support Distributed Logging and trac ing tools like slueth and zipkin , kinbana , splunk ..

Different Front End apps

=> The MicroServices of the Project can have different types of Clients (Front end Apps) mobile Apps , web mvc Apps, Third party Apps , UI Technologies App and etc..

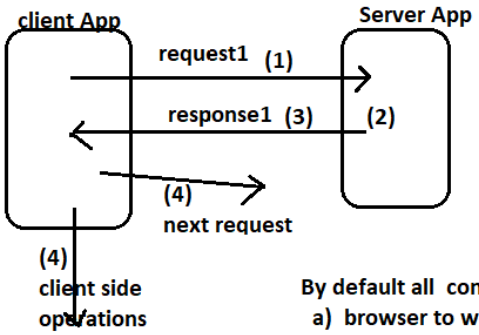
API Gateway

=> To use all these microServices and tools from different types of Clients /Front end Apps we need one common entry and exit point concept nothing but API Gateway like netflix zuul/Kong and etc.. providing facilities to apply Filters , Routers , SEcurity like SSO (Single Sign On) and etc..

=> Some times method calls based synchronous communication is not sufficient among the MicorSerivces becoz synchronous commication needs activate Microservices to participate in interaction .. To overcome thise problem we go for messages based asynchronous communication using the suppport of MQs (Message Queues) like kafka, rabit mq, active mq, jms and etc..

Synchronous communication

=> Here two parties of communication must be in active mode at a time
=> The client App should wait for Server App deliverd result/response in order to generate next request or to perform client side operation..

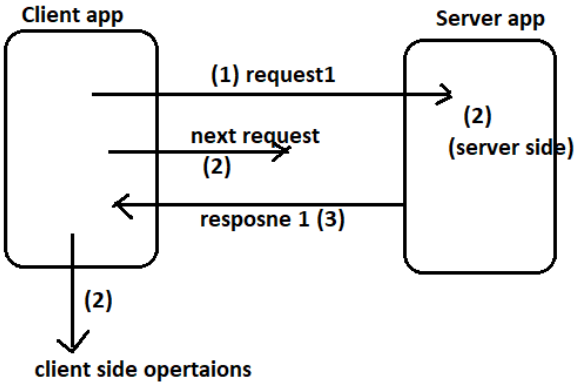


Here Client is not free to generate the next request or to perform' client side operation until given request related response comes back to Client app from Server App

By default all communications are synchronous commications
a) browser to web application b) Rest Api to rest api using RestTemplate and etc..

Asynchronous communication

=> Here two parties of communication need not be in activate state
=> Here Client App is free to generate the next request or to perform client side operations with out waiting for given request related response.



eg:: browser to web application using ajax
Java App to Java using JMS messsages, active mq ,....
Rest Api to Rest App using MQs (kafka , jms ,...) messages

24.NTSPBMS615- July 18th-2022- Spring Cloud-Eureka server

Spring Cloud - Netflix Eureka Server

=>Every MicroService must be registered or published with R & D (Register and Disocovery) server in order to make it discoverable/communicatable from other MicroServices

=> As of now netflix eureka server (best), apache zookeeper are two popular R & D Servers.

=> Every R & D server is spring boot Project having R &D Server dependencies (Do not except seperate installation like Tomcat, wildfly and etc..)

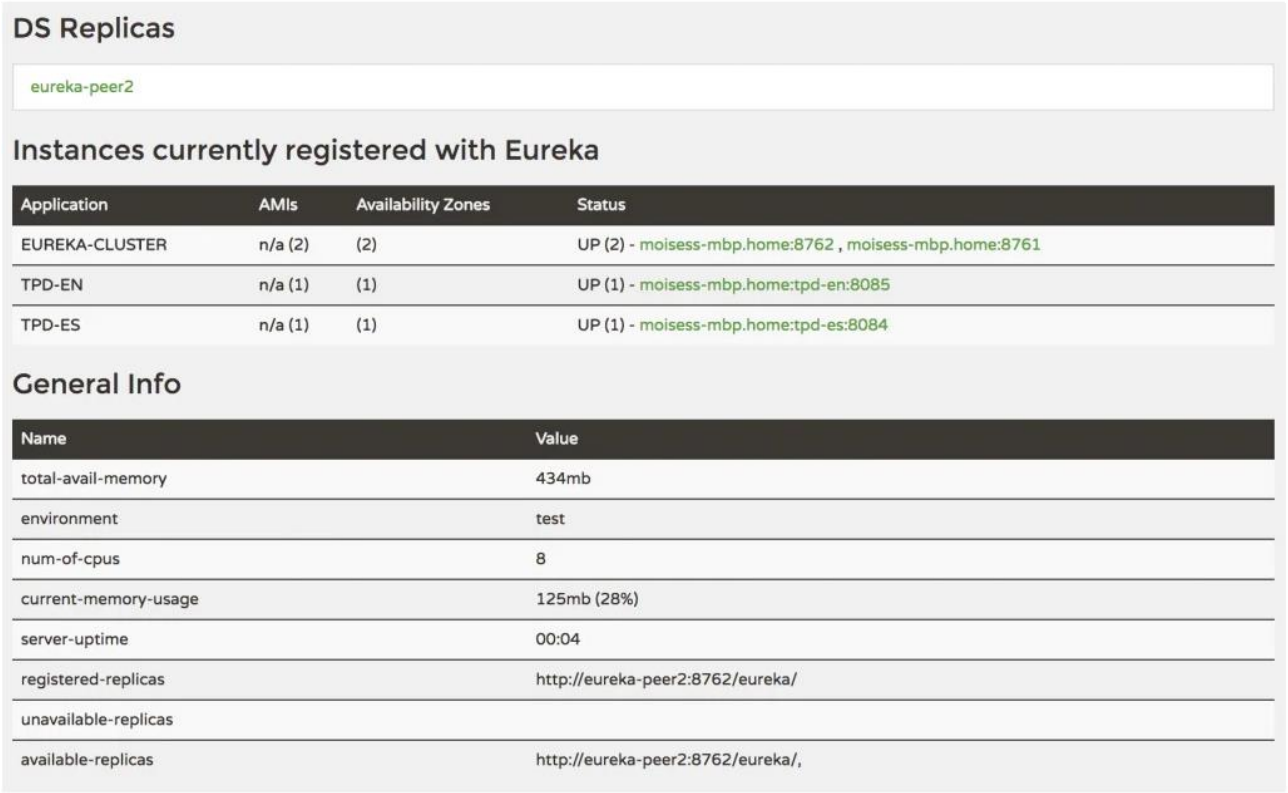
=> The Process of keeping /publishing MicroService details in R &D Server is called registration activity

=> The process of discovering/fetching MicroService details from R & D Server to esstablish communication/interaction from another MicroService is called Discovery Operation.

=> When we publish different instances of different micro services to R & D server like Eureka server .. then the details will look like this...

Service Id	InstanceId	HostName/IPAddrs	LoadFactor [CurrentLoad/ MaxLoad]
Search-Service	SS:566-a367	192.6.8.7 7878	0/200
Search-Service	SS:536-a461	192.6.8.7 7171	0/200
		IPaddrs port number	
Order-Service	OS: 455-a356	192.6.8.7 8989	0/200
Order-Service	OS: 415-a256	192.6.8.7 8182	0/200

=>Eureka Server is no documents server .. i.e it does not contain any xml files or Json files inside the server



DS Replicas

eureka-peer2

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLUSTER	n/a (2)	(2)	UP (2) - moisess-mbp.home:8762 , moisess-mbp.home:8761
TPD-EN	n/a (1)	(1)	UP (1) - moisess-mbp.home:tpd-en:8085
TPD-ES	n/a (1)	(1)	UP (1) - moisess-mbp.home:tpd-es:8084

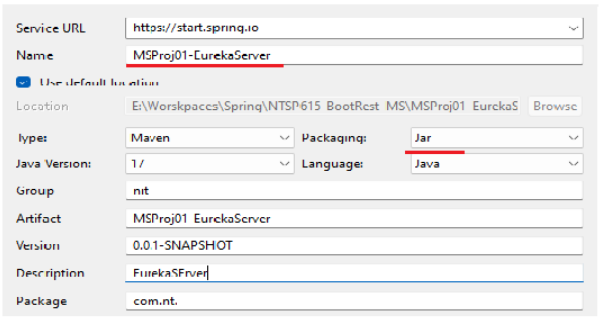
General Info

Name	Value
total-avail-memory	434mb
environment	test
num-of-cpus	8
current-memory-usage	125mb (28%)
server-uptime	00:04
registered-replicas	http://eureka-peer2:8762/eureka/
unavailable-replicas	
available-replicas	http://eureka-peer2:8762/eureka/,

=> Service Id is always Project name (Service Id is called as Service Name)
=> Instance id is the unique Id .. For every Instance one unique instance Id will be generated..
=>Providing isntance Id when single instance is there optional .. In that situation it takes ServiceId as the Instance Id of
=> The HostName and port details will be auto detected by Eureka server durigng the process Register/publishment
=> LoadFactor = current Load /Max Load.

Procedure to create Spring Boot project acting as Eureka server

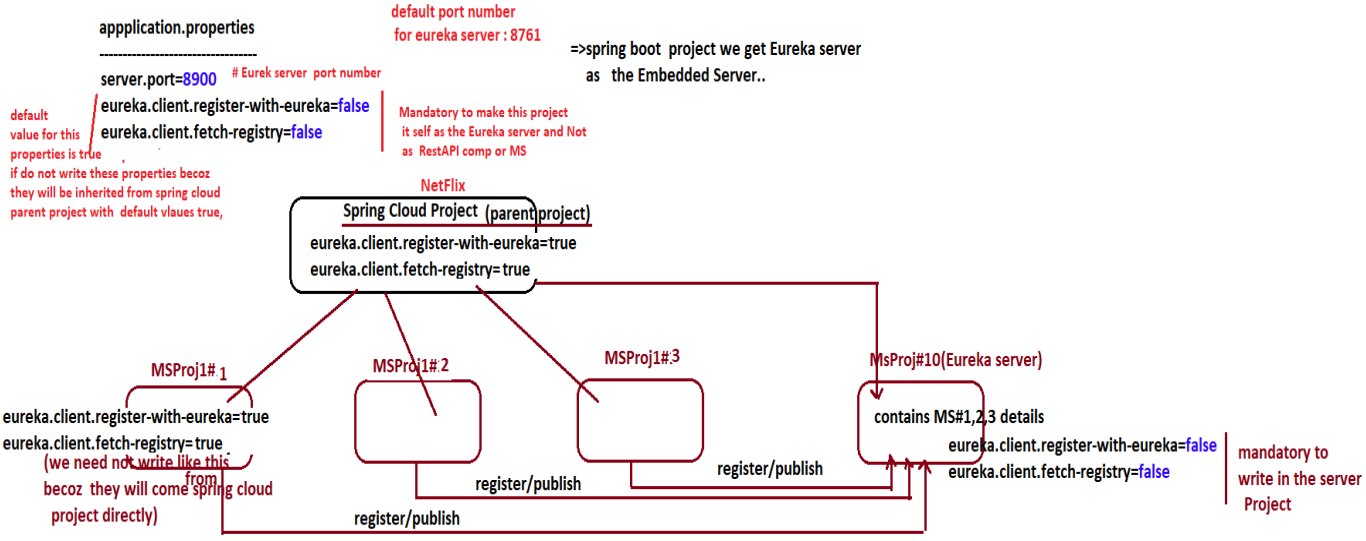
step1) create Spring Boot Project adding Eureka Server as depedencies
starter to select => eureka server (do not add spring web starter by mistake also)



step2) add @EnableEurekaServer on the top of main class

```
//MsProj01EurekaServerApplication.java
package com.nt;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer  //Annotation to activate Eureka Server related configuration EurekaServerAutoConfiguration
public class MsProj01EurekaServerApplication {

	public static void main(String[] args) {
		SpringApplication.run(MsProj01EurekaServerApplication.class, args);
	}

}
```

step3) Add the following entries in application.properties making To disable the process of this project as MicroService and to present this project as R & D Server



=> Every Ms Project should kept ready to register with Eureka Server .. Since we can not register Eureke server Project itself with Eureka server , So we need to make the default value "true" Inherited from the spring cloud project for "eureka.client.register-with-eureka" property as "false" in Eureka server Project.

```
eureka.client.register-with-eureka=false
```

=>Every MS project should be kept ready for discovery/fetching for communincation while registering with Eureka server .. Since Eureka Server Project can not kept ready for fetching/discovery so we need make the default "true" inhreited from the spring cloud project for "eureka.client.fetch-registry" as "false" in Eureka server Project

```
eureka.client.fetch-registry=false
```

The default port number for eureka server is :: 8761

step4) Run the application...

Right click on Project -->run as --> java app/spring boot app

step5) Test the SErver App

http://localhost:8900/

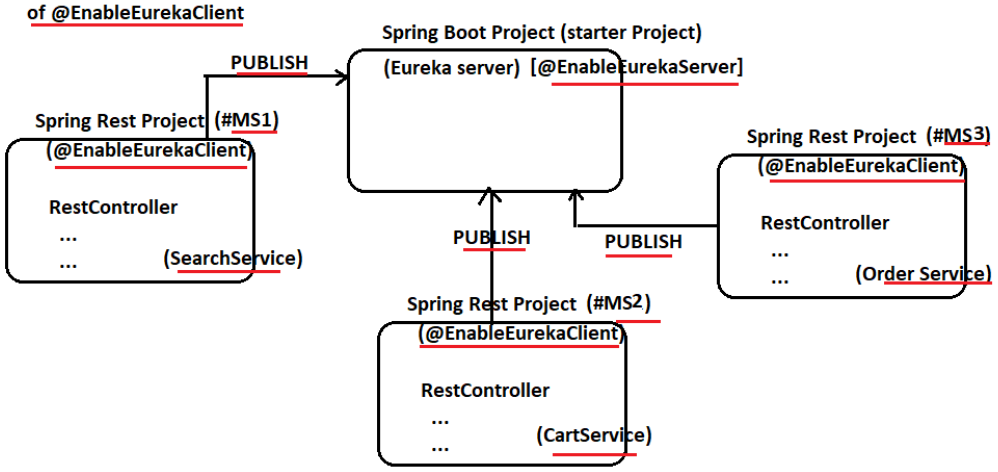
25.NTSPBMS615- July 19th-2022- publishing MSProject to EurekaServer

Publishing MicroService to Eureka server (R & D Server)

==========

=> Every Ms must be published/registered with Eureka Server (R&D Server) by becoming Eureka client

=> we need to develop MicroService as Spring RestController adding the the support of @EnableEurekaClient



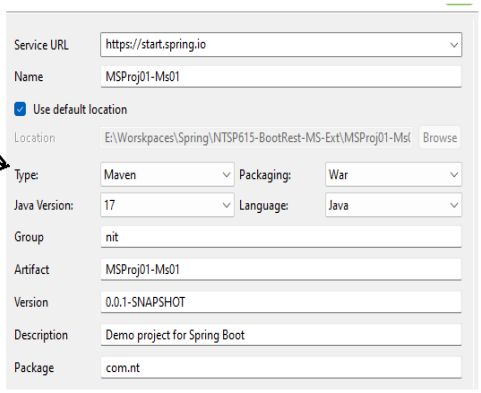
Procedure for MS Development and Publishing / registering with Eureka server

==========

step1) Make sure One Spring boot Project alreay developed and running as Eureka Server (Previous class) *(make sure that it is running the default port : 8761)*

Keep Eureka Project in running mode

step2) Create Spring Boot Starter Project adding spring web, Discovery EurekaClient Dependencies (MicroService Development)



step3) Place @EnableEurekaClient Annotation on top of main class.

```
@SpringBootApplication
@EnableEurekaClient
public class SpringBootMsProj01SearchServiceApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootMsProj01SearchServiceApplication.class, args);
	}

}
```

step4) add the following entries in application.properties file

in applicaiton.properties

```
# MS serivce Port no on Embedded Tomcat server
server.port=7171

# MS  name
spring.application.name=MS-One

# Specify Eureka server url  for  publishing
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

fixed request path in Eureka Server

step5) Develop RestController representing the MicroService

```
//TestServiceController.java "
package com.nt.rest;

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("/TestMs")
public class TestServiceController {

        @GetMapping("/show")
        public  ResponseEntity<String>  display(){

                return  new ResponseEntity<String>("Welcome to Spring Boot MS ", HttpStatus.OK);

        }

}
```

step6) Run the MicroService Project as spring boot App

(This process automatically publishes MS to Eureka server)

step7) Refresh the home page of eureka server (http://localhost:8761) and observe the instance section

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
MS-ONE	n/a (1)	(1)	UP (1) - DESKTOP-QENT2RN:MS-One:7171

General Info

step8) Collect url from status section and modify it to generate the request to MS

http://desktop-qent2rn:7171/TestMs/show

((or)

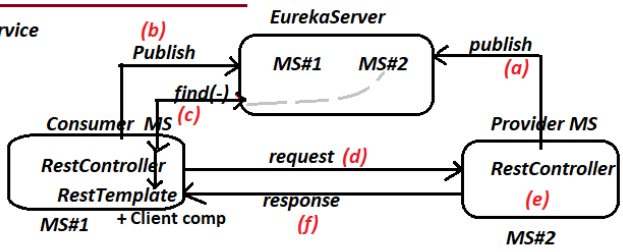
http://localhost:7171/TestMs/show

26.NTSPBMS615- July 20th-2022- Inter Communication of MS

Inter communication between MicroServices (MicroServices Intra communication)

=> To see communication b/w two micro services .. both micro services must be published in the Eureka Server

=> The MS that provides services is called Provider/Server /Producer /Parent Service
=> The MS that cosumes services is called Cosumer/Client /Child Service



CartService <----------> PaymentService
(consumer/Client) (Procudure/Server)

Employee <----------> Department
(consumer/Client) (Procudure/Server)

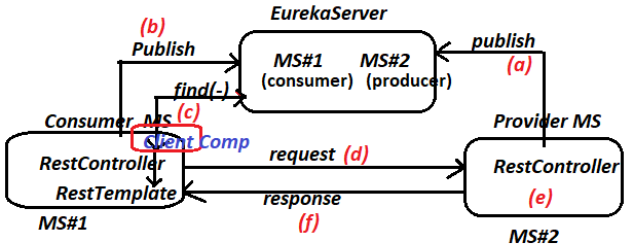
CartService <----------> Billing Service
(consumer/Client) (Procudure/Server)

Generally the the Cosumer and Producer Apps will be represented as show below
(parent and child)

Payment service --------◇ cart service
(parent) (child)

Doctor ——◇—— Patient
(parent) (child)

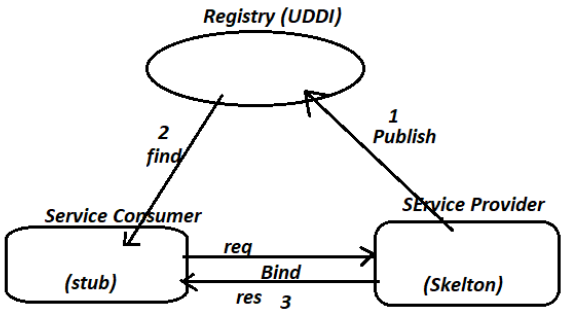
MicroService style intra communication



Here both Consumer and Provider services will be published to Eureka server before starting interaction

=>Here any MS can act as cosumer or producer or both at a time in given context

SOA style Intra communication



Here only provider Service will be published.
=> Here Fixed consumers and fixed publishers will be there

=>In the intra communication of MicroService through Eureka Server the Consumer /Client/ Child MS should find and get the details of Provider/Producer/Server MS by submitting its service Id .. For this the Cosumer MS must use one special comp "Client Comp/ Client Type comp".

=> This Client Comp or ClientType comp also helps to choose one instance of Provider MS among the multiple instance based on Load Factor
=>The work of "Client comp" in Consumer Ms is

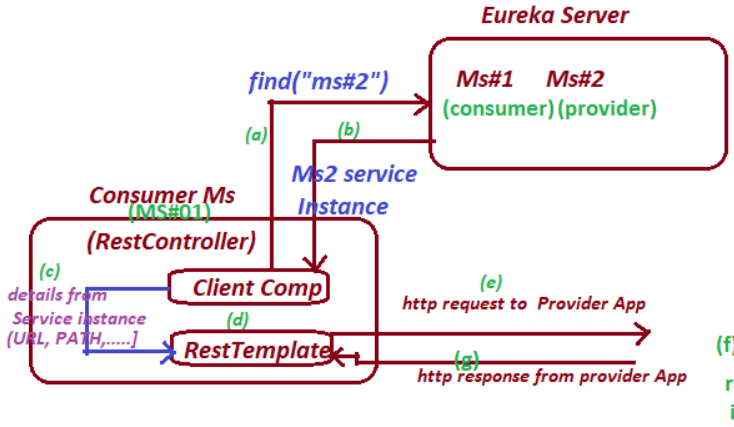
a) getting Producer MS service Instance from Eureka SErver by submitting its Service Id (if needed peforms load balnace)
b) Gathers Producer MS details like URL/URI ,method type, PATH and etc.. from Service Instance
c) Passing the above details to RestTemplate of Consumer MS to make RestTemplate to send http request to Producer Ms and get http response from Producer MS

As of now In Eureka server env.. 3 types of "Client Comps/Client type comps" are possible

a) Discovery Client Comp (Very Basic Client -- Legacy) (No support for LoadBalacing)

b) LoadBalancer Client comp (good) (performs LoadBalacing)

c) Feign Client Comp (Abstract Client -- Good) (generates every thing + Load balacing) (Best)



Most of the times we can inject diffeent Client Comps to Spring beans of Consumer MS through AutoConfiguration

=> We can perform intra communication between micro services by publishing them in the Eureka Server ...

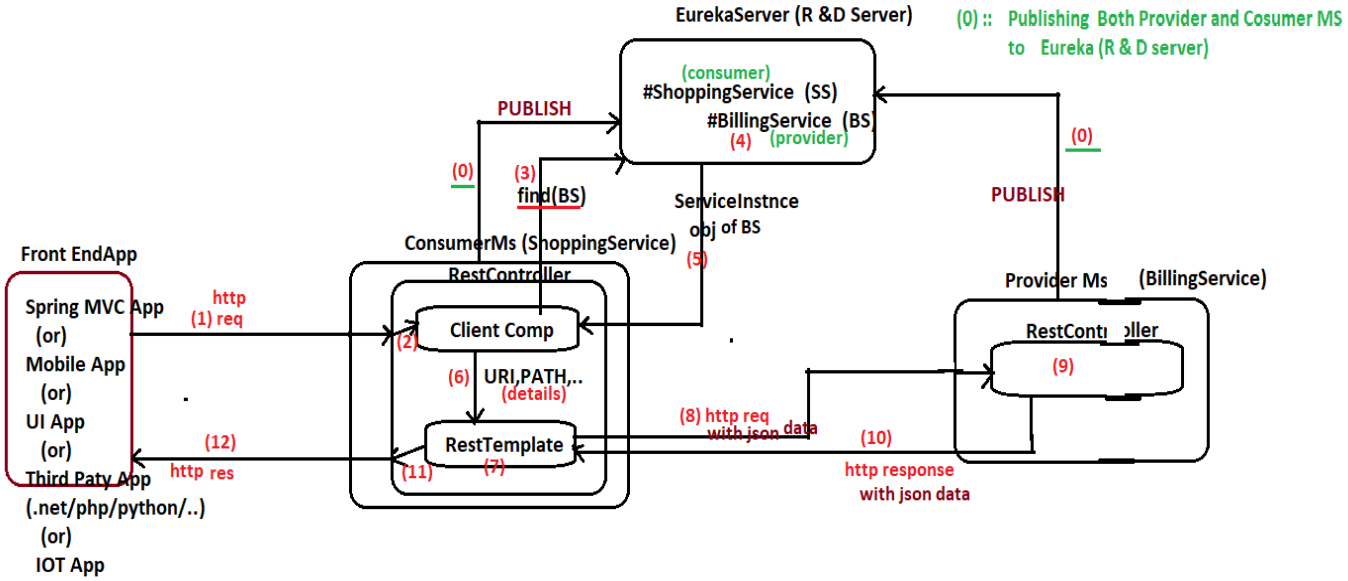
=> The Cusumer Ms must use one or another "Client Comp" or "Client Type comp" to get multiple details about Provider MS by passing service Id .. These multiple details are like URI, PATH, method type,path variables and etc.. will be passed RestTemplate obj to make Http Call to connect with Provider Ms

=> The Consumer MS can work with 3 types of "Client Comps" or "Client Type comps"

> DiscoveryClient (Legacy)

> LoadBalancerClient (for Load balacing) (good)

> FeignClient (Abstract Client) (good)



Example App development (as POC) (Using DiscoveryClient as the Client Comp)

step1) Develop Project as Eureka Server

=>dependencies :: Eureka service of

=> add @EnableEurekaServer on the top main class

=> application.properties

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

step2) Develop Project as Producer MicroService

=>dependencies :: web , EurekaDiscoveryClient

=> add @EnableEurekaclient on the top of main class

=> application.properties

```
---------------------------------
#Ms Properties
#Port number
server.port=9900
#Service id
spring.application.name=Billing-Service
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

=> Develop RestController having producer b.methods

//RestController (provider)

```
//BillingInfoController.java
package com.nt.rest;

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("/billing/api")
public class BillingInfoController {

        @GetMapping("/info")
        public  ResponseEntity<String>  fetchBillingDetails(){
          return new ResponseEntity<String>(" Final BillAmt= BillAmt- discount (Rs.5000)",HttpStatus.OK);
        }

}
```

step3) Develop the Cosumer MicroService (Shopping Service)

=>dependencies :: web , EurekaDiscoveryClient ,

=> add @EnableEurekaclient on the top of main class

=>application.properties

```
-------------------------------
#Ms Properties
#Port number
server.port=6600
#Service id
spring.application.name=Shopping-Service
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

-> Develop spring bean (@Componet) as helper class having logic to use DiscoveryClient as the ClientComp to find and get Producer/provider Ms Instance and Get its details

http://localhost:9900 (URI)
http://localhost:9900/billing/info (URL)

URL=URI +PATH

```
//BillingServiceCosumerClient .java (Client App)
package com.nt.client;

import java.net.URI;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class BillingServiceCosumerClient {
	@Autowired
	private  DiscoveryClient client;

	 public  String  getBillingInfo() {
		// Get Billing-Service  Instance from eureka server
		List<ServiceInstance> listInstances=client.getInstances("Billing-Service");
		// get Single Instance from List of Instnace  (no load balacing)
		 ServiceInstance instance=listInstances.get(0);
		 // get details from Serivce Instance
		 URI uri=instance.getUri();
			//prepare  provider MS related url to cosume method
		String url=uri.toString()+"/billing/api/info";

		//create RestTemplate class obj to cosume the provider  service
		  RestTemplate template=new RestTemplate();
		//  consume the provider service
		  ResponseEntity<String> response=template.getForEntity(url,String.class);
		 // get response content from ResponseEntity object
		  String responseContent=response.getBody();

		  return responseContent;

	 }

}
```

note:: All producer and consumer Ms must be annotated with @EnableEurekaClient

=> Develop RestController in consumer Application by taking the support of above helper class..

//Rest Controller

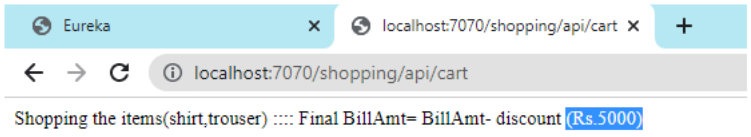
```
//BillingServiceCosumerClient .java (Client App)
package com.nt.client;

import java.net.URI;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class BillingServiceCosumerClient {
	@Autowired
	private  DiscoveryClient client;
```

```
    public  String  getBillingInfo() {
            // Get Billing-Service  Instance from eureka server
            List<ServiceInstance> listInstances=client.getInstances("Billing-Service");
            // get Single Instance from List of Instnace  (no load balacing)
             ServiceInstance instance=listInstances.get(0);
             // get details from Serivce Instance
             URI uri=instance.getUri();
                   //prepare  provider MS related url to cosume method
            String url=uri.toString()+"/billing/api/info";

           //create RestTemplate class obj to cosume the provider  service
             RestTemplate template=new RestTemplate();
           //  consume the provider service
             ResponseEntity<String> response=template.getForEntity(url,String.class);
            // get response content from ResponseEntity object
             String responseContent=response.getBody();

              return responseContent;

    }                                   One MS  RestController is  using another MS RestControlle

}
```

step4) Execute the Projects/Apps in the following order

=>run the Eureka Server
=>Run the Produder App (BillingService) — **8761 -server**
=> Run the Cosumer App (ShoppingService) — **9900 -producer**
=> Go to Eureka Server Console to modify Consumer Service Url — **6600 -consumer**
for testing..

http://localhost:7070/shopping/api/cart



27.NTSPBMS615- July 23rd-2022- Intra MS communication using LBC

Micro Service Intra Communication
==========

(For this we need to use 3 types of clients
a) Discovery Client b) LoadBancerClient c) FeignClient

Limitations of DiscoveryClient type ClientComp
==========

(a) We get list of target MS instances and we need to instance manually .. But actually want chooose we is one instance of target MS(Producer Ms) which having less Load Factor (Load balacing is not possible)

(b) This Baisc Client Comp or Client Type which comp is very much Legacy .. i.e it is not indurstry standard..

(c) Collecting one instance from the list of instance s is pure responsibity the of Cosumer App that to manual process.. So other instancces of target/produce Ms may sit idle..

(d) we need to write all the logics in Client Comp manually

note:: To overcome these problems take the support of LoadBalancerClient type Client Comp.

LoadBalancerClient
==========

Load Factor = current Load/Max Load

=> It is another Client type Comp or Client Comp .. which choose the less load factor instance though they are multiple instances for Target /Procuder Ms i.e we never get List of instances though they are muliple instances for MS.. we always get one instance of MS which is having Less LoadFactor
(Producer MS) (Producer MS)

=>LoadBalnaceClient is an interface and implementation is given by Spring Clould NetFlix people in the form of RibbonLoadBalanceClient class which can be injected to Cosumer App through AutoConfiguration..

=> The method choose(-) with instance Id called on the LoadBalanceClient obj will bring the Less LoadFactor Instance of target /producer Ms.

note:: To create multiple instances for any MS run that Ms App for multiple times with different Port numbers becoz Two Servers or Two Ms of same computer can not take same port number .. but possible across the multiple machines.

Example App using LoadBalanceClient Type Client Comp (MS intra communication using LoadBalanceClient)
==========

step1) Develop Eureka Server App

same as previous App

step2) Develop Producer/Provider/Target MS

(This time provide random number as the instance Id)

=>Since we are planning take multiple instances of producer App by running the Producer App for multiple times it is better to give seperate name for every instance generally it is serviceId:<randomvalue> using

```
eureka.instance.instance-id=${spring.application.name}:${random.value}
```

${spring.application.name} — App name as the ServiceId

${random.value} — System property giving one psuedo random number for every execution

(do not add dev tools)

=> create project having sprigng web , CurekaDisoveryClient, dependencies
=> add @EnableEurekaClient on the top main class
=> add the following entries in application.properties

application.properties

```
#Ms Properties
#Port number
server.port=9900
#Service id
spring.application.name=Billing-Service
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka
# provid  applicaiton + radom vlaue as  Instance Id
eureka.instance.instance-id=${spring.application.name}:${random.value}
```

=> Develop Restcontroller actiging Producer MS

```
package com.nt.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/billing/api")
public class BillingInfoController {
        @Value("${server.port}")
         private  int port;
        @Value("${eureka.instance.instance-id}")
         private  String instanceid;

        @GetMapping("/info")
        public  ResponseEntity<String>  fetchBillingDetails(){
          return new ResponseEntity<String>(" Final BillAmt= BillAmt- discount (Rs.5000) :: using instance::-->"+instanceid+" @port::"+port,HttpStatus.OK);
        }
}
```

p3) Devlop the Consumer App with support of LoadBalanceClient
(do not add dev tools)

=> create project having sprigng web , CurekaDisoveryClien dependencies (Ribbon comes automatically)
=> add @EnableEurekaClient on the top main class
=> add the following entries in application.properties

```
#Ms Properties
#Port number
server.port=6600
#Service id
spring.application.name=Shopping-Service
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

=> develop helper having LoadBalanceClient comp Injection

```
package com.nt.client;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import org.springframework.cloud.client.loadbalancer.LoadBalancerClient;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class BillingServiceConsumerClient {
    @Autowired
    private LoadBalancerClient client;

    public String getBillingInfo() {
        // Get Billing-Service Instance from eureka server based LoadFactor
        ServiceInstance instance=client.choose("Billing-Service");
        // get details from Serivce Instance
        URI uri=instance.getUri();
            //prepare provider MS related url to cosume method
        String url=uri.toString()+"/billing/api/info";

        //create RestTemplate class obj to cosume the provider service
        RestTemplate template=new RestTemplate();
        // consume the provider service
        ResponseEntity<String> response=template.getForEntity(url,String.class);
        // get response content from ResponseEntity object
        String responseContent=response.getBody();
        return responseContent;
    }
}
```

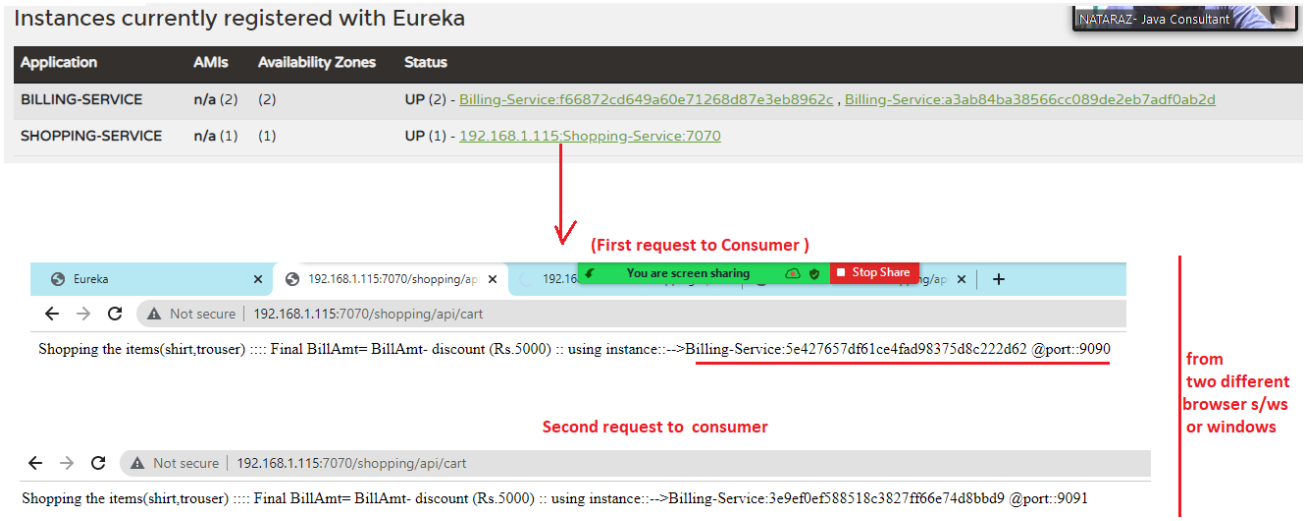
=>Develoe the RestController

```
@RestController
@RequestMapping("/shopping/api")
public class ShoppingSerivceOperationsController {
    @Autowired
    private BillingServiceConsumerClient client;

    @GetMapping("/cart")
    public ResponseEntity<String> doShopping(){
        //use Client Comp
        String resultMsg=client.getBillingInfo();
        try {
            Thread.sleep(20000);
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        return new ResponseEntity<String>("Shopping the items(shirt,trouser) ::::"+resultMsg,HttpStatus.OK);
    }
}
```

ep4) run Apps in the following order

=>run Eureka server App
=> Run Producer Apps multiple times but change port number (server.port value)
in application.properties each time (at least 2 times)
=> Run the Cosumer App

=>Go to Eureka server Consle modify the
Consumer App url



(First request to Consumer)

Second request to consumer

from two different browser s/ws or windows

28.NTSPBMS615- July 24th-2022- Intra MS communication using Feign Client

MicroServices Intra Communication

==========

Limitation with Discovery Client , LoadBalancerClient

=> They can find and get producer MS ServiceInstance and other details from Eureka Server .. But they can not give http calls to interact with target/Producer MS For that we need to use RestTemplate seperately

=> we need to do coding manually to find and get Target MS Service Instance

To overcome this problems use Feign Client as Client Comp/Client Type comp

Feign Client /Open Feign

==========

=> It is called abstract client becoz we just provide interface with method declaration and adding annotation .. But entire logic will be generated in the InMemory dynamic proxy class (Proxy pattern)

=>It is Combination Client i.e it takes care of getting taget/producer Ms service Instance from Eureka server and also takes care of interacting target/Producer Ms by generating http calls that to with out wrting code. (This entire code will be generated in the InMemory proxy class)

=> It is internally LoadBalancer Client i.e it gets Target/Producer Ms service Instance from the List of seerviceInstances which is having Less Load Factor.

=> This mode of Client Comp development improves the productivity of the App. (Faster development)

=> Here we do not develop helper RestConsumer for Consumer RestController rather we develop Interface having @FeignClient("ServiceId") and the generated InMeory DyamicProxy class object will be injected to ConsumerRestController.

=> While worokign with FeignClient we need to add 2 annotation special annotations along with regualar annotations in the Consumer App

a) @EnableFeignClients on the top main class along with @EnableEurekaClient

b) @FeingClient on the top of interface for which dynamic Inmemory proxy class will be generated.internally

```
(b)   must match with taget/producer MS service Id
@FeingClient("Billing-Service")
public interface IBillingServiceRestConsumer{  (a) (any name of interface)
@GetMapping("/billing/api/info")  (e)  // target/producer MS method /operation path
 public ResonseEntity<String>  getBillingInfo();        (complete path = <global path>+ <request path>
        (d)                          (c)
}      singnature              method name need not to match
       should mattch                taret MS method name
       with target Ms
       method
```

Example App Using Feign Client

==========

For this interface the Spring cloud setup generates the Dynamic InMemory Proxy class implementing the getBillingInfo() having logic to use LoadBalancerClient to Leass LoadFactor ProducerMS instance and also use RestTemplate to call Producer MS method using http calls.

step1) Develop Project as Eureka Server

=>dependencies :: Eureka service of

=> add @EnableEurekaServer on the top main class

=> application.properties

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

step2) Develop Project as Producer MicroService

=>dependencies :: web , EurekaDiscoveryClient

=> add @EnableEurekaclient on the top of main class

=> application.properties

```
#Ms Properties
#Port number
server.port=9900
#Service id
spring.application.name=Billing-Service
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka
# Generate Instace id dynaically having service name + random value
eureka.instance.instance-id=${spring.application.name}:${random.value}
```

=> Develop RestController having producer methods

```
package com.nt.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RequestMapping("/billing/api")
public class BillingInfoController {
        @Value("${server.port}")
        private int port;
        @Value("${eureka.instance.instance-id}")
        private String instanceid;

        @GetMapping("/info")
        public ResponseEntity<String> fetchBillingDetails(){
          return new ResponseEntity<String>(" Final BillAmt= BillAmt- discount (Rs.5000) :: using instance::-->"+instanceid+" @port::"+port,
                                                                                                HttpStatus.OK);
        }
    }
}
```

step3) Develop Cosumer MS Project adding spring web , EurekaDiscoveryClient ,open Feign

=> Add @EnableEurekaClient , @enableFeignClients annotations on top of main class

```
@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients
public class SpringBootMsProj04ShoppingSErviceConsumerApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootMsProj04ShoppingSErviceConsumerApplication.class, args);
    }
}
```

=> add the following entries in application.properties

```
#Ms Properties
#Port number
server.port=6600
#Service id
spring.application.name=Shopping-Service
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

=> Take an interface supporting FeignClient code as InMemory Dynamic Proxy class

```
package com.nt.client;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;

@FeignClient("Billing-Service")
public interface IBillingServiceConsumerClient {
        @GetMapping("/billing/api/info")
  public  ResponseEntity<String>  fetchBillingInfo();
}
```

=>Develop the Cosumer RestrController Injecting FeignCleint related Proxy object to consume the target /Producer Ms services.

//RestController

```
package com.nt.rest;
import java.util.Arrays;
import org.springframework.beans.factory.annotation.Autowired;
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;
import com.nt.client.IBillingServiceConsumerClient;
@RestController
@RequestMapping("/shopping/api")
public class ShoppingSerivceOperationsController {
        @Autowired
       private  IBillingServiceConsumerClient client;

        @GetMapping("/cart")
         public  ResponseEntity<String>  doShopping(){
              System.out.println("Proxy class name ::"+client.getClass()+"....."+Arrays.toString(client.getClass().getInterfaces()));
               //use  Client Comp
               String resultMsg=client.fetchBillingInfo().getBody();

               return  new ResponseEntity<String>("Shopping the items(shirt,trouser) ::::"+resultMsg,HttpStatus.OK);

        }//method

}//class
```

step5:

Execute the application

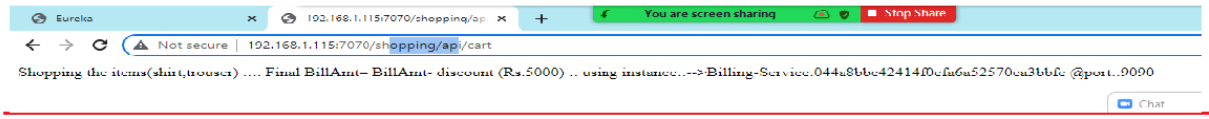
==========

=>Run Eureka server app

=> Run the Producer MS application for 2 or 3 times with different port numbers changed in the application.properties file

=> Run the Cosumer Ms application

=> Go to Eureka Server Home page and modify the Cosumer SErvice url to as shown below



Different pratices to develop Feign Client Interfaces

Producer Ms / Target Ms

Feign Client Interface at Cosumer MS

a) ServiceId /applicaiton -name:: Vendor-Service

```
@RestController
@RequestMapping("/vendor")
public class VendorServiceController{
   @GetMapping("/all")
   public ResponseEntity<List<Product>>
                          getAllProducts(){
      ...
   }
}
```

```
a) @FeginClient("Vendor-Service")
public interface IVendorServiceConsuner{
   @GetMapping("/vendor/all")
   public List<Product> fetchAllProducts();
         (or)

   @GetMapping("/vendor/all")
   public ResponseEntity<List<Product>> fetchAllProducts();

}
```

b) Service Id /app name :: Payment-Service

```
@RestController
@RequstMapping("/payment")
public class PaymentServiceController{

  @PostMapping("/save")
  public String saveCard(@RequestBody CarDetails details){
    ...
    ..
  }

 @DeleteMapping("/delete/{cardNo}")
 public String removeCard
            (@PathVariable Integer cardNo){
   ...
   ..
 }
```

```
@FeignClient("Payment-Service")
public interferface IPaymentServiceClient{

  @PostMapping("/payment/save")
  public String callSaveCard(@RequestBody CardDetails details);
                          ->we are assuming Cosumer MS also
                            getting JSON body along with request
 @DeleteMapping("/payment/delete/{cardNo}")
 public String callRemoveCard(@PathVariable Integer cardNo);

}
```

29.NTSPBMS615- July25th- 27th-2022- Config Server

More on Feign Client
==========

In MicroServices intra communication we need Client Type comps to find the SService Instance Details of Target Ms and to perform Http calls based communucation with taget MS/Producer MS from Consumer Ms.

Client Type comps are
a) Discovery Client
b) LoadBalacerClient (LBC)
c) Fiegn Client (Indurstry standard)

no.of feign interfaces = no.of Producer MS consumed by Consumer MS

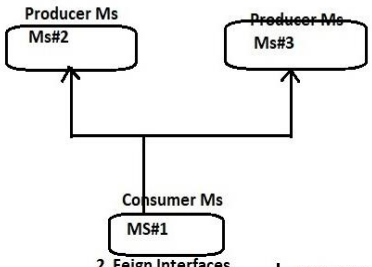
no.of methods in each interface = no.of methods that consumer MS wants to inovke from Producer MS

=> if we have 1 producer - 1 consumer type of MicroServices using Feing client then we need to take 1 Feign Interface (@FeignClient) at Cosumer Ms side... if the the Conumer Ms is consuming the multiple Producer Ms services then we need to take multiple feign Interfaces.

Feign Interface count = no.of producer Ms services consumed by Cosumer Ms

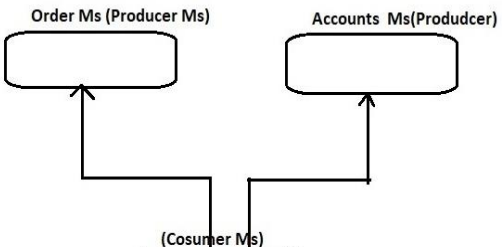
methods count in each feign interface = no.of Producer MS methods that we want to invoke from Consumer Ms

=> For example if MS#1 is consuming Ms#2 and Ms#3 producer services then Ms#1 consumer Ms should have 2 feign Interfaces as shown below



2 Feign Interfaces
=> 1 for Ms#2 service
=> 1 for Ms#3 service

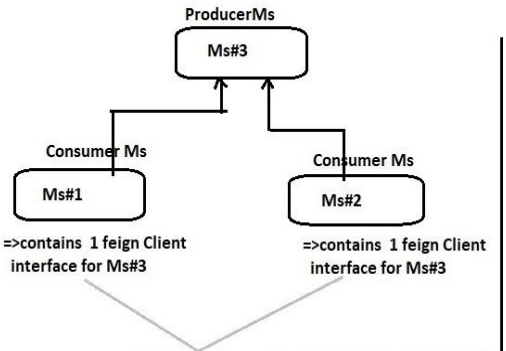
In this Cosumer Ms we need to place 2 @FeignClient("....") annotations on two different feign interfaces having the two different service ids of two different producer Ms. But we need to place @EnableFeignCleints only for 1 time on Main class.



=>Contains 1 Feign Interface for Order Ms
=>Contains 1 Feign Interface for Accounts Ms

if Cosumer MS wants to inovke 5 methods of One Producer MS then the Consumer Ms should declare 5 methods in the Feign Inteface Cosumer MS

=> if one Producer Ms services are used by multiple consumer Ms then we may need to place same feign Client Interface in multiple consumer Ms.

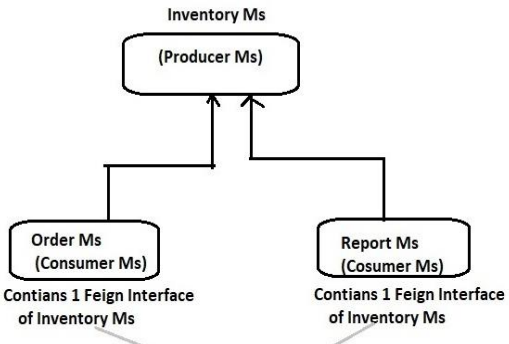


=>contains 1 feign Client interface for Ms#3

=>contains 1 feign Client interface for Ms#3

(Both may be same interfaes or different interfaces)

note:: if both MS#1 and MS#2 Cosumer MSs wants to invoke same methods of Producer MS then we need to place same method declaration in Feign Interface of Consumer MS otherwise we need to place different methods in Feign interface



Contians 1 Feign Interface of Inventory Ms

Contians 1 Feign Interface of Inventory Ms

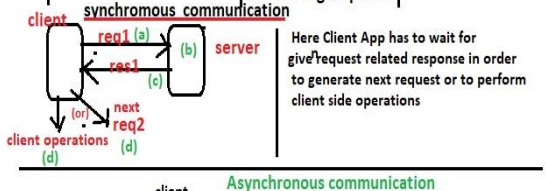
Both can be same or different

Summary on Clien Type comps

Client Type comp	support for Load Balancing	Required Annotation	Is Abstract Client or Concrete Client	industry standard or not	required dependnecy	operations
DiscoveryClient	no	@EnableEurekaClient (main class)	concrete client	not	Discovery Cleint Starter	=>call getInstances(-) to get SService Instance and use RestTemplate to make http calls (instance should be picked manually here)
Load Balancer Client	yes	@EnableEurekaClient (main class)	concrete client	not	Discovery Client Starter	=>same as above but method is choose(-) Here instance will be picked up dynamically
FeignCleint	yes	@EnableEurekaClient, @EnableFeignClents (main class) @FeignClient(---) (interface level)	abstract client (Intenrally InMemory proxy will be generated)	yes	Open Feign	=>The Inmemory Proxy class for @FeignClient interface will take of getting service Instance and making http calls

What is webClient and when should is use it?

Ans) Spring webflux module is given to develop reactive application (mainly supporting asynchronous communication and non-bloking programming from Clients).. To invoke spring webflux module based Reactive apps from client apps we need webclient object.,



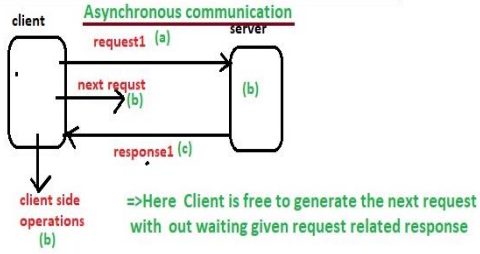
Spring Cloud Config Server

(Allows to keep common properites in Global place) and access

=> It is also called Configuration server(CS) and this is useful to get common key=value pairs required for the multiple micro services from the common place.. i.e instead of placing same key=value pairs in multiple micro services .. we can place them in common place and we can get them through configuration server for multiple micro services

=> These common key=value pairs are generally DB conneection properties , email properties , security properties and etc.. which are required as same properties in mutltiple in MicroServices..

=> We place these common key=value pairs seperate properties file outside of all MicroServices Projects and we take seperate project for Configuiration Server having one application.properties and this file will be linked with that common/seperate properties file.



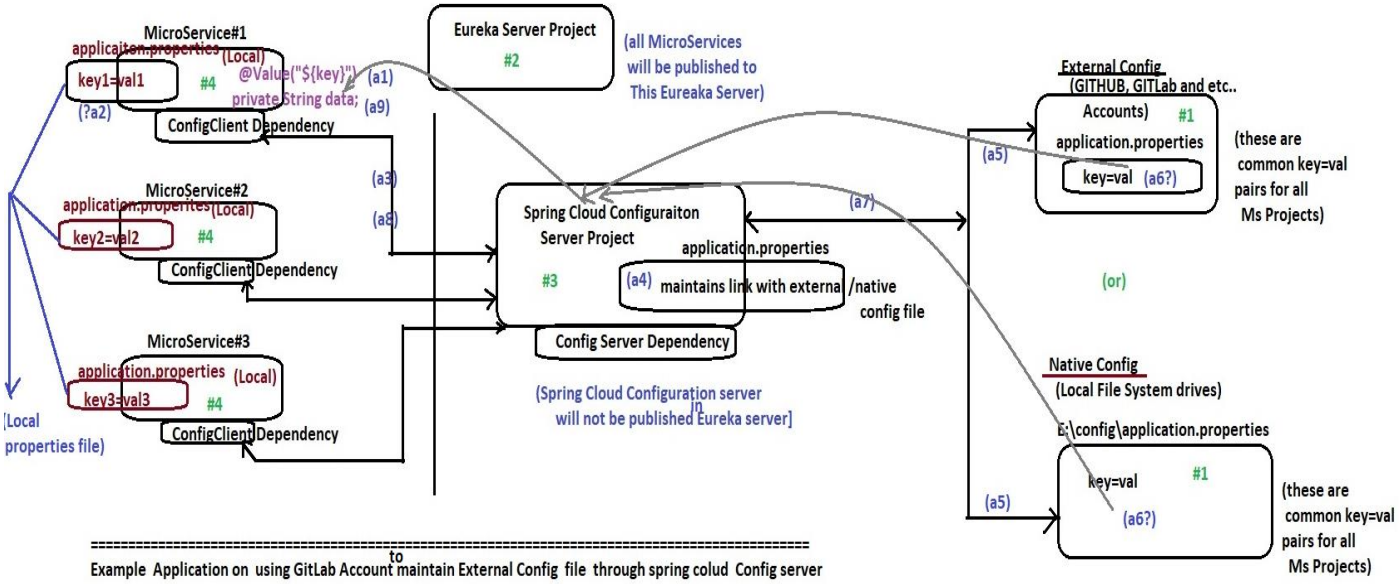
=> WE need to create Configuration server Project(Also a Ms Project) adding "ConfigurationServer" dependendency and this Configuration server by default runs on the Port number 8888.

=> We can make Configuration server Project getting common key=value pairs in two ways

(best) a) Using External Server Configuration (i.e we can place common properties file in GIT accounts like github, gitlab(best), bitbucket,..)
(Good for all env.. dev, test, uat, prod)

b) Using Native Server Configuration (i.e we can place common properties file in Local File System Drives like E:,D: drives and etc..
(Generally used in dev env.. bit)

=> The real micro service projects that want to use Configuration server managed common properties file content (key=value pairs) must be added with "Configuration Client " dependency.



Example Application on using GitLab Account to maintain External Config file through spring colud Config server

step1) create application.proeprties (External config file) in Git Lab account

-> go gitlab.com
-> register/singup account , verify through email address

-> singin/Login to git lab account by submitting username,password
that were created above

-> create new Project giving Project name

Menu bar ---> Projects ---> create new Project ----> Blank Project-->
project name :: CsProj1 --->select public ---> create project.

->add applicaiton.properties file in that Project

Go to home page CsProj1 ---> + ---> new file ---> application.properties

add --> key=value pairs

application.properties
---
dbuser=system
dbpwd=manager

commit .. chanages

-> Gather GitLab account Project url

Go to CsProj1 home page ---> clone ---> gather url ::

https://gitlab.com/nareshit_ameerpet/csproj3.git

protocol domain gitlab username project name

step2) create Eureka server Project in Eclipse IDE

(add :: Eureka server as dependency)

--> place @EnableEurekaServer on the top of main class
--> add the following entries in application.properties

```
# server port
server.port=8761
#disable registration and fetching
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

step3) create Spring Cloud Configuration server Project (ConfingSeever Project)

(add Config server Dependency)
(select from spring cloud config section)

->add @EnableConfigServer on main class
-> add the following entries in application.properties file

```
# server port
server.port=8888

# Link to GITLib acaccount for extenal config file
spring.cloud.config.server.git.uri=https://gitlab.com/nareshit_ameerpet/csproj3.git
```

Link Url

We can use GITHUB , BitBucket also
External Configuration for Configuration server using the same process.
note:: while working with GIT hub
there is no need of passing .git
in the link url of application.properties

step4) create Mutliple MicroService Projects having the following dependencies
Eureka
a) spring web b) Discovery Client , c) Config Client (new)
(select from spring Cloud Config section)

-> add @EnableEurekaClient on the main class
-> add the following entries in application.properites file

```
# server port (MS Port)
server.port=9900
# service name or applicaiton name
spring.application.name=EMP-SERVICE
#provide Eureka server Url to register EUreka server
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```

spring.config.import=optional:configserver: (if config server is in default 8888 port)
spring.config.import=optional:configserver:http://localhost:8899 (if the config server is in other port number)

Develope RestController (MS) reading content of extenal config file (GibLab account application.properties file)

//controller class

```
package com.nt.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping("/employee")
public class EmployeeOperationsController {
	@Value("${dbuser}")
	private String dbusername;
	@Value("${dbpwd}")
	private String dbpassword;

	@GetMapping("/show")
	public ResponseEntity<String> showDBDetails(){
		return new ResponseEntity<String>("Emp -->DB Details"+dbusername+"...."+dbpassword, HttpStatus.OK);
	}

}
```

#Ms2 *application.properties*

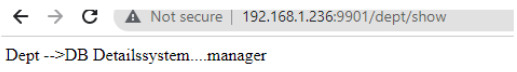
```
# server port (MS Port)
server.port=9901
# service name or applicaiton name
spring.application.name=DEPT-SERVICE
#provide Eureka server Url to register EUreka server
eureka.client.service-url.default-zone=http://localhost:8761/eureka
# To make Ms Connecting to 8888 port number ConfigurationSErver (required from spring boot 2.4 onwords)
spring.config.import=optional:configserver:
```

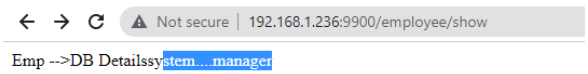
-> add @EnableEurekaClient on the main class

```
//DeptOperationsController.java
package com.nt.rest;

import org.springframework.beans.factory.annotation.Value;
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("/dept")
public class DeptOperationsController {
	@Value("${dbuser}")
	private  String  dbusername;
	@Value("${dbpwd}")
	private  String   dbpassword;

	@GetMapping("/show")
	public  ResponseEntity<String> showDBDetails(){
		return  new ResponseEntity<String>("Dept -->DB Details"+dbusername+"...."+dbpassword, HttpStatus.OK);
	}

}
```

step5) ***Run the Applications in the following order***

->Run Eureka server Project
-> Run Config SErver Project
-> Run All Ms Projects
-> Go to EurekaSever Home page and modify the URL both
Emp, Dept services





Making Multiple MicroSErvices getting common data from Native Config file (Local system drivers)
With the support spring Cloud Configuration server

=> It is suitable only in Dev , Test env.. but not in UAT , production env..
=>Use this in dev, test env.. if u r not ready with GIT Accounts
=> Generally we place this Native Configuration related application.properties file
in the spring Cloud Configuration project itself (which also uses Local system Drives)

Example App
==========

step1) Develop Eureka Server App

(same as previous App)

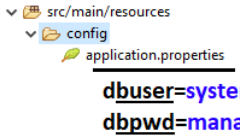
step2) Develop Confiauration SErver

(add Config server Dependency)

(select from spring cloud config section)

->add @EnableConfigServer on main class

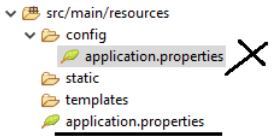
-> add new applicaiton.properties by creating a folder like "config" in src/main/resources folder to keep common key=value pairs required for all Ms projects.




```
dbuser=system
dbpwd=manager
```

src/main/java , src/main/resources folders of maven project will be placed in classpath by default.

->add the following entries in application.properties file of src/main/resources folder




```
# server port
server.port=8888
#active native profile of its Parent Project
spring.profiles.active=native
# specify "config" folder of classpath as NativeConfig Location for Configuration Server
spring.cloud.config.server.native.search-locations=classpath:/config
```

Native Cofig Location

The Parent project of Configuration server is having profiles .. the default profile is designed to get Linked with Git Accounts i.e taking Extenal Configuration where as "native" profile is designed to get linked with Native Config (local drives) So we are activating native profile.

step3) Develop Multiple MicroServices

(same as previous)

step4) Execute the Applications/Projects in the following order.

->Run Eureka server Project

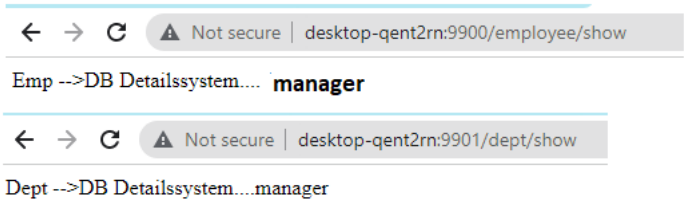
-> Run Config SErver Project

-> Run All Ms Projects

-> Go to EurekaSever Home page and modify the URL both Emp, Cust Ms services

http://desktop-iudaavl:9901/cust/display ---> To generate request Cust Ms

http://desktop-iudaavl:9900/emp/show ---> To generate request Employee Ms

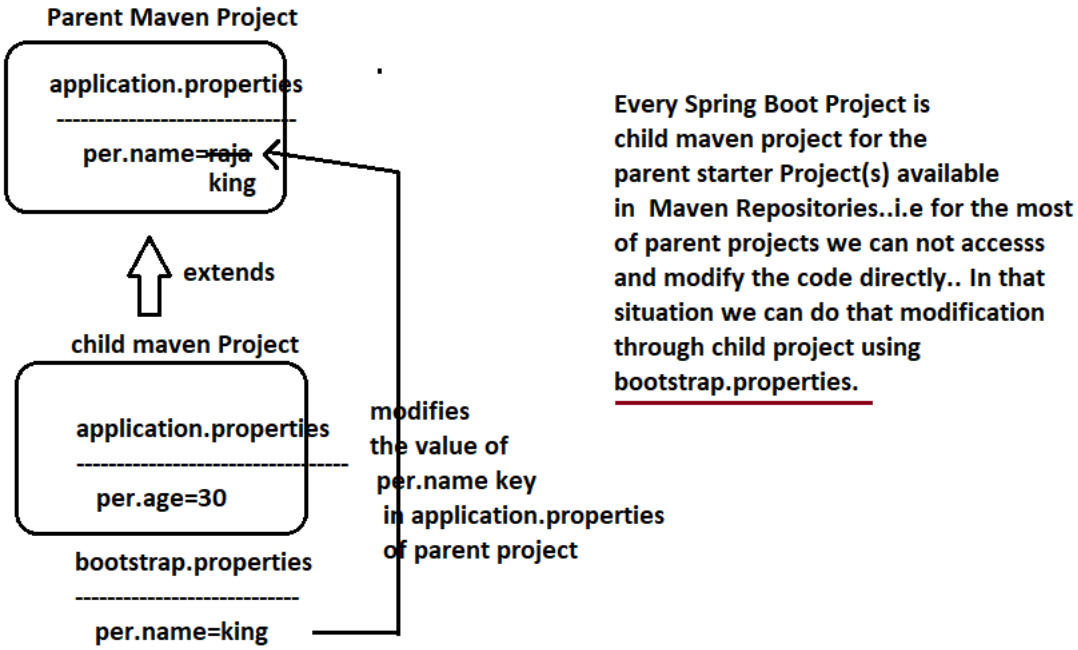


To get dynamic modifications to MS Projects reflecting 'the modifications done in confing/application.properties file we need use @RequestScope + spring boot actuators

bootstarp.properties

if u want change the value of certain key belonging to parent project being from child project then the child project can use bootstrap.properties placed in the classpath like in src/main/resoruces folder.

=>Being from Child Maven Project.. if u r looking to change the parent Project's application.properties file values then use bootstrap.properties in the child project.



Q) What is default port number of spring Clould ConfigServer
ans) 8888

Q) What is spring Clould ConfigServer url/uri
ans) http://localhost:8888

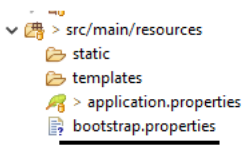
Q) Can we change the Port the number of Config Server?
Ans) yes .. use the application.properties file of Config server project
server.port=8811

accordingly config server uri will change http://localhost:8811

Q) How can we connect to ConfigServer from Ms App if the Config Server is not running on the default port number?
ans) a) using seperate bootstrap.properites (best) b) using spring.config.import of application.properties) (does not work in old versions)
Using bootstrap.properties

step1) make sure Config SErver Port number is changed (server.port=8811)

step2) add bootstrap.properties in src/main/resources folder of Ms Project having config server uri as shown below.



spring.cloud.config.uri=http://localhost:8811

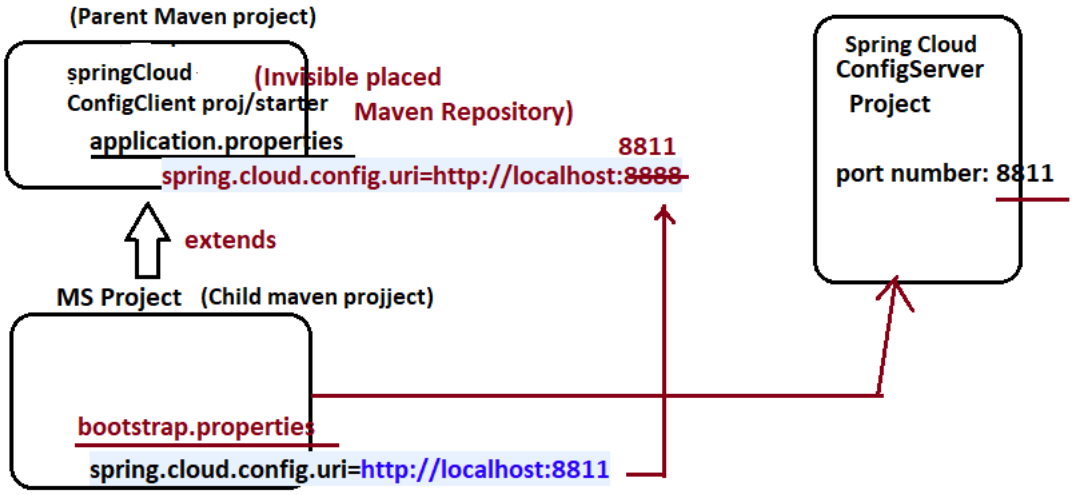
step3) make sure that spring-cloud-starter-bootstrap dependency jar file is added to build path. (In Ms Projects)

```
<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-bootstrap -->
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-bootstrap</artifactId>
</dependency>
```

(collect from mvnrepository.com)

step4) Run the PRojects

=> Run EurekaServer
=> Run ConfigServer
=> Run Ms Project
=> Test Ms Project
http://localhost:9900/employee/show



=> While executing any Ms Project the bootstrap.proeprties will execute first and modifies parent project settings

note: Every Starter added to spring boot project ..makes the starter as the parent project and our spring boot project as the child project.. Being from sprong boot project if u want to modify the added starter project's application.properties file content then use bootstrap.properties

---

Q) How can we connect to ConfigServer from Ms App if the Config Server is not running on the default port number? (How can we do this work with out bootstrap.properties)

Ans) Possible by adding following entries in the application.properties file of MS Project reflecting the Changed port number /uri of Config Server.

application.properties

```
# To make Ms Connecting to certain port number ConfigurationSErver (required from spring boot 2.4 onwords)
spring.config.import=optional:configserver:http://localhost:8811
```

note:: Here no need of adding bootstarp.properties and adding spring-cloud-starter-bootstrap dependency.

Q) In Ms Project if we specify two different uris with two different port numbers of Config Server using both application.properties and bootstrap.properties then which will be taken?

Ans) The url/uri of Cloud Config server specified in the bootstrap.properties of MS Project will override the uri/url of config server specified in the application.properties file of MS Project

Assingment : How connect to Eureka server from MS Project if its port number is changed?

31.NTSPBMS615- July 29th-2022- @RefreshScope-Spring Boot actuators

==========

Working with RefreshScope using @RefreshScope Annotation

==========

done in

=>The modifications GitLab /GITHUB Extenal Config file content will reflect to all the MicroServcies
we
olnly when restart the Config serrver and all Ms .. But Restrating Configserver every time for each modification
and all Ms
done in External Config is a not recomanded process.

=>To overcome that problem we can use RefreshScope (@RefreshScope) with support of spring boot Actuators (readymade endpoints /ready made Microservices given by spring boot)

Procedure to work @RefreshScope

==========

step1) spring boot actuator dependency to our MS Project (EmpRestSErvice kind of Project)
(Config Client)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

right click on project ---> properties
---> spring ---> add starter ---> search for
actuator and select spring boot actuator.

step2) Activate the readymade actuator "refresh" in our MS Project (Config Client)
adding the entries in application.properties

In application.properties

```
# Activate all actuators (*) or only refresh actuator
management.endpoints.web.exposure.include=*
```

(additional entry)

step3) place @RefreshScope on the top RestController of In Ms Project

```
@RestController
@RequestMapping("/emp")
@RefreshScope
public class EmployeeOperationsController {
    ...
    ....
    ....
}
```

Convenience annotation to put a `@Bean` definition in `refresh scope`. Beans annotated this way can be refreshed at runtime and any components that are using them will get a new instance on the next method call, fully initialized and injected with all dependencies

step4) start Eureka Server ,Config Server , MsProject in regularfashion

step5) Test the Ms using the regular url
http://192.168.1.236:9900/employee/show
http://192.168.1.236:9901/dept/show

shows the original values

step6) Modify entries in application.properties file of GitLab account (External Config file)
open properties file in git lib --> edit web editor ----> ....

step7) Test the MS using regular url

http://192.168.1.236:9900/employee/show
http://192.168.1.236:9901/dept/show

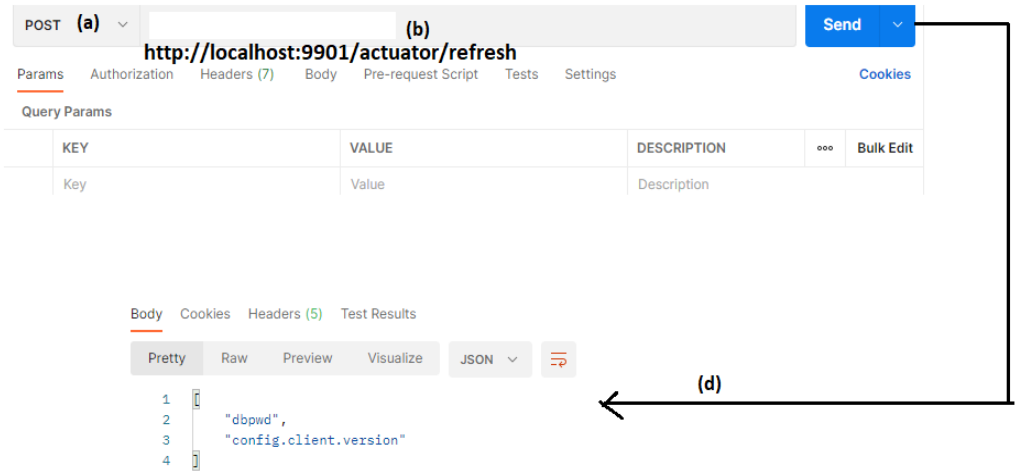
still shows old values

(Modifications does not reflect suprisingly)

step8) Gather EndPoint details of "refresh" spring boot actuator and give POST mode request to it using POSTMAN tool. (becoz giving POST mode request is not possible from browser address bar)

"refresh" actuactor url is http://localhost:9900/actuator/refresh
http://localhost:9901/actuator/refresh

http://localhost:9900/actuator/refresh (c)

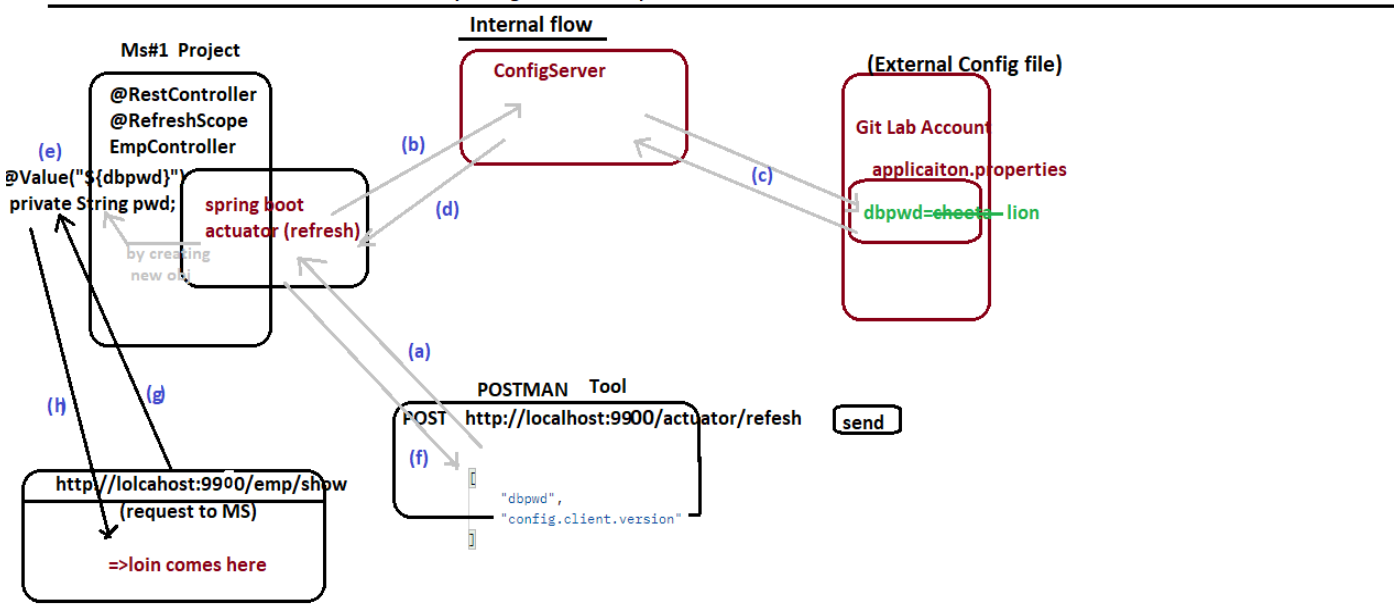


step9) Test the MS using regular url

http://192.168.1.236:9900/employee/show
http://192.168.1.236:9901/dept/show

The modifications will come here

(chanages will reflect)



=> @RefreshScope + refresh actuator based services can also be used to reflect the changes done in native config to MS Projects with @RefreshScope annotations

32.NTSPBMS615- July 31st july - spring boot actuators

What is the benifit of spring boot actuators ?

=>There are non-functional features that required in any project's deployment and maintainance .. Earlier teams used independent various tools given by jdk and thirdparty to get these non-functional features..

=>For example when bug is raised we need logging info and threaddump info
Earlier people have used seperate logging tools like log4j , slf4j and etc..
also seperate thread dump tools like jconsole , JMC , jstack ,Visual Vm and etc..
After the arrival of spring boot .. we can use actuators support like logging , threaddumps for same.

To get all actuators

http://localhost:9901/actuator

List of imp actuators

health ,info , configprops , mappings, threaddump, heapdump, env, beans, refresh , loggers , scheduledtasks,cache ,metrics

beans :: Gives info about all spring beans of current spring boot App (Both AutoConfigured and manually configured)

URL :: http://localhost:9901/actuator/beans

How to add Json Formatter Extension to chrome?

=> https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en
(open this url chrome browser ---> add to chrome ---> install ...

configprops :: Gives all the @ConfigurationProperties based key=value loadings done in both user-defined and pre-fined spring bean classes. note: The user-defined spring beans should setters and getters to get the fields info

http://localhost:9901/actuator/configprops

env :: Gives the current env.. of the Project in the form of key=value pairs like jars files added classpath , cpu details , os details and etc..

http://localhost:9901/actuator/env

loggers :: Gives all log messages related details along withe their logging levels.

http://localhost:9901/actuator/loggers (Gives various classes of the current spring boot project and their' enable logger levels)

scheduledtasks : Gives all @Scheduled methods info like initialDelay , fixedDelay,fixedRated, cron and etc.. details

threaddump :: Gives info about all the daemon threads(background threads) that are created and running continusoly

http://localhost:9901/actuator/threaddump

mappings :: gives info all handler methods , RestController methods mappings info
========== like request path , method name, method signature , method type and etc..

http://localhost:9901/actuator/mappings

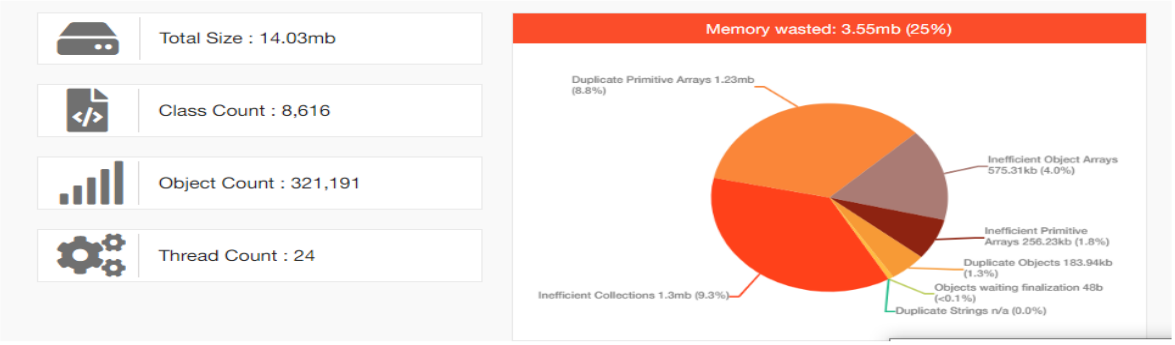
heapdump :: Gives gives heapdump (memory details) in the form downloade byte code info..
So we need addtional tools to analyze that content like Eclipse Memeory Analyzer ,
IBMheapMemoryAnalyzer , heap hero and etc... (MAT, DynaTarace, App Dynamics , Datadog and etc..)

http://localhost:9901/actuator/heapdump

↓

gives "heapdump" file in downloads folder havig byte /machine code

To analyze the heap dump
---
go to https://heaphero.io/ -----> select the above heapdump file --->
get the analyzation report



caches :; gives info all CacheManagers that are configured in different parts of the application.

======

http://localhost:9901/actuator/caches

metrics :: gives Info about properties list related to current project.

http://localhost:9901/actuator/metrics

What is the base path for actuators?

http://localhost:9901/actuator

Can we change the basepath of actuator?

=>Like changing context path of web application.. we can also change basepath or context path of actuators

in application.properties

```
# To change basepath of actuctors
management.endpoints.web.base-path=/nitinfo
```

To test the change

---

http://localhost:9901/nitinfo
http://localhost:9901/nitinfo/health
http://localhost:9901/nitinfo/beans

note:: we can not take "/" as the basepath for actuators becoz it is already assigned to DispatcherServlet.

Q) While working with "refresh" actuator if Extenal Config file (GitLab account file) and local application.proeprties file of the MS project contains same keys with different values Can u tell me which value will be injected.

GitLab's application.properties

---

dbuser=system

local application.properties (not the native config file)

---

dbuser=ramesh

```
@Value("${dbsuer}")
private String user;  --- gets what value :: .....system.....
```

(First reads from local application.properties and that value will be overrident with External Config (GitLab) value.

33.NTSPBMS615- aug 2nd 2022 - Circuit breker

Q) While working with "refresh" actuator if Extenal Config file (GitLab account file) and local application.proeprties file of the MS project contains same keys with different values Can u tell me which value will be injected as the final value ?

GitLab's application.properties

---

dbuser=system

local application.properties

---

dbuser=ramesh

In Ms App

@Value("${dbsuer}")

private String user; --- gets what value :: ..... *system*

(takes from External ConfigServer)

Circut Breaker in spring Boot MicroSErvices

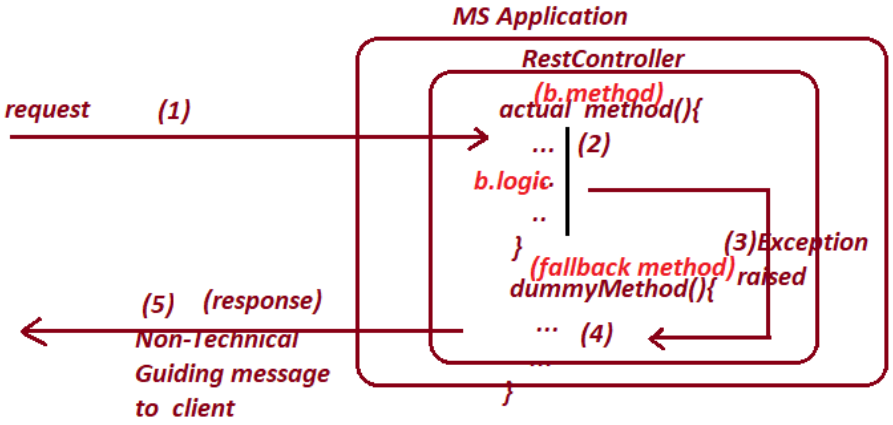
==========

=> if the method b.method of Application is continuosly throwing exception then avoid execution of actual b.method logic and provide dummy response or non-technical guiding response to Client is called working with fallback method with circut breaker.

What is fallback method?

=> The method that executes automatically when the exception raised in the actual b.method to provide non-technical guiding messages to enduser is called fallback method.

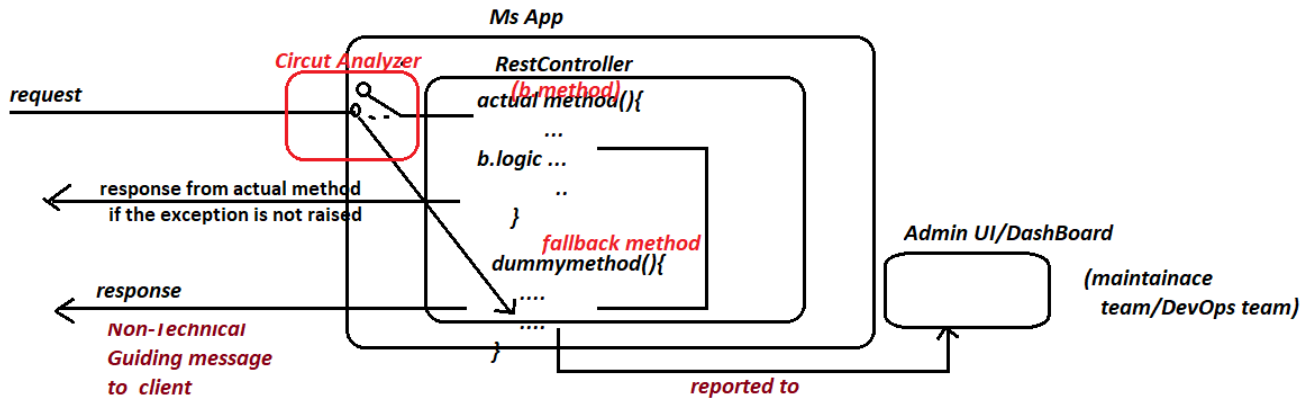
The non-technical guiding messages are " Service not found " , "please try after some time" "Inconvience regratted" , " service down due technical issues" , " site on maitainace" and etc..



What is fallack method with Circuit breaker support?

if the actul b.method of the application is continuosly throwing exception.. then avoid sending request b.method for some amount of time and make DispatcherSErvlet kind of FrontController redirecting request of actual b.method to dummy fallback method again for certain amount of time is called fallback method with circut breaker.

=>let us assume the actual b.method has thrown exception for 20 times already and the fallack method is executed .. if circut breaker concept is added the next few requests given actual method directly goes to dummy method for certain amount of time . this is called fallback method with circuit breaker.



case1:: actual method has got 20 requests and all of thrown exception so fallback methid is executed (only fallack method)

case2: after 20 times exeception raising continuosly(back to back) in actual method, the next requests given to actual method goes to fallaback directly to deliver non-technical guidling messages for some amount time .. after that case1 repeates (fallback with circuit breaker)

Two types of Circuits on Circuit Breaker with Fallback method mechansim

(a) Open Circuit :: It indicates that the b.method had continuosly thrown the exception (back to back) So circut to b.method is broken (open) for client requests.. So client request will redirect to dummy fallback method as shown in the diagram for certain amount of time

(b) Closed Circuit :: It indicates the request given by client goes to b.method directly

Realtime use-cases for Circuit breaker impl ::

a) card payment b) Online flash sale c) online exam results d) online ticket booking and etc..

e) shutdowning the Db s/w for maintainance

=>To implement Circut breaker concept in Spring Boot Ms we need to use spring Boot hystrix implementation given by nextfilx For this we need two annotations

a)@EnableHystrix on the top of Main class

b)@HystrixCommand on the top of actual b.method specifying the dummy method name

Example App on fallback method

add manually

step1) create spring boot Project adding web , hystrix starters.

(Choose spring boot version as <=2.3)

note : Latest versions of spring boot does not support netflix hystrix .. so use resilliance4j as the alternate

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-hystrix -->
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
  <version>2.2.10.RELEASE</version>
</dependency>
```

add manually by collecting from mvnrepository.com

step2) add @EnableHystrix on the main class

step3) Develop RestController as shown below having b.method and dummy method

```
@RestController
@RequestMapping("/ticket")
public class TicketBookingRestController {

    @GetMapping("/book")
    @HystrixCommand(fallbackMethod = "dummyBookTicket")
    public String bookTicket() {
        System.out.println("TicketBookingRestController.bookTicket()");
        if(new Random().nextInt(10)<5)
            throw new RuntimeException("Problem is b.logic");

        return "output from b.logic";

    }

    public String dummyBookTicket() {
        System.out.println("TicketBookingRestController.dummyBookTicket()");
        return "Place Try later -- Inconvience is regratted";
    }
}
```

(This is just fallback for actual b.method i.e we have still not circuitbreaker parameters for the fallback method.

=> Circuit breaker concept can be implemeted directly using Rest API or using MircorSerivce

MicroService = Rest API + MS designing priciples like registring with Eureka server, Circuit breaker, api gateway, config server and etc

This dummy fallback method must be there is same class and *that to not having any parameters.*

=>The return type of actual b.method and dummy mehtod (fallback method) need not to match
=> fallback methods are callback methoids i,e we do not call them manually ..that will be called automatically when needed

step3) Run the App and Test the Application.

http://localhost:9901/ticket/booking

application.properties

```
# MS port number
server.port=9901

# MS name
spring.application.name=TICKET-SERVICE
```

34.NTSPBMS615- aug 3rd 2022 - Circuit breaker

=>Spring Boot circuit breaker is the concept and implementations netflix hystrix and resilience4j

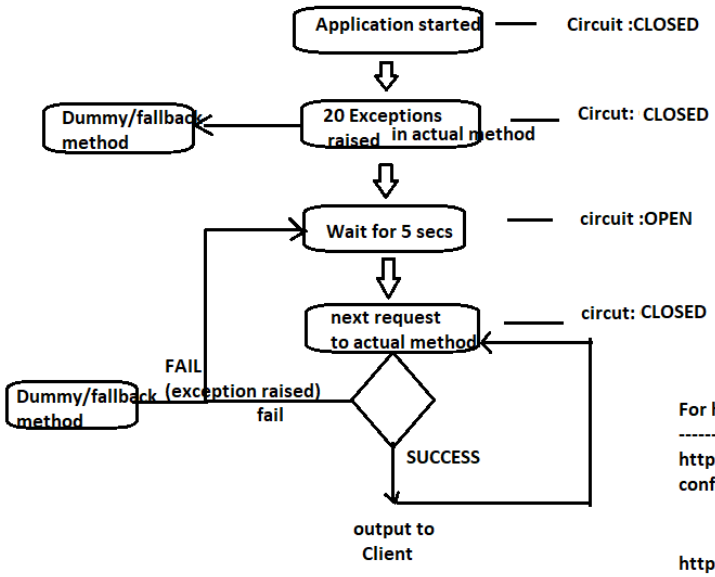
=> with respect to hystrix implementation

=>When the Applicaiton is started the Circuit will remain closed i.e intially request goes to atual method

=>The default exception count is : 20 (After raising exception for 20 times by actual method the curcuit will open)

=> The default retry-time is :: 5 sec (after getting 20 exceptions the circuit will open next request go for 5sec i.e request will to fallback method directly in these 5 secs and circuit will be closed after this 5 sec)

=> Once Circuit is ReClosed .. the next goes to actual method .. if exception is raised the circuit will be again opened for 5 secs .. and so on...



(retry-time)
During this 5 seconds if any request given to autual mehtod ..does not go actual mehtod.. it directly goes to dummy/fallback method.

For hystrik command properties
---
https://www.logicbig.com/tutorials/spring-framework/spring-cloud/hystrix-configuration-properties.html

(or)
https://github.com/Netflix/Hystrix/wiki/Configuration#CommandCircuitBreaker

Example1 (fallback .nethod with Circuit Breaker)
==========

step1) create spring boot project taking version as 2.3.6.RELEASE and adding web , netflix hystrix dependencies

step2) add @EnableHystrix on the main class

step3) Develop RestController having @HystrixCommand to specify dummy fallback and to enable circuit breaker

```
package com.nt.controller;

import java.util.Random;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```

realtime usecases::

=> overload issues in request amazon flash sale and tatkal ticket booking in IRCTC website.

```
import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;
import com.netflix.hystrix.contrib.javanica.annotation.HystrixProperty;

@RestController
@RequestMapping("/ticket")
public class TicketBookingRestController {
	int count=0;

	@GetMapping("/book")
	@HystrixCommand(fallbackMethod = "dummyBookTicket",
	    commandProperties = {
				@HystrixProperty(name="circuitBreaker.enabled", value="true")
			})
	public  String   bookTicket() {
		System.out.println("TicketBookingRestController.bookTicket()");
		 if(new Random().nextInt(10)<10)
			throw new RuntimeException("Problem is  b.logic");
		 return "output from  b.logic";
	}
	public  String dummyBookTicket() {
		count++;
		System.out.println("TicketBookingRestController.dummyBookTicket()::"+count);
		return "Place Try later -- Inconvience is regratted";
	}
}
```

enables the circuit breaker

0

step4) give 20 to 3 contionuos requests to RestController from brrowser observe server consoe

http://localhost:9901/ticket/book

```
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::1
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::2
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::3
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::4
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::5
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::6
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::7
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::8
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::9
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::10
TicketBookingRestController.bookTicket()

TicketBookingRestController.dummyBookTicket()::11
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::12
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::13
TicketBookingRestController.bookTicket()
```

```
TicketBookingRestController.dummyBookTicket()::14
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::15
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::16
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::17
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::18
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::19
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::20
TicketBookingRestController.dummyBookTicket()::21
TicketBookingRestController.dummyBookTicket()::22
TicketBookingRestController.dummyBookTicket()::23
```

Circuit is closed for 20 exceptions So acutal is executing becoz of exception that is riased control is going to fallback method.

for 5 sec only fallback executes becoz the circuit open

```
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::24
TicketBookingRestController.dummyBookTicket()::25
TicketBookingRestController.dummyBookTicket()::26
TicketBookingRestController.dummyBookTicket()::27
```

//for 5secs only fallback mehtod executes becoz the circuit is open

---

Hystrix Command properties

```
@RestController
@RequestMapping("/ticket")
public class TicketBookingRestController {
	int count=0;

	@GetMapping("/book")
	@HystrixCommand(fallbackMethod = "dummyBookTicket",
			commandProperties = {
				@HystrixProperty(name="circuitBreaker.requestVolumeThreshold", value="5"),
				@HystrixProperty(name = "circuitBreaker.sleepWindowInMilliseconds", value = "10000"),
				@HystrixProperty(name = "circuitBreaker.enabled", value = "true")
					}
			)
	public String bookTicket() {
		System.out.println("TicketBookingRestController.bookTicket()");
		if(new Random().nextInt(10)<8)
			throw new RuntimeException("Problem is b.logic");

		System.out.println("End of Ticket Booking Opeation");
		return "output from b.logic(Success)";
	}
	public String dummyBookTicket() {
		count++;
		System.out.println("TicketBookingRestController.dummyBookTicket()::"+count);
		return "Place Try later -- Inconvience is regratted";
	}
}
```

back to back exception count

10 secs (retry-time)

circuitBreaker.requestVolumeThreshold :: *allows us to specify exception count (default is 20)*

circuitBreaker.sleepWindowInMilliseconds :: *allows to sepcify retry sleep time (default is 5 sec)*

circuitBreaker.enabled :: *allows us to specify wheather circuit beaker should be enabled or not (false)*

step4) give *5 to 6* *contionuos requests to RestController from brrowser observe server consoe*

http://localhost:9901/ticket/booking

```
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::1
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::2
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::3
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::4
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::5
TicketBookingRestController.bookTicket()
```

becoz of circut is closed

```
TicketBookingRestController.dummyBookTicket()::6
TicketBookingRestController.dummyBookTicket()::7
TicketBookingRestController.dummyBookTicket()::8
TicketBookingRestController.dummyBookTicket()::9
TicketBookingRestController.dummyBookTicket()::10
```

becoz of circuit is open

(During retry time only fallback method executes

```
TicketBookingRestController.bookTicket()
TicketBookingRestController.dummyBookTicket()::11
```

next request after retry time

```
TicketBookingRestController.dummyBookTicket()::12
TicketBookingRestController.dummyBookTicket()::13
TicketBookingRestController.dummyBookTicket()::14
TicketBookingRestController.dummyBookTicket()::15
```

becoz of circuit is open

35.NTSPBMS615- Hystrix DashBoard -aug 4th-2022

Hystrix DashBoard

==========

=>It is given to provide GUI env.. to know current running MS details and its Circuit Breaker Details like checking wheather Circuit is open or closed.

=>Since hystrix is kept in maintainace mode or deprecated mode in latest spring boot version , we need to use bit old spring boot versions like <2.4

=> For this we need to add new dependencies like hystrix dashboard and we need to place @EnableHystrixDashboard on the main class /starter class..

Example

==========

step1) create spring boot project adding web , hystrix , hystrix dashboard,actuators as dependencies..

from mvnrepository,com

```
<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-hystrix -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
    <version>2.2.10.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-hystrix-dashboard -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
    <version>2.2.10.RELEASE</version>
</dependency>
```

from list of starters

(if starters not possible to add then add them manually)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

step2) modify or add spring cloud version in pom.xml file and perform maven update

```
<properties>
    <java.version>11</java.version>
    <spring.cloud.version>Hoxton.SR4</spring.cloud.version>
</properties>
```

To make the project compitable with hystrix dashBoard.

step3) add @EnableHystrix , @EnableHystrixDashBoard on main class.

=>CircuitBreaker is concept Desing Pattern to resovle the issues related to fault tolerance. (problem) (fault and latency tolerance)

=>Circuit breaker concept says when there is possibility of getting error while using certain microservice locally or remotely .. then make request going to certain fallback method after getting certain count of continuos exceptions/errors from the actual method... This avoid wastage of resources like CPU utilization or network intrastructure and etc..

step4) Develop MS as RestController having @HystrixCommand to specify fallback method

```
@RestController
@RequestMapping("/ticket")
public class TicketBookingRestController {
    int count=0;

    @GetMapping("/book")
    @HystrixCommand(fallbackMethod = "dummyBookTicket",
                    commandProperties = {
                            @HystrixProperty(name="circuitBreaker.requestVolumeThreshold", value="5"),
                            @HystrixProperty(name = "circuitBreaker.sleepWindowInMilliseconds", value = "10000"),
                            @HystrixProperty(name = "circuitBreaker.enabled", value = "true")
                                }
                            )
    public  String   bookTicket() {                                    //same as previous class
          System.out.println("TicketBookingRestController.bookTicket()");
           if(new Random().nextInt(10)<8)
                 throw new RuntimeException("Problem is  b.logic");

           System.out.println("End of Ticket Booking Opeation");
           return "output from  b.logic(Success)";

    }

    public  String dummyBookTicket() {
          count++;
          System.out.println("TicketBookingRestController.dummyBookTicket()::"+count);
          return "Place Try later -- Inconvience is regratted";
    }
}
```

step4) Add addtional properties in application.properties fil;e

application.properties

```
server.port=9901
#enable actuators
management.endpoints.web.exposure.include=*

# enable hystrix streaming list
hystrix.dashboard.proxyStreamAllowList=*
```

hystrix.stream is a dynamically generated spring boot actuator.

ep5) Run the Application.. (Test through DashBoard)

a) give reqeust to MS

http://localhost:9901/ticket/book ing

b) open dashboard of hystrix

http://localhost:9901/hystrix

enter following url related to our app below the image of dashboard

t

1

Hystrix Dashboard

http://localhost:9901/actuator/hystrix.stream

Cluster via Turbine (default cluster): https://turbine-hostname:port/turbine.stream
Cluster via Turbine (custom cluster): https://turbine-hostname:port/turbine.stream?cluster=[clusterName]
Single Hystrix App: https://hystrix-app:port/actuator/hystrix.stream

Delay: 2000 ms Title: Example Hystrix App

Monitor Stream *2*

c) gives multiple requests our MS application using browser (refresh button) and observe the hystrix dash board messages

http://localhost:9901/ticket/book **ing**

(Observe the changes here)

Hystrix Stream: http://localhost:9901/actuator/hystrix.stream

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5
Success | Short-Ci

bookTickets

0 0 0.0 %
0 0
0 0

Host: 0.0/s
Cluster: 0.0/s
Circuit Open

Hosts	1	90th	1ms
Median	1ms	99th	1ms
Mean	1ms	99.5th	1ms

Thread Pools Sort: Alphabetical | Volume |

TicketBookingController

Host: 0.0/s
Cluster: 0.0/s

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	10	Queue Size	5

36.NTSPBMS615- aug 5th 2022 Distributed Tracking&Logging Using slueth -zipkin

Distributed Logging and Tracing

Tracing :: Finding out execution flow /path from request to response is called Tracing

Logging :: Finding out various lines of code and various comps that are involved in the flow of execution having different Logger Levels like DEBUG,INFO,WARN and etc.. is logging

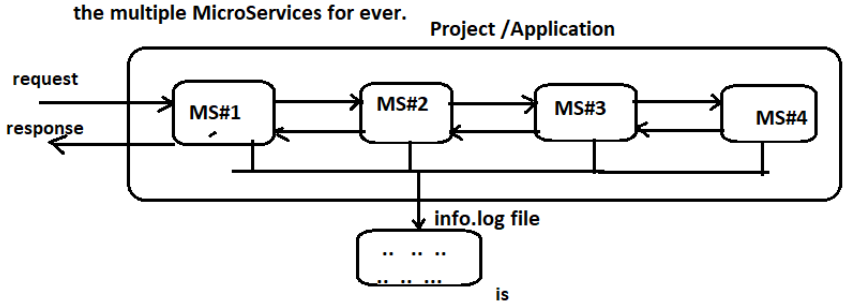
Intra communication of MicroServices

=>It speaks about the communication that happend between multiple microseervices.
=>The request given to one MS taking to another MS is called Intra communication b/w MicroServices.

Distributed Logging and Tracing

=>When the MicroServices are in intra communication enabling tracing and logging activities across the multiple MicroServices to findout exection flow of each and every request to response called Distributed Tracing and Logging.

=>This will enabled on multiple MicroServices for one time to help new developers/ testers /debuggers /UAT team to know execution flow from request to response across the multiple MicroServices for ever.

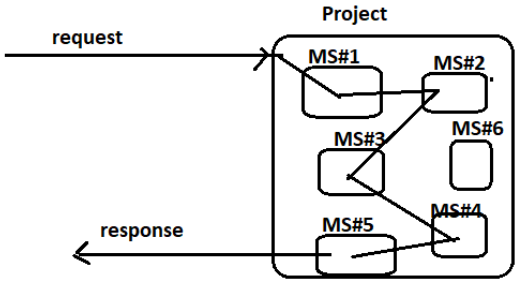


=>Just sl4fj and log4j based logging in every MS keeps track of that MS related Flow of execution having seperate log file for each MS

=> Distributed TRacing or logging also uses SL4FJ or log4j internally but it keeps track of flow of execution /logging across the multiple MicroServices.

=> Logging is extension of Tracing .. In fact logging a kind of tracing activilty .. The only difference is the logging activity writes its log messages to destination called file/Db/mailserver and etc. having ability to filter the messages based on the Logger Levels we have choosen.

TRACING : just keeps track of flow
LOGGING : TRCING+ Logger Levels for messages



if execution is taking place across the multiple MicroSerivces then we need the support of Distributed Logging and Trac ing.

In Tracing -- we can not filter the messages related flow of execution
In Logging --- we can categorize and filter the messages related to flow of execution

Slueth and Zipkin

These two are components/ tools provided by the spring cloud env.. to enable distributed Logging and tracing on the micro services that are in Intra communication.

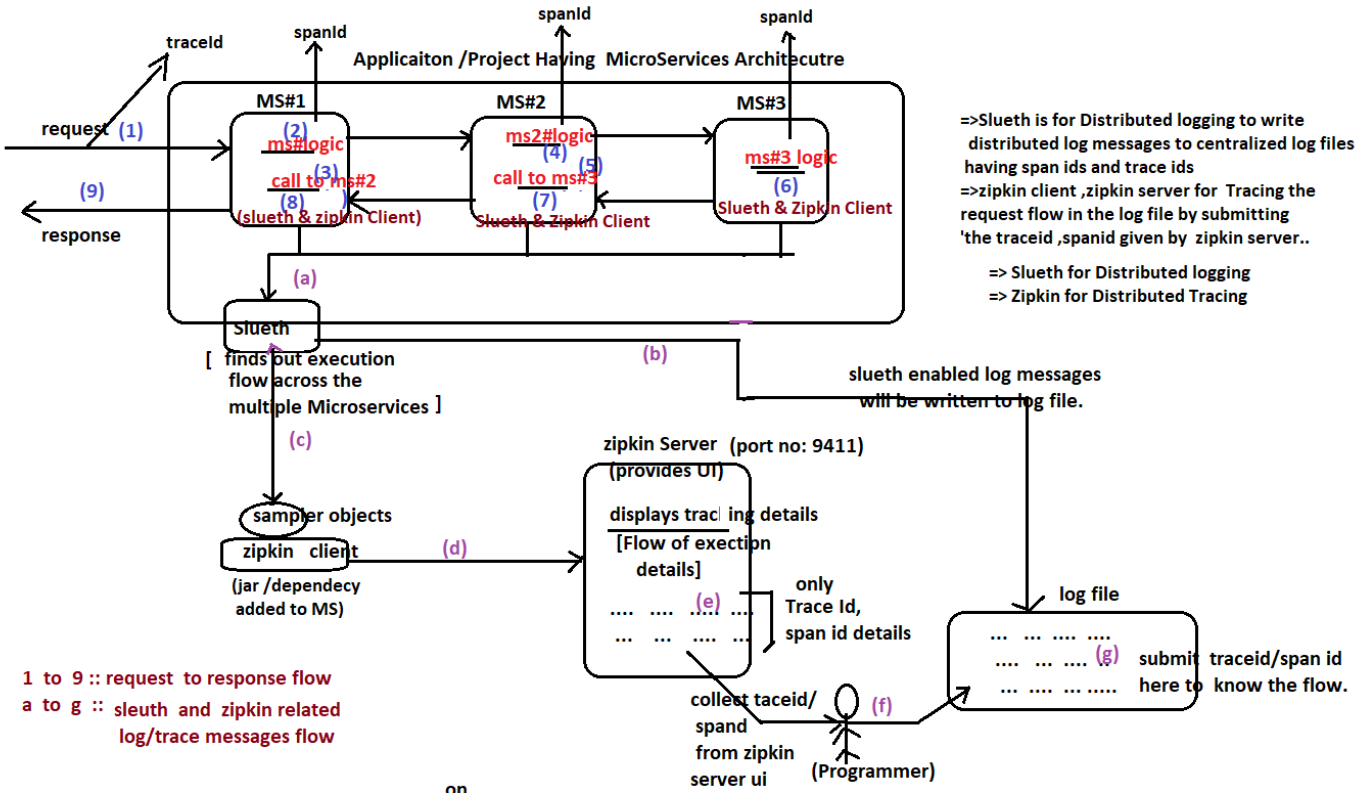
Slueth :: A spring cloud comp providing unique id for each request flow ... The Programmer can use this unique ids to find out the execution flows of requests.

Two types of Ids

traced Id :: unique id for each request flow across the multiple micro services.. if developers gets this traced id he can find out all the microservices that are involved in a request flow.

span id :: Unique id given for each micro service .. if developer gets this span id then he can findout all the executions that happend in a micro service.

Zipkin Client :: We need to add Zipclient dependency to every MS along with
========== sleuth dependency .. It contains sampler obj nothig but
the data collected from MicroServices using Slueth tool to Zipkin
Server) So it can provide lots of details related to trace id , sapnid
and etc.. with respect to the Distributed loggig and tracking enabled on
across the multiple microservices.



Zipkin Server :: provides UI env.. running the port number (9411 default)
========== The zipkin clients of each Ms collects trace ids/spand ids of distributed tracing, logging from slueth comp and gives to to zipkin server to display having UI.

note:: while working with Distribued Logging and tracing each MS contains its own zipclient and slueth comp..but there must be only one zipkin server as centralized server that collects data from all zipkin clients of different MS as sampler objs to display as UI.

note :: The develop /programmer collects trace id/span id from zipkin server ui and seraches in the distributed log file generated by the slueth to find out log messages for flow of execution.

Q) In One Project of MicroServices architecture , can u tell me how many micros serivces will be there?
ans) Mulitple as needed

Q) What is microServices Intra communication
ans) It is the communication b/w microservices (One Ms to another MS communication)

Q) In MicroSerivces Intra Communication , how can we find out which comps of which microservices are executed in the request flow?
Ans) We need enable distributed logging / tracing on multiple Microservices that are in intr Communication.
(In this process every MS should have slueth and zipkin client support connected to the centralized zipkin server)

Q) When to use log4j and when to use slueth and zipkin?

Ans) log4j is required to write log message in both indepndent Microservices execution and the Intra communication enabled microservices execution .. But we link log4j with slueth and zipkin to write the log4j genertd log mesages having trace ids and span ids which helps for distributed logging and tracing.

note: the log messages generated by log4j or sl4fj will be used by slueth for Distributed logging where log messages will appended with span ids , trace ids

---

Keeping Zipkin Server ready
==========

step1) download jar file that represents zipkin server

https://zipkin.io/pages/quickstart ----> go to java section --->
click on latestrelease which gives "zipkin-server-2.23.16-exec.jar" representing zipkin server.

step3) start zipkin server ..

copy "zipkin-server-2.23.16-exec.jar" file to u r choice and execute the jar file

E:\zipkinserver>java -jar zipkin-server-2.23.16-exec.jar

step3) open zinkin server home page

http://localhost:9411/zipkin/

Different Distributed Tracing Logging Tools
==========

- SigNoz
- Jaeger
- Zipkin
- Grafana Tempo
- Serverless360
- Dynatrace
- New Relic
- Honeycomb
- Lightstep
- Instana
- DataDog
- Elastic APM
- Splunk

- Reimann. ...
- Prometheus. ...
- Elastic Stack. ...
- Kibana. ...
- Glowroot. ...
- AWS Cloudwatch. ...
- Datadog.

Example App

==========

=>Keep Zipkin server running mode

=> develop 3 microservices having intra communication + zipkin client and slueth support and execute them

=> Give request to First MicroService

=> perform the following operations in zipkin server console

a) open the home page ::http://localhost:9411/zipkin

b) Click on FindTrace

c) Click on BlueColor Bar

d) Click any One Option (App1 --First MicroSerivce name)

e) collect and copy the TraceId

f) open the log file (App.log) and serach for (ctrl+f) traceId to see all the log messages realated to current request

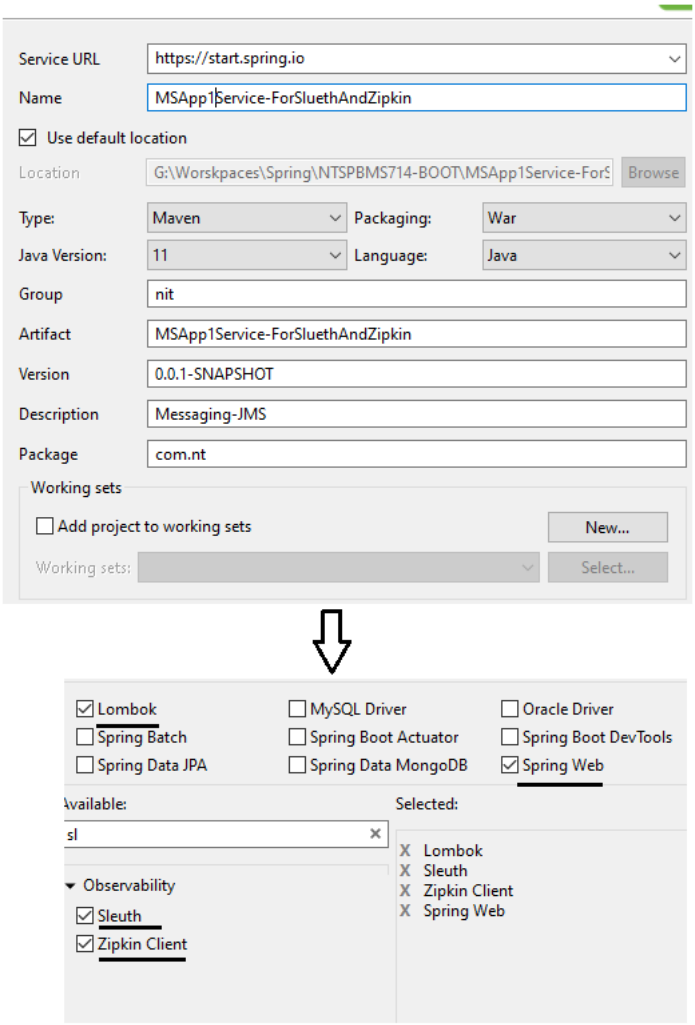
MicrServices Development

==========

First MicroService (App1)

---

a) create spring boot starter project of type war file adding web,lombok , slueth, zipkin client dependencies



b) Add the following entries in application.properties

application.properties

```
server.port=9091
spring.application.name=App1
logging.file.name=E:/logs/App.log
```

c) create objects to make them as spring beans using @Bean methods in @Configuration class or main class

AppConfig.java
==========

```
package com.nt.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

import brave.sampler.Sampler;

@Configuration
public class AppConfig {

    @Bean
    public Sampler createSampler() {
        return Sampler.ALWAYS_SAMPLE;
    }

    @Bean
    public RestTemplate createRestTemplate() {
        return new RestTemplate();
    }
}
```

Spring boot internally uses sl4fj logging linked with logback
=> if we write log messages System.out.println(-) we have the following limitations
a) can write only to Console b) can not categorize log messages
b) can not filter log messages while retrieving them

To over come these problems use logging api/frameworks like log4j, logback , commons-logging and etc.

slf4j provides abstraction on multiple logging frameworks and privides unified env.. to work with all logging frameworks

logging api/frameworks advantages
a) Can categorize log messages
DEBUG< INFO<WARN<ERROR<FATAL
b) can write log messages to different destinations
c) can filter log messages ,
d) can format log messages using layouts

d) *Develop RestController as MicroService*

```
package com.nt.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class ShoppingOperationsController {
    @Autowired
    private RestTemplate template;
    Logger logger=LoggerFactory.getLogger(ShoppingOperationsController.class);

    @GetMapping("/shopping")
    public String Shopping() {
        logger.info("Welcome to shopping Module");
        //communicate with BillingSErvice
        String resp=template.getForObject("http://localhost:9092/billing", String.class);
        logger.info("Back to shopping module::"+resp);
        return resp;
    }

}
```

note: Develop Second MicroServcie and Third MicroSerivce in similar fashion having neccessary chanages
note:: Second MS communicates with Third Ms .. but Third MS does not interact with any Ms
note:: Registering these MicroServices with Eureka is optional .. but recomanded to do..

Second MicrService
==========

=>Since All MicroServices are Rest API comps for intra communucation b/w MicroServices registering with Eureka Service is optional process .. But recomanded to do

=>Project creation :: same as first mS
=>AppConfig.java :: same as first MS
=>application.properties

```
server.port=9092
spring.application.name=App2
logging.file.name=E:/logs/App.log
```

=>Rest Controller class

```
package com.nt.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class BillingOperationsController {
        @Autowired
        private  RestTemplate  template;
        Logger logger=LoggerFactory.getLogger(BillingOperationsController.class);
        @GetMapping("/billing")
        public String doBilling() {
                logger.info("Welcome to Billing Module");
                //communicate with  PaymentSErvice
                String  resp=template.getForObject("http://localhost:9093/payment", String.class);
                logger.info("Back to  Billing module::"+resp);
                return  resp;
        }
}
```

Third MicroService Development
==========

=>Project creation :: same as first mS
=>AppConfig.java :: same as first MS
=>application.properties

```
server.port=9093
spring.application.name=App3
logging.file.name=E:/logs/App.log
```

RestController class

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PaymentOperationsController {
        Logger logger=LoggerFactory.getLogger(PaymentOperationsController.class);
        @GetMapping("/payment")
        public String doBilling() {
                logger.info("Welcome to payment Module");
                return  "payment is done";
        }
}
```

To run the Application

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

=>Keep Zipkin server running mode

```
E:\zipkinserver>java -jar  zipkin-server-2.23.16-exec.jar
```

=> develop 3 microservices having intra communication + zipkin client and slueth support and execute them

order of execuion :: 3rd , 2nd and 1st

=> Give request to First MicroService

http://localhost:9091/shopping

=> perform the following operations in zipkin server console ***+ log file***

a) open the home page ::http://localhost:9411/zipkin

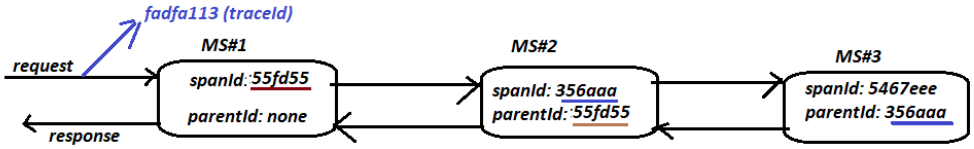
b) Click on FindTrace

c) Click on BlueColor Bar ***(Run query)***

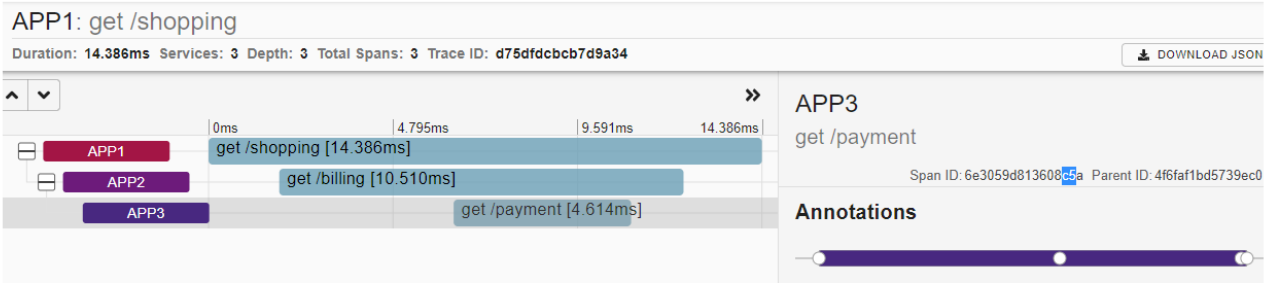
d) Click any One Option (App1 --First MicroSerivce name)

e) collect and copy the TraceId

f) open the log file (App.log) and serach for (ctrl+f) traceId to see all the log messages realated to current request



spanId current MS = parent Id of next MS



37.NTSPBMS615 - API Gateway -zuul -Aug 8-2022

API -Gateway

=> Different MicroServices of project run on different port numbers having different urls .. it is pratically impossible to remember all those port numbers and urls seperately.. So we need single entry and exit point having unique url for all the micro services of the App ..that is API gateway

API Gateway provides
==========
=> Acts as single entry and exit point for the application (For all the micro services of the application)
=> Can perform Authentication and Authorization (Security)
=> Provides Filters for Data /request logging (To find out flow related to request/response/error)
=> Dynamic Routing to the instances of MicroServices (Intenrally uses Ribbon Client Code (Proxy Client code)for this)
=> API Gateways also one kind of MicroService Which is able to call all other MicroSErvices of the application or Project using Eureka s erver.

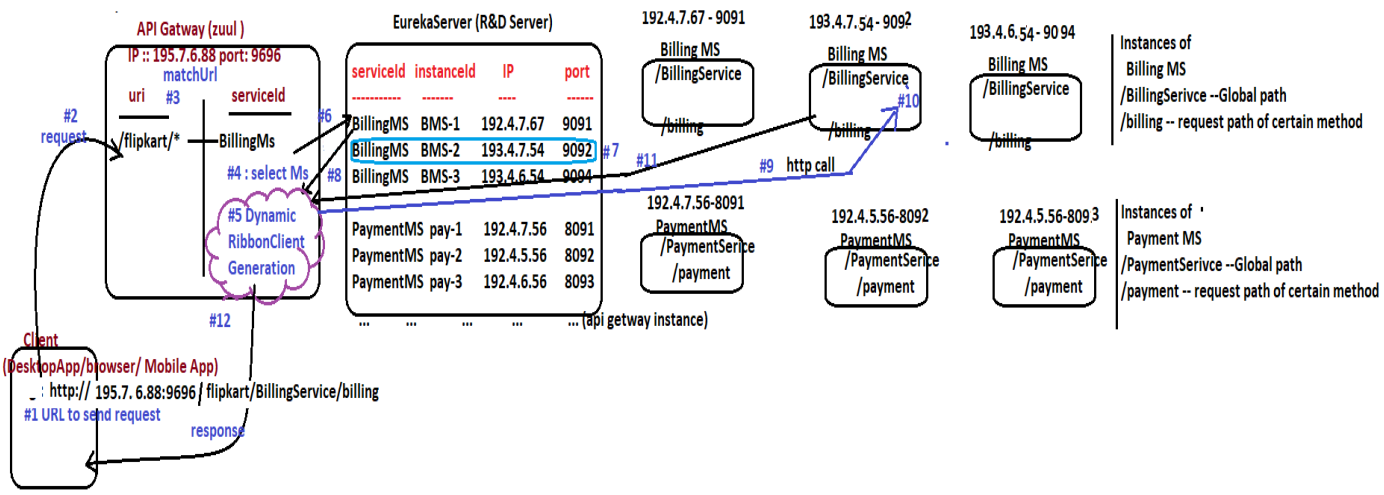
note:: Eureka Server is for registering microService and for finding MicroService ...
Eureka server itself can not communicate with other MicroService in any angel .One MS gets details of other MS from EurekaServer using one or another client code (like LBC, Feign Client and etc..) later uses RestTemplate support to make http call from One MS(source) to another Ms(Destination).

conclusion :: Eureka Server can not communicate with Ms and can not make http calls to interact with any Ms .. It Is just there to register details of One MS, So other MS can use one or another Client Code to find the details about MS(dest) and to interact with that MS (dest)

note ::Even API Gateway MS also should be registered with Eureka service

What is API Gateway ? what is use of it?

=> It provides single entry and exit point all for MicroServices of the Application/ Project.. It itself acts as MicroService having ability to take http requests from Clients and to communicate with other MicroSErvices using http calls through EurekaServer by Generating RibbonClient Code as Proxy code.



URL in the client App
==========
syn :: http://<zuulIP>: <port>/<zuulpath>/<MsGlobalPath>/<Ms Method or req path>
eg:: http:// 195.7. 6.88:9696 / flipkart/BillingService/billing

=> Zuul (API gateway) is also one MS , registering with Eureka
=> Zull Generatoes Client Code to interact with EurekaServer Dyanically as InMemory Proxy class
in the form RibbonClient
=> Zuul interacts with Eureka Server using Dynamic Client Code to get Less Load instance (load balance)

=> The dyanamically generated Client code in Zuul makes http request to communicate with MS whose instanace is gathered from EurekaServer .. later it sends the recieved output to Client as response

With respect to the diagram
==========
#1 :: enduser give URL to Client App/Browser to send request
#2 :: The url based generated request goes Zuul (API gateway)
#3 :: Zuul matches/compares current request url with common paths/zuul paths that are maintained by linking with serviceIds
#4 :: Selects One Service Id based on the matching zuul /common path

#5 :: Generates Dynamic Client Code (as In Memory Proxy class) as Ribbon Client Code

#6 ,#7,#8 :: This Ribbon Client Code contacts the EurekaServer and gets less load instanceId of matched MS

#9 : Ribbon Client generates http call to interact with recieved instance id based MS instance
#10,#11 :: Based on the global path , method path of the URL ... the method in MS intance will execute
and the generate results comes back Ribbon Client
#12 :: Ribbon Client sends the results to Client App/Browser as response

API Gateway - Netflix zuul
==========

Advantages of API Gateway
==========
=>Single entry and exit point for all services of the App for Clients (single url)
=> Dynamic Routing and Load Balancing
[Assigning one SErvice Instance of MS from Eureka Server and making http call to it by checking less load balance service instance availability]
=> Supports SSO (Single Sign On using Oauth2)
[Perform login operation once and access all the MicroServices of the Application using the same credentials)
=> Dynamic Proxy class generation having Client Code (Ribbon Client Code Generation)
=> Provides Filters for data logging.

and etc..

Important points on API Gateway impl (netflix -zuul)
==========
a) API Gateway (like netflix zuul) is also a Micro Service , So it can be used to communicate other real MS
b) API Gateway must be registered with Eureka Server like any other MS
c) In API gateway development we must provide other MS service ID collected from Eureka Server and that should be linked with common path or zuul path
d) The Dynamic Proxy Client Code (Ribbon Client Code) generated based Service Id gets required MS instance from Eureka Server based Load balancing..

Example App
==========
Apps/Projects to develop
==========
a) Eureka Server
b) ShoppingService (calls BillingService)
c) BillingService
d) Zuul Server

step1) keep any old EurekaServer Project ready (use old project)

step2) keep old ShoppingService , BillingService MS Projects Ready PaymentService (use old projet)

step4) Develop API gateWay (netfix Zuul server)

i) create starter Project adding zuul, EurekaClient, web dependencies

=>use spring boot version below 2.5 (2.3.6.RELEASE) (<version>2.3.6.RELEASE</version>)

=>specify the following spring.cloud.version

```
<spring-cloud.version>Hoxton.SR9</spring-cloud.version>
```

=> add the spring cloud netflix zuul as shown below

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
</dependency>
```

ii) Add @EnableZuulProxy and @EnableEurekaClient on the top of main class

```
@SpringBootApplication
@EnableZuulProxy
@EnableEurekaClient
public class SpringBootMsProj08ZuulServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootMsProj08ZuulServerApplication.class, args);
    }

}
```

Spring Cloud api gateway is alternate for netflix zuul api gateway..

iii) Add the following entries in application.properties file

```
#Ms Properties
#Port number
server.port=9797
#Service id
spring.application.name=Zuul-Server
#Eureka server publishing info
eureka.client.service-url.default-zone=http://localhost:8761/eureka

# Common url/path linking with service Id
# syn : zuul.routes.<mod name>.path = common path/Based path
zuul.routes.flipkart.path=/flipkart-api/**
zuul.routes.fpkt.path=/fpkt-billing/**

# specify initial SErivce name pickup
zuul.routes.flipkart.service-id=Shopping-Service
zuul.routes.fpkt.service-id=Billing-Service
```

** indicates multi level path

Execution order

==========

=> Run EurekaServer

=> Run PaymentSErvice for multiple times by changing the port number

=> Run BillingSerivce for multiple times by changing port number

=> Run ShoppingService for multiple times by changing port number

=> Run Zuul Server

=> Open Eureka Server Home page and change the url to the zuul common url

http://desktop-iudaavl:9797/actuator/info (select zuul server link in eureka home page)

to

http://desktop-iudaavl:9797/flipkart-api/shopping/info

- desktop-iudaavl:9797 — zuul server host name and port number
- flipkart-api — zuulpath/ common path
- shopping — ms controller path
- info — Ms method path

for Shopping Ms >>>> BillingMS >>>> PaymentMs

http://192.168.1.236:9797/fpkt-billing/BillingMs/billing

For Billing MS >>> Payment MS

38.NTSPBMS615- API Gateway -zuul Filters -Aug 10th-13th2022

other api gateways KONG server , google clould api server

Top Alternatives to Zuul

- Apigee. API management, digital. ...
- Eureka. ...
- Kong. ...
- HAProxy. ...
- Istio. ...
- NGINX. ...
- Jenkins. ...
- Consul.

*spring cloud gateway

From spring boot 2.5 the zuul api gateways is kept insuperseded mode or maintainance mode ...So think using alternates

Zuul Filters (API Gateway Filters)
==========

note: zuul server acting as API gateway performs request trapping, request rotung to MS Instance , response trapping and etc.. internally and automatically .. To trace' those operations of ZUUL API gateway take the support of Zuul Filters

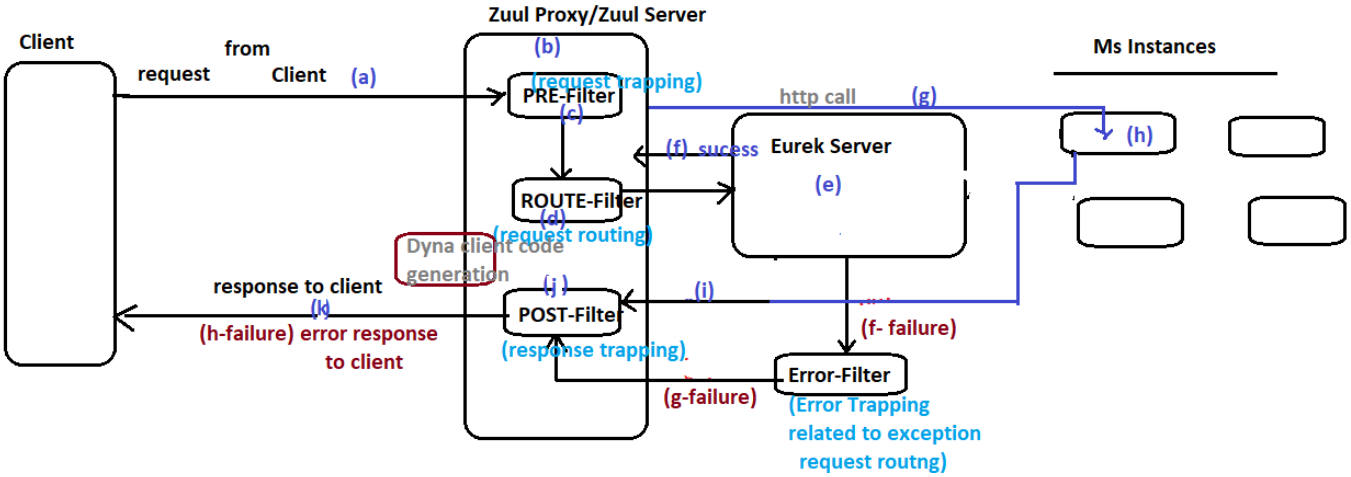
=> These are given to perform logging activies for request to response flow w.r.t zuul server
=> No so much used in real pratices (In real spring boot Ms Projects)
=> These filters are bit similar to Servlet Filters .. but these are based on different api
=> Zuul filters allows us to trace each request , each response , each error details with respect to the client generated request to response flow being from zuul server
=> zuul filters uses log4j/slf4j for tracing logging activities .. These are with respect to zuul server .. These are for addtional logging activiees on request to response flow along with other requalr logging activities done by slueth and zipkin
=> Zuul Filters takes less memory compare to Servlet flters .. But usign zuul filter we can do only logging /tracing activies where as using servlet filters we can do logging/tracing activies and also data compression activies. (some b logic)
=> slueth, zipkin internally slf4j -log4j to perform logging and tracing activities on Real Ms .. To bring the zuul server's request trapping, response trapping, request delegation activities into logging and tracing operations we need to use zuul filters.

3types of servlet fitlers
=========
a) request filter (contains only pre-request processing logic)
b) response filter(contains only post-response generation logic)
c) request-response filter(contains both pre-request processing and post-response generation logic)

4 types of zuul flters
=========
PRE Filter :: executes for every client request before entering into zuul server (executes for req trapping)
ROUTE Filter:: Executes before the request flow goes to Eureka server and Selectes MS Instance (executes for request routing)
POST Filter :: exeutes after MS execution and before sending response to client (executes for response trapping)
ERROR Filter :: executes only when the problem is there in routing (going to eureka server to get MS Instance) like unable to fetch service instance , Eureka server not found or eureka server not responding and etc..
(Executes for the problem is request routing to MS Instance)

picking up the MS instance through Eureka server

PRE/POST/ROUTE Filters execute for every request to response flow .. But the Error filters executes only when problem is there in routing.



com.netflix.zuul.IZuulFilter(I)
|--->shouldFilter()
|---->run()
method declarations (abstract methods)
implements

com.netflix.zuul.ZuulFilter(AC)
|--->filterType()
|--->filterOrder()
abstract methods
extends

com.nt.MyFilter (c) (our class)

=>our class must provide impl to all the 4 methods

To develop Zuul Filter , we must take a class extending from com.netflix.zuul.ZuulFilter(AC)
and should override 4 methods giving certain info

a) Enable /Disable Filter (true/false) :: shouldFilter()
b) Filter logic/code :: run()
c) FilterType (PRE/POST/ROUTE/ERROR) :: filterType()
d) Filter Order (Priority Order when multiple filters are available) :: filterOrder()

//Sample Filter Code
==========

```
package com.nt.filter;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.exception.ZuulException;

public class MyZuulFilter extends ZuulFilter {

        @Override
        public boolean shouldFilter() {
                //false  :: disable filter
                // true  :: enable filter
               return true;
        }

        @Override
        public Object run() throws ZuulException {
           //fiter logic /code  ::(Logging logics using slf4j)
               return null;
        }

        @Override
        public String filterType() {
                //"pre" or "post" or "route" or "error"
               return "pre";  //pre filter
        }

        @Override
        public int filterOrder() {
               return 0;  //low value indicates high priority and  high value indicates low priority
        }
    }
```

we can use also the constants of FilterConstants class to specify the filter type .. They are
POST_TYPE (internal value is :: post)
PRE_TYPE (internal value is :: pre)
ROUTE_TYPE (internal value is :: route)
ERROR_TYPE (internal value is :: error)

=> Zuul Filters are given for logging operations with respect to different phases of request-response in flow in zuul proxy server integration env.. (To trace internall actives of zuul server like request trapping, request routing, response trapping and error trapping)

=> 4 types of Zuul filters are avaiable
a) PRE b)ROUTE c) POST d) ERROR

=> For every request to response flow PRE, ROUTE, POST zuul Filters will execute but the ERROR will be execute only when problem is there towards routing the request to Eureka server or MS

Normal request -response flow executes
PRE,ROUTE,POST Filters
request -response flow having problem in request routing executes
PRE,ROUTE,ERROR,POST Filters

=> To develop Zuul Filter , the class must extend from com.nefflix.zuul.ZuulFiter(AC) having implementation of 4 abstract methods.

a) boolean shouldFilter() --- true for enabling filter
flase for disabling filter
b) String filterType() ---- should one of the four "pre","route","post","error"

c) Object run () ----> Filter logic /code .. Generally logging activity using slf4j
d) int filterOrder() ---> prority order if we develop multiple filters of same type
(high value indicates low priority and low values indicates high priority)

Example App on Zuul Filters

a) keep Eureka server App ready (collect old App)

b) keep One or two MicroServices Apps ready
registering with Eureka Server. (Collect old ShoppingService, BillingServer Apps ,PaymentServuce Apps)

c) Develope App acting as ZuulServer (Collect previous Zuul Server App) and perform the addtional activities

i) Add the following addtional entries in application properties to enable the logging

```
# For logging operation
logging.file.name=E:\logs\zuul-filters.log
logging.level.com.nt.filter =info
```

ii) Develop all the 4 types of filters by taking classes extending from ZuulFilter(AC) and implementing 4 methods in that class.

//Pre Filter code

```
package com.nt.filter;
import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.cloud.netflix.zuul.filters.support.FilterConstants;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
@Component
public class ZuulPreFilter extends ZuulFilter {
   private Logger logger= LoggerFactory.getLogger(ZuulPreFilter.class);
      @Override
      public boolean shouldFilter() {
            return true; // enables this  filter
      }

      @Override
      public Object run() throws ZuulException {
        //  get RequestContext object  (it contains access to multiple other objects)
            RequestContext context=RequestContext.getCurrentContext();
            // get HttpServletRequest object from this
            HttpServletRequest  req=context.getRequest();
            System.out.println("ZuulPreFilter.run():: from pre-filter");
            //write log messages about  current  request
            logger.info("======From  PRE filter=====");
            logger.info("request content type type::"+req.getContentType());
            logger.info("request mode::"+req.getMethod());
            logger.info("request path::"+req.getServletPath());
            logger.info("request uri:: "+req.getRequestURI());
           return null;
      }
      @Override
      public String filterType() {
            return  FilterConstants.PRE_TYPE;  //makes the filter as prefilter
      }
      @Override
      public int filterOrder() {
            return 0;  //high priority
      }
}
```

//PostFilter code

```
import javax.servlet.http.HttpServletResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.cloud.netflix.zuul.filters.support.FilterConstants;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
 @Component
public class ZuulPostFilter extends ZuulFilter {
   private Logger logger= LoggerFactory.getLogger(ZuulPostFilter.class);
      @Override
      public boolean shouldFilter() {
            return true; // enables this  filter
      }

      @Override
      public Object run() throws ZuulException {
        //  get RequestContext object  (it contains access to multiple other objects)
            RequestContext context=RequestContext.getCurrentContext();
            // get HttpServletResponse object from this
            HttpServletResponse  res=context.getResponse();
            System.out.println("ZuulPostFilter.run():: frompostfilter");
            //write log messages about  current  request
            logger.info("======From  Post filter=====");
            logger.info("response content type type::"+res.getContentType());
            logger.info("response status::"+res.getStatus());
            logger.info("response status code::"+res.getHeader("host"));
           return null;       }

      @Override
      public String filterType() {
            return  FilterConstants.POST_TYPE;  //makes the filter as postfilter
      }

      @Override
      public int filterOrder() {
            return 0;  //high priority
      }

}
```

```java
// Roiter Filter
package com.nt.filter;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.cloud.netflix.zuul.filters.support.FilterConstants;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
@Component
public class ZuulRouteFilter extends ZuulFilter {
   private Logger logger= LoggerFactory.getLogger(ZuulRouteFilter.class);
      @Override
      public boolean shouldFilter() {
            return true; // enables this  filter
      }

      @Override
      public Object run() throws ZuulException {
        //  get RequestContext object  (it contains access to multiple other objects)
            RequestContext context=RequestContext.getCurrentContext();
            System.out.println("ZuulPostFilter.run():: from route-filter");
            //write log messages about  current  request
            logger.info("======From  Route filter=====");
            logger.info("router host::"+context.getRouteHost().toString());
           return null;        }

      @Override
      public String filterType() {
            return  FilterConstants.ROUTE_TYPE;  //makes the filter as routefilter
      }

      @Override
      public int filterOrder() {
            return 0;  //high priority
      }

}
```

if we take multiple zuul filter of same type then this priority order comes into picture. otherwise no spcific significance for zuul filters.

ErrorFilter

```java
package com.nt.filter;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.cloud.netflix.zuul.filters.support.FilterConstants;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;
@Component
public class ZuulErrorFilter extends ZuulFilter {
   private Logger logger= LoggerFactory.getLogger(ZuulErrorFilter.class);
      @Override
      public boolean shouldFilter() {
            return true; // enables this  filter
      }

      @Override
      public Object run() throws ZuulException {
        //  get RequestContext object  (it contains access to multiple other objects)
            RequestContext context=RequestContext.getCurrentContext();
            System.out.println("ZuulErrorFilter.run():: from Error-filter");
            //get Throwable object
            Throwable th=context.getThrowable();
            //write log messages about  current  request
            logger.info("======From  Error filter=====");
            logger.error("Exception message"+th.getMessage()+" class name:: "+th);
           return null;
           }

      @Override
      public String filterType() {
            return  FilterConstants.ERROR_TYPE;  //makes the filter as routefilter
      }

      @Override
      public int filterOrder() {
            return 0;  //high priority
      }
}
```

=>Zuulfilter will be excuted automatically in request -response flow of zuul server integrated application.
In that process .. the call back methods will also executed automatically..

Execution order1 (To feel pre,post , route filters)

==========

a)start Eureka server
b) start all MS (either singgle instance or multiple instances)
c) Start Zuul Server
d) go to EurekServer home page and change
Zuul url as show below

Filter execution order
pre
route
post

From :: http://desktop-iudaavl:9797/actuator/info
To :: http://desktop-iudaavl:9797/flipkart-api/shopping/info
(for success response)

Execution order2 (For Error Filter execution)

---

a)start Eureka server
b) Start Zuul Server
c) Do not start any microServices ***

d) go to EurekServer home page and change
Zuul url as show below

From :: http://desktop-iudaavl:9797/actuator/info
To :: http://desktop-iudaavl:9797/flipkart-api/shopping/info

Filter execution order
pre
route
error
post

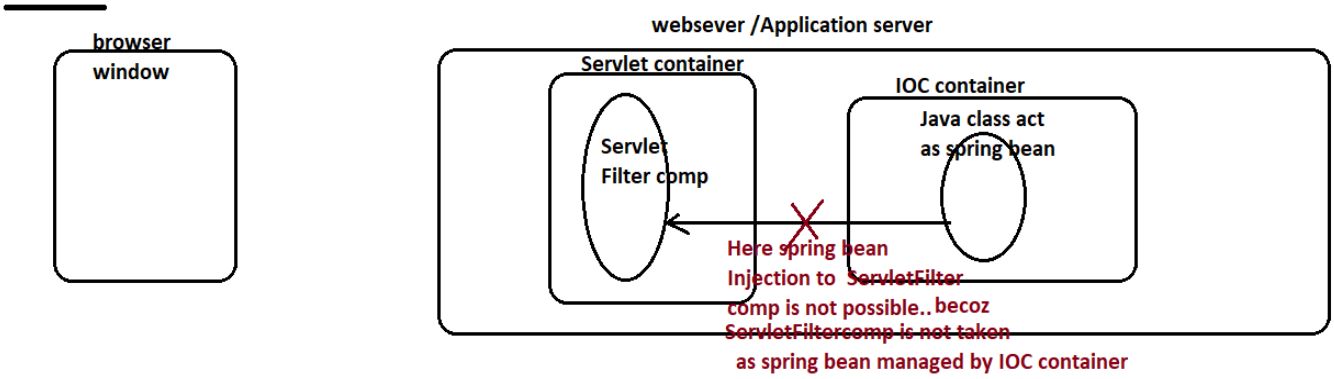
This leads to execution of ErrorFilter
becoz No MicroSErvice in started.

39.NTSPBMS615-Intro to spring security -aug14th

Spring boot security
==========

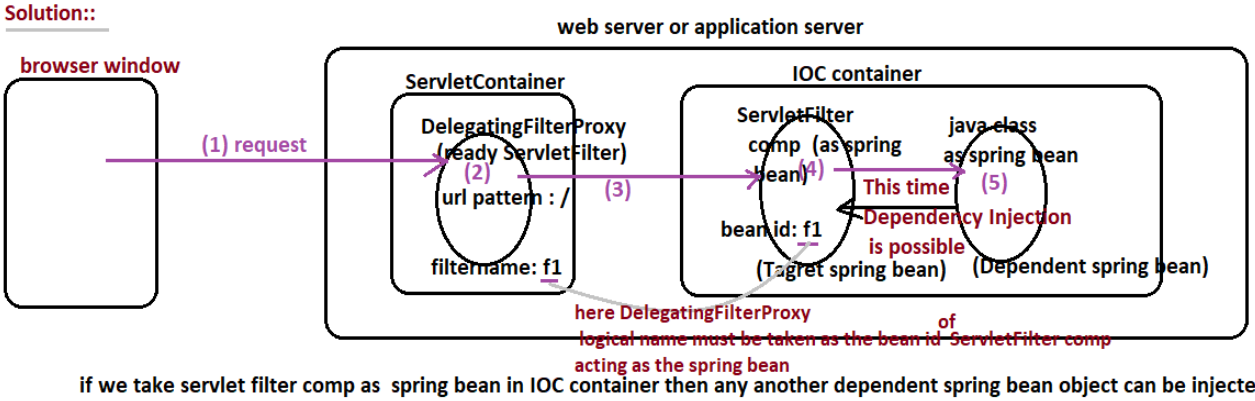
=> It is spring extension module that provides multiple readymade filters to enable security on spring boot MVC and spring boot rest applications.

Problem:



to
=>In order inject Dependent spring bean to servlet filter comp , we need to take ServletFilter comp also as spring bean in IOC container

Solution::



if we take servlet filter comp as spring bean in IOC container then any another dependent spring bean object can be injected to that Servlet filter spring bean.. but that servlet filter can not trap and take requests given browser window.. For this we need to cfg special ready made ServletFilter comp given by spring security module that is "DelegatingFilterProxy" having url pattern "/" and logical name matching with ServletFilter comp's bean id.

w.r.t diagram
==========

(1) browser gives the request to web application
(2) DelegatingFilterProxy traps and takes the request
(3)DelegatingFilterProxy delegates/passes the request to ServletFilter comp that is acting as spring bean in IOC container
(4) the logics in ServletFilter comp executes
(5) ServletFilter comp uses the injected dependent spring bean services through method calls.

=>Security in web application is nothing but enabling Authentication + Authorization on the web application

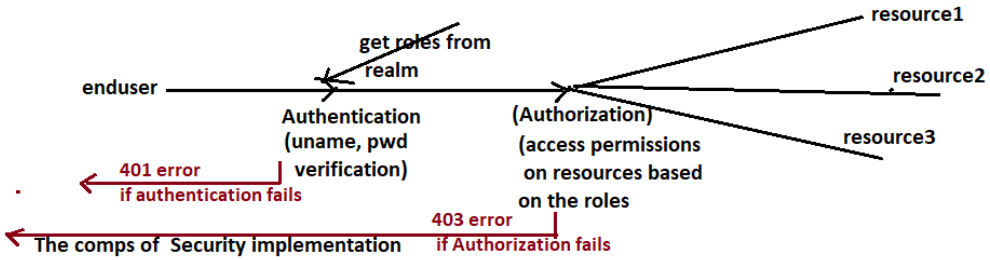
Authentication:: checking the indentity of user using usernames, passwords , thumb impressions, iris , digital signatures and etc..
Authorization :: Checking the access permissions of the autheticated users on different resources of the Project.
(Here the roles of the user will be verified before allowing the user to access different resources of the Project)

=>Roles are nothing but designations given to users .. based on roles of the users the access permissions on different resources will be decided.

=>All customers, employees of the bank must be authenticated to use bankApp..
=>The users having customer role will get less access peromission on the resoruces where as the users having employee role will get more access peromission on the resoruces.

=> It is always recomanded to enable authorization of accessing the resoucers not based on the username.. It is recomanded to perorm on the roles of the users.

=>during the authentication process .. get the roles of the user and use those roles for authorization.



realm means small DB s/w or repository where the usernames, passwords and roles are managed.

The comps of Security implementation
==========
a) Authentication provider/ Authentication Info provider
b) Authentication and Authorization manager

a) Authentication provider/ Authentication Info provider (also called realm)

It is the small realm where usernames, passwords ,roles are managed and will be used during both authentication and authorization.

a) Properties file
b) xml file file
c) json file
d) Db s/w
e) LDAP server (best)
g) InMemory DB

and etc..

LDAP :: Lightweight Directory Access Protocol

=>In all other Authentication providers we can get the password of the users if they forgot the passwords.. where as in LDAP server .. no proivision to get password.. only resetting of password is possible.

All latest apps like gmail, yahoo mail and etc.. are using LDAP Server as the Authenticaiton Info Provider

b) Authentication and Authorization manager

=>It is the component that verifies given username and password to perform authentication and gives 401 error if authenticatio fails.
=>The same component collects the roles of authenticated user and performs Authorization activities for accessing different resources.. gives 403 error if authroization fails.

=>The Authentication and Authorization manager can be arranged in two ways
a) Using Programatic approach (bad)
=>Here we need to develop the logics of Authentication and Authorization manager explicitly by spending huge amounts of time and brain manually (Not recomanded)
b) Using Declarative approach (good)
=> Max Web server /App servers provide built-in security support by providing built-in Authentication and Authorization manager ... By adding entries web.xml file we can activate that Authentication and Authorization manager (refer security in web application in servlket,jsp env..)

Limitations of Declarative approch for securiing web applications (working with servlet container supplied Authentication and Authorization manager)
==========

a) Only selected servers supports this features .. some times we need to purchase License to costly web server or Application server to use this feature
b) As of now this facility possible by adding addtional entries in web.xml file .. i.e not suitable in 100% code driven cfgs , spring boot apps
c) When move from one server to another server these configurations in web.xml file may change (web.xml entries related to security are not portable across the multiple servers)
d) No support for LDAP server as authentication info provider.

and etc..

To overcome these problems we use spring security or spring boot security

Advantages of spring security /spring boot security
==========
a) Can be used in spring MVC/spring rest /spring micros services apps and also in non-spring based web applications like servlet ,jsp web applications, JSF web applications and etc...
b) The security cfgs code are portable across the multiple servers
c) We can this security irrespective of wheather the underlying server supports the ServletContainer level Declarative security service or not (spring security no way related to Servlet container's security)

d) Supports different Authentication Info providers includng LDAP.

e) We need not to arrange costly servers only for security..

f) supports 100% code driven security cfgs , So very useful for spring boot apps

In spring env or spring boot env.. we can apply security on MVC apps or spring rest apps or micro services Apps

3 approaches

a) Using Spring security/spring boot secuirty

i) Basic Authentication (browser generates dialog box asking username, password)

ii) Form based Authentication (readymade or user-defined form page will be there asking username, password)

-> Using InMemory DB as authentication Info provider (RAM Level DB)

-> Using Properties file as authentication Info provider

-> Using DB s/w as authentication info provider

with the support of spring JDBC/ sprjing ORM/spring data JPA / UserDetails service

->Using LDAP server as authentication info provider.

b) Using JWT (JSON web tokens)

c) Using Oauth 2.x (Open Authorization)

d) okta

note: oauth and okta can be used for SSO (SSO :: SingleSignOn)

Spring Boot security

==========

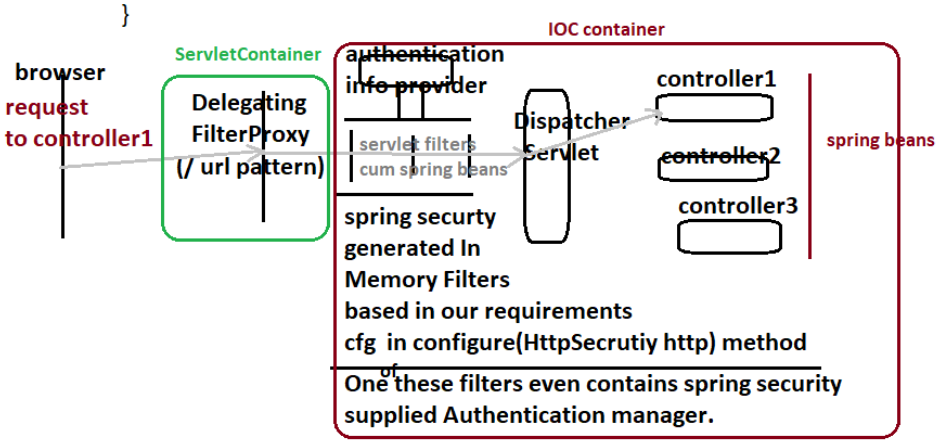
=>Once we add spring-boot-stater-security to spring MVC /spring Rest/ microservices project one ready made filter called "DelegatingFilterProxy" will be registered with "/" url pattern having logical name "springSecurityFilterChain" ..

=> For this One class will be genearted exteing from AbstractSecurityWebApplicationInitializer(AC) class (Internal class)

=> we need to develop ConfigurationAdapter class as @Configuration class extending from WebSecurityConfigurerAdapter overriding two configure(-) methods having authentication info provider, authentication, authorization details.

Every details we add in authentication and authorization seperate InMemory Filters will be generated as spring beans and they will be linked with DelegatingFilterProxy filter i.e DelegatingFilterProxy traps the request and links these requests dynamically generate filters then passes to controller clases through DispatcherServlet.

```
@Configuration
@EnableWebSecurity
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {
        // provide logic for configuration Authentincation Info provider  like InMemoryDB, DB s/w and etc..

    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
            // provide  logic  for  Authentication and authorization  and etc..

    }

}
```



Authorization levels in spring security

---

a) permitAll() :: No authentication + No Authorization (no role checking)
eg:: home page , about us page , contact us page , terms and conditions page

b) authenticate() :: Only Authentication on the given request url resource(controller) and no authorization (no role checking)
eg:: main menu page ,inbox page ,send/composite mail page

c) hasARole() :: Authentication + authorization (role checking will be there)
eg:: checking balance page , transfer money ,withdraw/deposite money page ,changing password
deleting mails and etc..

d) hasAnyRole :: Authentication +authorization (any one role should be there for user
in the list of given roles)

eg:: checking balance page , transfer money ,withdraw/deposite money page ,changing password
deleting mails and etc..

controller classes request paths
/home ----------> permitAll
/contactUs ---------> permitAll
/aboutUs ---------->permitAll
/inbox ----------> authenticated
/checkbalance ---------> hasAnyRole "USER","MANAGER"
/transferMoney ----> hasARole "USER"
/depositeMoney ----> hasAnyRole "USER","MANAGER","VISITOR"

40.NTSPBMS615-Example App on Spring Boot Security -Aug 16th

=>using InMemory DB (RAM Level DB) as authentication pprovider is good only in Dev,Test env../profile of the Project..
In Uat, Prod env../profile prefer using DB s/w or LDAP SErver as the authentication provider

Procedure to develope spring boot security App that is InMemory DB (RAM Level) as the authetication provider using DB

(DB be will be created on the startup of application and DB will be vanished on shutdown of Apps)

url	jsp page	security enabled
/ --->	home.jsp	permitAll()
/offers	offers.jsp	authenticated()
/balance	balance.jsp	authenticated() + Authorization() :: hasAnyRole("CUSTOMER","MANAGER")
/loanApprove	loan.jsp	authenticated + authrozation :: hasRole("MANAGER")

=>once we add spring security starter to spring web mvc/sping rest/ spring Ms project then it automatically takes care of configuring DelagatingFilterProxy Servlet Filter with "/" url pattern

step1) create spring stater project adding the following starters
a) spring security b) spring web c) spring devtools
(The changes will be updated automatically)

step2) Develop the regular controller class having different handler methods with different request paths

controller class

```
package com.nt.controller;

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

@Controller
public class BankOperationsController {

	@GetMapping("/")
	public String showHome() {
		return "home";
	}

	@GetMapping("/offers")
	public String showOffers() {
		return "offers";
	}

	@GetMapping("/balance")
	public String checkBalnace() {
		return "show_balance";
	}

	@GetMapping("/loanApprove")
	public String approveLoan() {
		return "loan";
	}

	@GetMapping("/denied")
	public String accessDenied() {
		return "access_denied";
	}
}
```

Handler method to show authorization failure page

jsp pages

home.jsp

```
<%@page isELIgnored="false"%>

<h1 style="color:red;text-align:center"> Welcome Xyz Bank --Home page </h1>

<a href="balance"> Check Balance</a> <br> <br>
<a href="loanApprove"> Approve Loan</a> <br> <br>
<a href="offers"> Show offers</a> <br>
```

//loan.jsp

```
<%@ page isELIgnored="false" import="java.util.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
	<h1 style="color:blue;text-align:center"> Loan Approval Page </h1>
		<b> u r approved for loan amount :: <%=new Random().nextInt(1000000) %>
	 <a href="./">Home</a>
</body>
</html>
```

show_balance.jsp
==========

```
<%@ page  isELIgnored="false" import="java.util.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
      <h1 style="color:blue;text-align:center"> ShowBalance Page  </h1>
            <b> balance ::  <%= new Random().nextInt(100000) %></b>
       <a href="./">Home</a>
</body>
</html>
```

offers.jsp
=======

```
<%@ page  isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
      <h1 style="color:blue;text-align:center"> Show Offers Page  </h1>
            Home Loan   ROI :: 7%  <br>
            FourWheeler  Loan   ROI :: 8% <br>
            Personal Loan  ROI :  12% <br>
       <a href="./">Home</a>
</body>
</html>
```

step3) **Decide authentication and authorization level for different requeest urls**

/ ---> permitAll()
/offers --> authenticated()
/balance ---> authenticated() + Authorization() :: hasAnyRole("CUSTOMER","MANAGER")
/loanApprove ---> authenticated + authrozation :: hasRole("MANAGER")

step4) Develope SecurityConfig class extending WebSecurityConfigurerAdapter and having annotations @Configuration + @EnableWebSecurity and also overriding two configure(-) methods

SEcurityConfig.java
==========

```
package com.nt.config;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity   // makes the Normal @Configuration class Spring Security Configuration class
public class SecurityConfig extends ~~WebSecurityConfigurerAdapter~~ {
                                    deprecated from  spring security 5.4
    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {
      // Build Authentication Manager by taking  given Authentication Info Provider (InMemoryDB)
        /*    auth.inMemoryAuthentication().withUser("raja").password("{noop}rani").authorities("CUSTOMER");
              auth.inMemoryAuthentication().withUser("ramesh").password("{noop}hyd").authorities("MANAGER");*/

        auth.inMemoryAuthentication().withUser("raja").password("{noop}rani").roles("CUSTOMER");
        auth.inMemoryAuthentication().withUser("ramesh").password("{noop}hyd").roles("MANAGER");
    }
```

To cfg Authentication Provider

Creates InMemory DB (RAM level DB) and uses it a Authentication Provider

```
    @Override
    public void configure(HttpSecurity http) throws Exception {
        //authorize requests
        http.authorizeRequests().antMatchers("/").permitAll()  //Not authetication an no authorization
        .antMatchers("/offers").authenticated()  //only authentication
        .antMatchers("/balance").hasAnyRole("CUSTOMER","MANAGER")  // authentication + authorization for  "CUSTOMER","MANAGER" role uses
                                .antMatchers("/loanApprove").hasRole("MANAGER")  //// authentication + authorization for
                                .anyRequest().authenticated()  //remaing all requests url mus be authtenticated
```

Authentication + Authorization cfgs

```
        //specify  authentication mode
        .and().httpBasic()

        //exception/error handling    (for 403 error)
       .and().exceptionHandling().accessDeniedPage("/denied");
```

This request path based handler method of Controller class executes to display the error page for authrization failure.

```
    }

}
```

step5) develop the application.properties having required entries

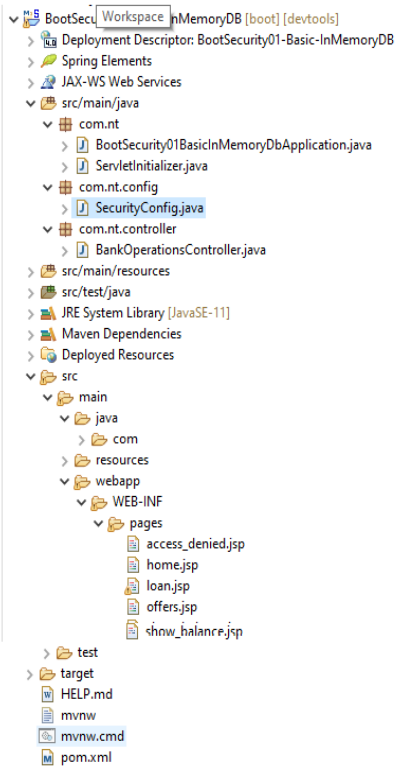
application.properties

```
# Embedded Tomcat server port number
server.port=4041

#View Resolver cfg
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp
```

step5) Run the Application (we can use embeded Tomcat or extenal Tomcat)

note:: while runnung the above web application as spring boot app using embedeed Tomcat server , we need to add tomcat embedded jasper dependency

```
<!-- https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-jasper -->
<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-jasper</artifactId>
 </dependency>
```



Limitations of BASIC mode authentication

a) The dialog box asking username,password is browser specific dialog box and it can not be customized

b) Does not allow to add the following features

a) Logout b) rembermber Me c) SessionMaxActiveCount Limit and etc...

=>To overcome the above problems take the support of form login

.and().httpBasic() ---> gives BASIC mode of authetication

.and().formLogin() --> gives FORM mode of mode of Authetication. with default/fixed url called login

=> .and().rememberMe()

=>Adds another Filter supporting Remember me Authentication .. Internally uses Persistent cookies to remember given username,password having 2-weeks : expiry time.. In this 2 weeks after successful singin .. if u close the browser by taking the URL from browser address bar .. then we can use the same url to get back to the page with out any signin activitty.

```
// add SessionMaxConcurrency count
.and().sessionManagement().maximumSessions(2).maxSessionsPreventsLogin(true);
```

=>Adds another Filter to controller max sessions for each user to operate the application.

```
//add Logout Filter
        .and().logout()
```

code in UI page

```
<a href="logout">logout</a>
```

This fixed /default logout url

==>adds another Filter providing signout activity having the url or request path "/logout" .. by default This can be changed can be with addtional code.

```
.and().logout().logoutRequestMatcher(new AntPathRequestMatcher("/signout"))
```

Code in UI page

```
<a href="signout">logout</a>
```

Internal of Session Management (It is the process of remembering client data across the multiple requests during a session like login to logout)

---

if Login is successful it creates new Session::

HttpSession ses=req.getSession(); or HttpSession ses=req.getSession(true);

To access the existing Session::

HttpSession ses=req.getSession(false);

To stop/invalidate the Session

ses.invalidate();

To specify max inactive interval period for a Session

ses.setMaxInactiveInterval(20); //20 secs //default is 30 secs

AntMachers in spring boot security

==========

=> Spring boot seurity App every URL (nothing but request path of handler method) must be configured with security using permitAll()(no authentication and no authorization) , authenticatied() (only authentication) hasRole() , hasAnyRole() (Authetication +Authorization) .. For this we need to use AntMatches concept in SecurityConfig class.

case1:: AntMatcher for Multipath

```
@Controller
@RequestMapping("/customer")
public class CustomerController{
    @GetMapping("/register")
    public String registerCustomer(){.....}
    @GetMapping("/delete")
    public String deleteCustomer(){.....}
    @GetMapping("/update")
    public String updateCustomer(){.....}
}
```

=> To match with mutlilevel path we can give </path>** like /customer** in AntMatcher

note:: multiple level path is like /customer/register , /customer/delete , /customer/update, /customer/register/abc , /customer/update/type

```
.antMatchers("/customer**").hasRole("MANAGER"); // Only "MANAGER" role authenticated users
                    can access web pages whose URLs starts with /customer and contains multi path
```

case2: AntMatcher for singleLevel path

```
@Controller
public class CustomerController

    @PostMapping("/registerCustomer")
    public String registerCustomer(){
        ...
    }
    @PostMapping("/registerProduct")
    public String registerProduct(){
        ...
    }
    @PostMapping("/registerFaculty")
    public String registerFaculty(){
        ...
    }
}
```

we can give AntMatcher using </path>* pattern..

```
.antMatchers("/register*").hasAnyRole("MANAGER","CUSTOMER")
```

matchers with /registerCustomer , /registerProduct , /registerFaculty urls

case3: Multiple URLs can be given in single AntMacher expression

version1:

```
.antMatchers("/save").hasRole("MANAGER")
.antMatchers("/update").hasRole("MANAGER")
.antMatchers("/delete").hasRole("MANAGER")
```

version2 (Improved code of version1)

```
.antMatchers("/save","/update","/delete").hasRole("MANAGER")
```

case4: Left over request urls can be identified and mapped using .anyRequest() expression.

Let assume we are having multuple request urls/paths as shown below

"/save", "/update" , "/delete","/report" ,"/upload",
"/download" , "/paging" ,"/info" ,"/aboutUs"

```
.antMatchers("/save","/update").hasRole("CUSTOMER")
.antMatchers("/report","/upload","/download").hasRole("MANAGER")
.anyRequest().authenticated();  // represents the left over urls like "/paging","/aboutUs","/info"
```

=>{noop}<pwd> indicates Password is not encoded indirectly it says "NoopEncoder" is to econde the passowrd.
=>Initial spring security used allow not encoded passwords.. Later it stopped allowing them .. So to pass Non-encoded passwords we need use {noop}pwd.

=> We can use different Encoders like "BCryptEndcoder" and etc.. to encode the passowrds..
"Base64Encoder"
"SHA512Encoder"
"MD5Encoder"

=> If do not to use {noop} expression based "NoopPasswordEncoder" then we need to pass encoded passwords as shown below.

step1) take sepeate App to get Encoded passwords.

```
package com.nt.encrypt;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public class PasswordEncoder {

    public static void main(String[] args) {
    BCryptPasswordEncoder encoder=new BCryptPasswordEncoder();
    String pwd1=encoder.encode("rani");
    String pwd2=encoder.encode("hyd");
    System.out.println(pwd1);
    System.out.println(pwd2);

    }

}
```

step2) Run the above App and get Encoded passwords and use them in 1st configure (-) method of Securityconfig class

```
@Override
public void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.inMemoryAuthentication().passwordEncoder(new BCryptPasswordEncoder()).withUser("raja").password("$2a$10$irWQwKOV6vm9Ksf7b11Ttu1RrjrJUrGQK4Pah8FGj8JvngJqjKKqe").roles("CUSTOMER");
    auth.inMemoryAuthentication().withUser("ramesh").password("$2a$10$SgFbtmR9u3SXkvR7rQrdbuJYhtJuHNqplB2uoVS.shEtSGkb2ibmS").roles("MANAGER");
}
```

step3) Run the application

note: always recomanded to work with encoded password to manage the passwords with strong encyption.

Do i need to need encode the passwords manually as shown above in the real projects?

ans) Definitely not.. As part of user registration logic we include our choice encoder to get encoded password for the given password that will be saved db table..

Working with jdbcAuthentication that uses spring JDBC based DB s/w as AutheticationInfo Provider

==========

step1) make sure that following db tables are avaiable in any db s/w like oracle

parent table having user details

```
CREATE TABLE "SYSTEM"."USERS"
   (    "UNAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,
        "PWD" VARCHAR2(70 BYTE),
        "STATUS" NUMBER(1,0),
         CONSTRAINT "USERS_PK" PRIMARY KEY ("UNAME"));
```

child db table having roles info

```
CREATE TABLE "SYSTEM"."USER_ROLES"
   (    "ROLE" VARCHAR2(20 BYTE),
        "UNAME" VARCHAR2(20 BYTE),
         CONSTRAINT "FK1" FOREIGN KEY ("UNAME")
          REFERENCES "SYSTEM"."USERS" ("UNAME") ENABLE));
```

=>Taking user details and roles details in single db table is bad pratice .. This approach do not support one user having multiple roles so prefer two db tables having FK relationship

(parent db table)

USERS

(PK) UNAME	PWD	STATUS
1 raja	$2a$10$Sws4eLbLoDoUZjC26QAa8ujWzquLfknQwZDkAtXI8f84gYhwe0ZvS	1
2 rajesh	$2a$10$VG.cYdU3TK47RXraenU6POAjlPyt7iJoZ0KiDz1vUCNZYWgIUhiyi	1
3 jani	$2a$10$X3Pay9m0duag8wXCqn.4FO8xGz8Kbktbkj0wWBnnPUp7Gd.Rn635.	1
4 suresh	$2a$10$uHrUdPt6kGK.c2vuYXlOYO2r8Syh2SjhpDhKSKVfpPHrU97fcMe/2	1

indicates boolean value

0 --> user is blocked/disabled
1--->user is enabled/active

(child db table)

USERS_ROLES

ROLE	UNAME (FK)
1 MANAGER	rajesh
2 CUSTOMER	jani
3 MANAGER	jani
4 VISITOR	suresh
5 CUSTOMER	raja

here db table names and col names are user-defined...

step2) add spring jdbc , oracle driver staters to project as additional staters..

Right click project --->spring -->add starters --->select JDBC API,Oracle

step3) add jdbc properties in application.properties file

in application.properties

```
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=manager
```

These entries are useful while creating DataSource obj pointing to oracle Db s/w throug AutoConfiguration process.

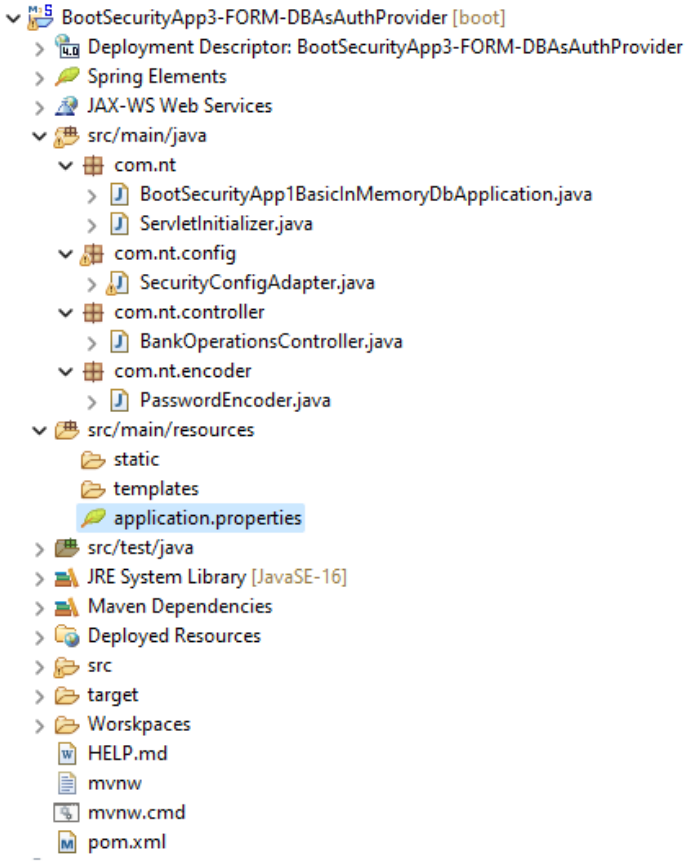
step4) Inject DataSource object to SecurityConfig class

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private DataSource ds;
```

step5) write the following code in 1st configure(-) method to enable jdbc Authentication

```
@Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {

        auth.jdbcAuthentication().dataSource(ds).passwordEncoder(new BCryptPasswordEncoder())
        .usersByUsernameQuery("SELECT UNAME,PWD,STATUS FROM USERS WHERE UNAME=?")  //for autheentication
        .authoritiesByUsernameQuery("SELECT UNAME,ROLE FROM  USERS_ROLES   WHERE UNAME=?"); //for authorization
    }
```

step6) run the application...



41.NTSPBMS615-Spring Boot Security with Spring data JPA -aug21st NTSPBMS615-Spring Boot Security with Spring data JPA -aug22nd

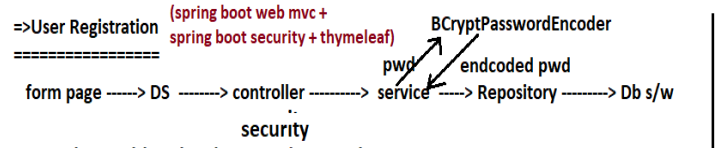
Spring boot security with Spring data JPA (using DB s/w as Authenticatin Provider)
==========

=> There is no direct provision to work spring data JPA or spring ORM based authentication info provider i.e we need to implement most of the logics manually as we do in other spring boot layer apps by taking seperate repository interfaces ,service classes, model classes and etc.. (Actually we should UserDetails(I) impl class having service, persistence logics)

auth.inMomeoryAuthentication() , auth.jdbcAuthentication() .. like this there is no direct template to work with spring data jpa (orm) based Authentication provider

Since the indurstry standard Persistence logic env.. in spring boot apps is spring data jpa .. so it it recomanded to use the same in spring boot security apps.

=>User Registration (spring boot web mvc + spring boot security + thymeleaf)
==========



step1) keep old spring boot project ready

step2) Add the following addtional starters and also remove few exisin starters

=>thymeleaf stater
=>thymeleaf extras starter
=>oracle driver
=> spring data jpa
=> lombok

remove spring jdbc stater
tomcat jaspser
(becozx we are using thymeleaf as the view comp)

=>jsp pages are heavy weight becoz they internally create lots of unnecessary objs.. so use the ligth weight thyme leaf as alternate

thymeleaf= html + dynamic data

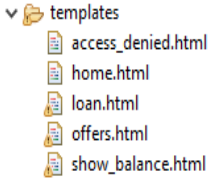
To get currently logged usename from sprin boot security env use the following code in jsp page

<%=SecurityContextHolder.getContext().getAuthentication().getName() %>

```
<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-security</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>
```

```
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <scope>runtime</scope>
                <optional>true</optional>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-tomcat</artifactId>
                <scope>provided</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </dependency>
        <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-test</artifactId>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>com.oracle.database.jdbc</groupId>
                <artifactId>ojdbc8</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>
                    <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-thymeleaf</artifactId>
            </dependency>

            <!-- https://mvnrepository.com/artifact/org.thymeleaf.extras/thymeleaf-extras-springsecurity5 -->
<dependency>
  <groupId>org.thymeleaf.extras</groupId>
  <artifactId>thymeleaf-extras-springsecurity5</artifactId>
</dependency>

            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <optional>true</optional>
            </dependency>
```

step3) convert existing jsp pages to thyme leaf html pages and move them to src/main/resoucces/template folder and also remove view resolver cfgs from application.properties file



step4) Create the Model class

```
//UserDetails.java      (Designed having Collection Mapping)
package com.nt.model;

import java.util.Set;

import javax.persistence.CollectionTable;
import javax.persistence.Column;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.Table;

@Table(name="SECURITY_USERS")
@Entity   @Data
public class UserDetails {

@GeneratedValue(strategy = GenerationType.AUTO)
@Id
private Integer  unid;          uid is reserved keyword in oracle
@Column(length = 20,unique = true,nullable = false)
private String uname;
@Column(length = 150,nullable = false)
private String  pwd;
@Column(length = 20,nullable = false)
private String email;
private  Boolean  status =true;

Collection Mapping
@ElementCollection(fetch = FetchType.EAGER)
@CollectionTable(name ="SECURITY_ROLES",
                 joinColumns = @JoinColumn(name="USER_ID",referencedColumnName = "unid"))
@Column(name="role")
private Set<String> roles;

}
```

Having collection data for all objects of Entity class seperate child table will created

Security_users (parent table)

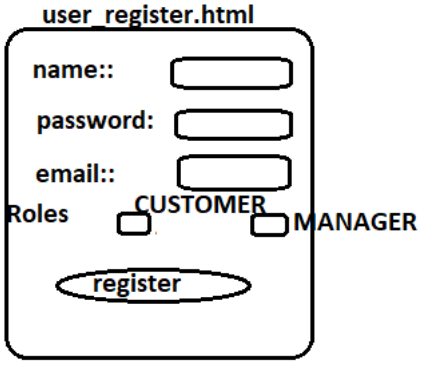
(PK) UID	UNAME	PWD	EMAIL	STATUS
1	raja	rani	r@r.com	1

Security_roles (child table)

ROLE	USER_ID(FK with UID)
CUSTOME	1
MANAGER	1

=>Here colllection mapping concept to maintain set collection values in child table.. if needed u can go One to many Association mapping.

step5) Design form page in thymeleaf having user registration details..



user_register.html (in src/main/resources /template folder)

```
<html xmlns:th="https://thymeleaf.org">
   < form th:action="@{/user/register}"  th:object="${userInfo}" method="POST">
   <table border="0" bgcolor="cyan"  align="center">
     <tr>
         <td>  username :: </td>
         <td>  <input type="text" th:field="*{uname}" /> </td>
     </tr>
      <tr>
         <td> password :: </td>
         <td>  <input type="password" th:field="*{pwd}" /> </td>
     </tr>
     <tr>
         <td> email :: </td>
         <td>  <input type="text" th:field="*{email}" /> </td>
     </tr>
     <tr>
         <td> Roles: </td>
         <td>  <input type="checkbox" th:field="*{roles}" value="CUSTOMER" checked > CUSTOMER
               <input type="checkbox" th:field="*{roles}" value="MANAGER" > MANAGER
          </td>
     </tr>
     <tr>
         <td> <input type="submit"  value="register"> </td>
         <td> <input type="reset"  value="cancel"> </td>
     </tr>

   </table>
    < form>
```

globalpath/reuqest path

model atribute name collected from @ModelAttribute(-) of hnadler method

model class property

In thymeleaf
@ ---> To specify addreses related hyperlinks,forms
$ ---> To read scoped data
* ---> for Bindig data of the form

step6) Develop Repostory Interface for UserDetails Model class

Repo

//IUserDetails.java

```
package com.nt.repository;

import org.springframework.data.repository.CrudRepository;

import com.nt.model.UserDetails;

public interface IUserDetailsRepo extends CrudRepository<UserDetails,Integer > {

}
```

step7) Develop Service Interfaace and Service Impl class using the above Repository

IUserService.java

```
package com.nt.service;

import com.nt.model.UserDetails;

public interface IUserService   extends UserDetailsService{
      public   String register(UserDetails details);
}
```

UserDetailService(I) is pre-defined Intefaface and no way related the project specific model class name and service Interface name

(spring boot security api supplied interface)

If u want to pass Service Impl class obj as the Authentication Info provider in Security Config class first configure(-) method.. that class must implement UserDetailsService(I) directly or indirectly

(org.springframework.security.core.userdetails.UserDetailsService(I))

UserServiceImpl.java

```
package com.nt.service;
```

```
import org.springframework.stereotype.Service;

import org.springframework.beans.factory.annotation.Autowired;

import com.nt.model.UserDetails;
import com.nt.repository.IUserDetailsRepo;

@Service("userService")
public class UserServiceImpl implements IUserService {
      @Autowired
      private IUserDetailsRepo userRepo;
       @Autowired
            private BCryptPasswordEncoder  encoder;
            @Override
            public String register(UserDetails details) {
                  details.setPwd(encoder.encode(details.getPwd()));
            }     return  userRepo.save(details).getUid()+" UserId is registered";
   @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
      return null;
    }
}
```

note:: UserDetails(I) is pre-defined interface avaiable in spring boot security api

org.springframework.security.core.userdetails.UserDetails(I)

method inherited from UserDetailsService(I)

step8) Develop Seperate Controller class for User registration , login activities having global path "/user"

UserController.java

```
package com.nt.controller;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import com.nt.model.UserDetails;
import com.nt.service.IUserService;

@Controller
@RequestMapping("/user")
public class UserController {
      @Autowired
       private IUserService service;

      @GetMapping("/register")  //for form launching
      public  String showUserRegisterForm(@ModelAttribute("userInfo") UserDetails details) {
            return "user_register";
      }

    @PostMapping("/register")
    public  String  registerUser(Map<String,Object> map,
                                 @ModelAttribute("userInfo") UserDetails details) {
        //use service
        String resultMsg=service.register(details);
        map.put("message", resultMsg);
        //return LVN
        return "user_registered_success";

    }

}
```

To handle the form launch

To handle the form submission

note: @ModelAttribute (-) creates Model class obj with given logical name as the attribute name to display initial values in the form comps on form launching ..and to bind form comps data to Model class object on form submission

step 9) Specify current service class as the Authentication Info Provider
also specify
and PasswordEncoder in SecurityConfig class

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserDetailsService service;
    @Autowired
    private BCryptPasswordEncoder encoder;

    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(service).passwordEncoder(encoder);
    }
    ....
    ....
}
```

step10) Provide permitAll() access to "/user/register" url

In SecurityConfig.java

```
@Override
    public void configure(HttpSecurity http) throws Exception {
        //authorize requests
      http.authorizeRequests().antMatchers("/bank/").permitAll() //Not authetication an no authorization
      .antMatchers("/user/register").permitAll()  // No authentication and no authorization on user registration request
       .antMatchers("/bank/offers").authenticated() //only authentication
       .antMatchers("/bank/balance").hasAnyAuthority("CUSTOMER","MANAGER") // authentication + authorization for "CUSTOMER","MANAGER" role uses
                    .antMatchers("/bank/loanApprove").hasAuthority("MANAGER") //// authentication + authorization for
    .anyRequest().authenticated() //remaing all requests url mus be authtenticated

  //specify authentication mode
  //.and().httpBasic()
        .and().formLogin().defaultSuccessUrl("/bank/", true)

    // add remember filter
        .and().rememberMe()

  //add Logout Filter
  //.and().logout()
        .and().logout().logoutRequestMatcher(new AntPathRequestMatcher("/signout"))


  //exception/error handling
  .and().exceptionHandling().accessDeniedPage("/denied")

 // add SessionMaxConcurrency count
        .and().sessionManagement().maximumSessions(2).maxSessionsPreventsLogin(true);
}
```

step11) Add hyperlink in the home page for userRegistration

In home.html

```
<br><br><br>
<a th:href="@{/user/register}"> Register User</a>
```

step12) specify DataSource , ORM properties in application.properties

application.properties

```
#DataSource cfg
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=manager
#ORM properties
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.Oracle10gDialect
```

step13) Develop user_registration_success.html file having thymeleaf code

only tommorrow session starts @ 6:40 pm

user_registration_success.html

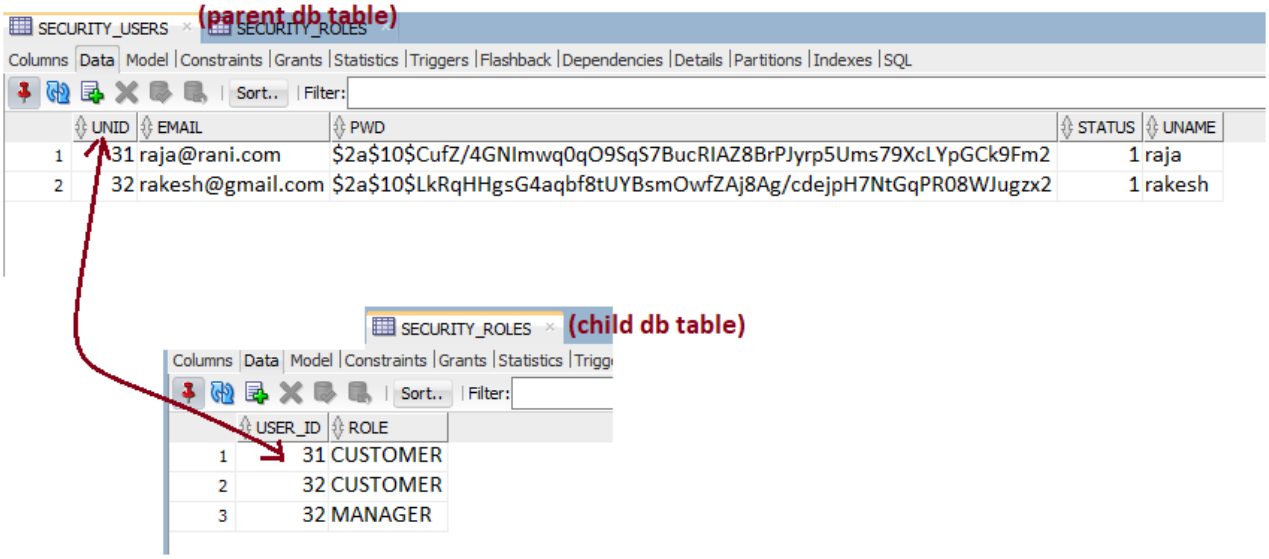
```
<html    xmlns:th="http://www.thymeleaf.org">
<h1 style="color:red;text-align:center"> Registration Success page</h1>
  <h3>       <span th:text ="${resultMsg}"/></h3>
 <br>
 <a th:href="@{/bank/}">home</a>
```

step14) In Main class , configure BCryptPasswordEncode as spring bean using @Bean method

In main class

```
@Bean
      public BCryptPasswordEncoder createPwdEncoder() {
            return new BCryptPasswordEncoder();
      }
```

step15) run the Application



42.NTSPBMS615- Aug 23rd -2022-Spring Boot Security with Spring data JPA

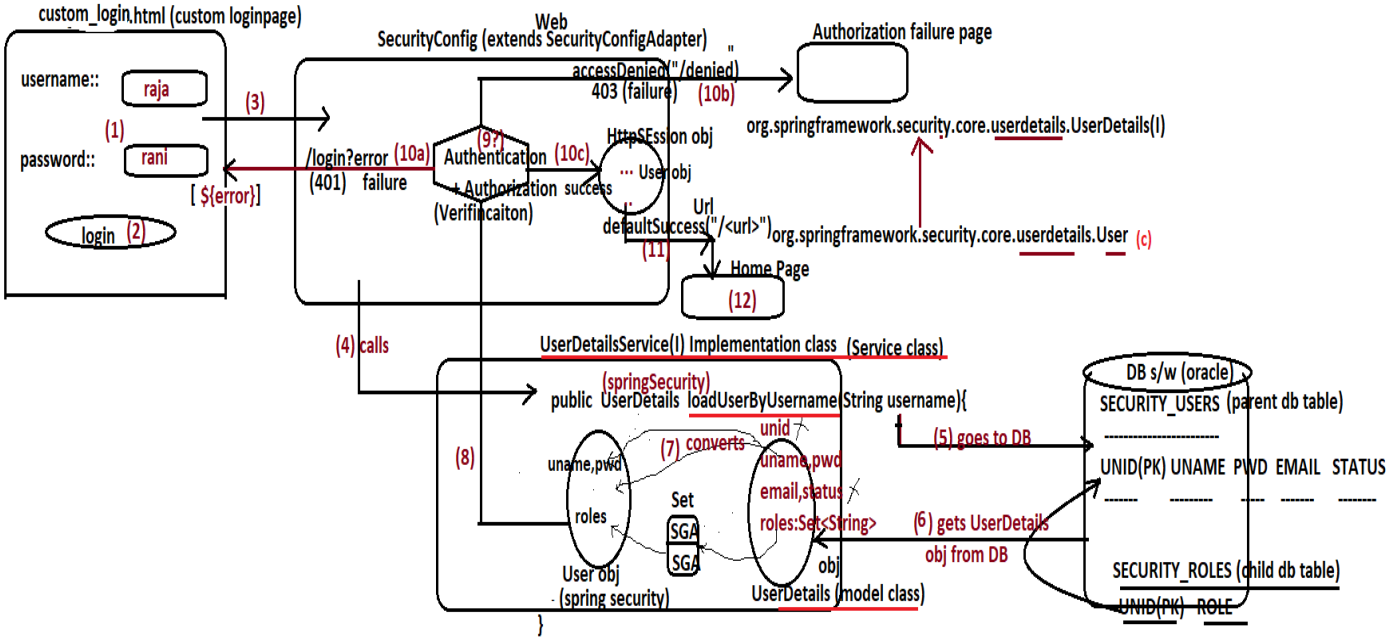
Spring Boot Securty with spring boot data jpa

==========

=> Here we need to configure service class that implements org.springframework.security.core.userdetails.UserDetailsService (I) directly or indirectly as Authentication Info Provider in SecurityConfig class

=>When give POST + /login request to spring boot security App , the SercurityConfig class sends to request to loadUserByUsername(-) method of the above service class ... that method returns org.springframework.security.core.userdetails.User obj that implements org.springframework.security.core.userdetails.UserDetails(I) having current logged user details like username, password, roles and etc.. to SecurityConfig class for validation/verification.. if the verification success then it keeps login details in HttpSession and show defaultSucessUrl page otherwise login error page.

shows



org.springframework.security.core.GrantedAuthority (I) (GA) (c)

↑ extends

org.springframework.security.core.authority.SimpleGrantedAuthority (c) (SGA)

Adding custom login page to the Project and performing Authentication + Authorization Using Spring data JPA UserDetails Service

==========

step1) First understand different default urls

/login + GET :: To show default form based authentication login page

/login + POST ::To process default form based authentication login page submission

/login?error + 401 status code :: if authentication fails

<url> + 403 status code :: if authroization fails

/login?logout :: default logout success url

Here /login is fixed as logn as u want to use the default Authentication + Authorization manager

step2) add Handler method in UserController class to show custom login page

In UserController.java

```
@RequestMapping("/showLogin")
public String showLoginPage() {
    return "custom_login";
}
```

step3) In SecurityConfig class (In 2nd configure(-) method)

-->add permitAll() for "/user/ShowLogin"

--> add "/user/ShowLogin" as loginPage(-) url

In SecuirtyConfig class

```
public void configure(HttpSecurity http) throws Exception {
        //authorize requests
      http.authorizeRequests().antMatchers("/bank/").permitAll()  //Not authetication an no authorization
      .antMatchers("/user/register","/user/showLogin").permitAll()
      .antMatchers("/bank/offers").authenticated()  //only authentication
      .antMatchers("/bank/balance").hasAnyAuthority("CUSTOMER","MANAGER")  // authentication + authorization for  "CUSTOMER","MANAGER" role uses
                  .antMatchers("/bank/loanApprove").hasAuthority("MANAGER")  //// authentication + authorization for
   .anyRequest().authenticated()  //remaing all requests url mus be authtenticated

      //specify  authentication mode
      //.and().httpBasic()
            .and().formLogin().defaultSuccessUrl("/bank/", true).loginPage("/user/showLogin")
                                  shows  home page
                                   on successful login                request path of handler method
                                                                      to show the custom login page
        // add remember filter
            .and().rememberMe()

      //add Logout Filter
      //.and().logout()
            .and().logout().logoutRequestMatcher(new AntPathRequestMatcher("/signout"))


      //exception/error handling
     .and().exceptionHandling().accessDeniedPage("/bank/denied");

   // add  SessionMaxConcurrency count
            .and().sessionManagement().maximumSessions(2).maxSessionsPreventsLogin(true);
  }
```

step4) develop custom_login.html as the customer login page in main/java/resources/templates folder.

=>text box names should be username,password
=> action url should be "/login"

These are fixed as long as
we want the spring boot security's
readymade AuthenitcationManager, Authorization Manager.
performing security operations

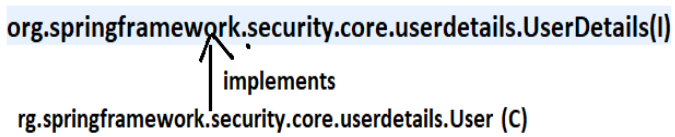
custom_login.html

```
<html xmlns:th="https://thymeleaf.org">

 <h1 style="color:blue;text-align:center">  Login Page </h1>

<form th:action="@{/login}"  method="POST">
  <table border="0" bgcolor="cyan"  align="center">
    <tr>
        <td>  username :: </td>
        <td>  <input type="text"  name="username" /> </td>
    </tr>
     <tr>
        <td> password :: </td>
        <td>  <input type="password" name="password" /> </td>
    </tr>
    <tr>
        <td> <input type="submit"  value="Login"> </td>
        <td> <input type="reset"  value="cancel"> </td>
    </tr>
  </table>
</form>
```

step5) Declare the following finder method in repository Interface to get UseDetails model class obj based on the given username

```
public interface IUserDetailsRepo extends CrudRepository<UserDetails,Integer > {

    public  Optional<UserDetails>  findByUname(String uname);   custom  methods
}
```

step6) write fillowing logic in loadUserByUsername(-) in UserDetailsService(-) implementation to get Model class obj(UserDetails obj) and to convert into User class obj (spring secuirty api class object)



In service class

```
@Override
    public org.springframework.security.core.userdetails.UserDetails loadUserByUsername(String username)
            throws UsernameNotFoundException {
        // get Model class object (com.nt.model.UserDetails )
        Optional<com.nt.model.UserDetails> opt=userRepo.findByUname(username);
        if(opt.isEmpty())
            throw new  IllegalArgumentException("user not found ");
        else {
            com.nt.model.UserDetails  details=opt.get();
        /*   //convert Set<String> type roles  to  Set<SGA> type roles
             Set<GrantedAuthority>  roles=new HashSet();
             for(String role:details.getRoles()) {
                 SimpleGrantedAuthority authority=new SimpleGrantedAuthority(role);
                  roles.add(authority);
                      }
            //convert  Model class object(com.nt.model.UserDetails obj) to  Spring secuirty User(org.springframework.security.core.userdetails.User) object
             User  user=new User(details.getUname(),details.getPwd(),roles); */
            User user=new User(details.getUname(),
                               details.getPwd(),
                               details.getRoles().stream().map(role-> new SimpleGrantedAuthority(role)).collect(Collectors.toSet()) );
                                                    LAMDA Expression + stream api  to covertn Set<String> to Set<SGA>
        return user;
        }//else
    }//method
```

step7) specify multiple urls related form login authentication in Spring security config class..

In security config class

```
//specify  authentication mode
        //.and().httpBasic()
                .and().formLogin()
                .defaultSuccessUrl("/bank/", true)  //HOME page url
                .loginPage("/user/showLogin")  //for  GET mode request to launch form page
                .loginProcessingUrl("/login")  // for POST mode request to submit and process the request
                .failureUrl("/user/showLogin?error")  //  Authentication failed url
                                                      fixed req param
           // add remember filter
                .and().rememberMe()
        //add Logout Filter
        //.and().logout()
                .and().logout()
                .logoutRequestMatcher(new AntPathRequestMatcher("/signout"))
                .logoutSuccessUrl("/user/showLogin?logout")  // after logout url
                                                        fixed req param
```

step8) In form form page read and display "error", "logout" req param value

In custom_login.html

```
<span style="color:red;text-align:center" th:if="${param.error}">Invalid Login detials (authentication failed)</span>
<span style="color:red;text-align:center" th:if="${param.logout}">User Logged succefully </span>
```

step9) Run the application..

What CSRF problem?

CSRF :: Cross Site Requesting Forgery

(pishing)

=> It fishing or hacking techniue hacker or attacker ...who makes the innocent enduser sending his data to user sites and accounts .

eg:: send spam emails , trapping emails ane etc..

```
<form action="https:///icicibank.com/trasferFunds">
    <input type="hidden" name="amount" value="10000">
    <input type="hidden" name="srcAccount" value="4455666">
    <input type="hidden" name="destAccount" value="6567788">
<input type="submit" value="Click here to winthe lottery">
```

43.NTSPBMS615- Aug 24th -2022-Spring Boot Security -CSRF Problem and Solution

What CSRF problem?

CSRF :: Cross Site Requesting Forgery

(pishing) of

=> It fishing or hacking techniue hacker or attacker ...who makes the innocent enduser sending his data to user sites and accounts .

ing

eg:: send spam emails , trapping emails and etc..

For different website the attacker make victim to send following details by showling lottery ticket benifit.

```
<form action="https:///icicibank.com/trasferFunds" method="POST">
   <input type="hidden" name="amount" value="10000">
   <input type="hidden" name="srcAccount" value="4455666">
    <input type="hidden" name="destAccount" value="6567788">
  <input type="submit"  value="Click here to winthe lottery">
```

(or)

```
<a href="https://icicibank.con/trasnferfunds?amount=10000&srcAccount=4455666&destAccount=6567788">
        Click here to win the lottery
</a>
```

Feeling CSRF Problem Pratically

==========

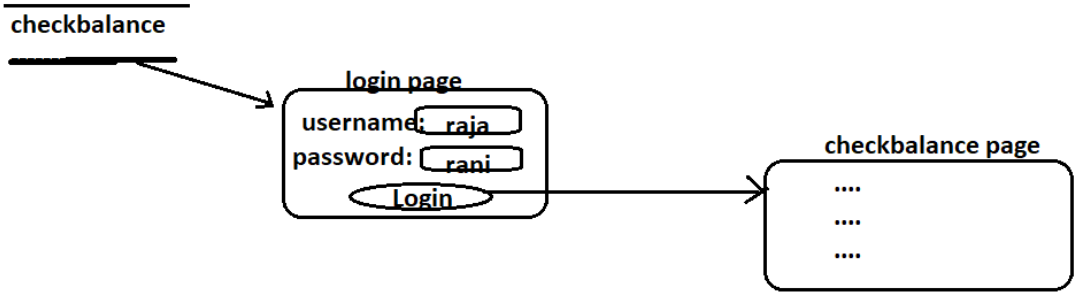
step1) Disable csrf protection in spring boot security Application

In SecurityConfig .java

```
// disable csrf protection
        http.csrf().disable();      // By default csrf protection is enabled
```

step2) Run Spring Boot Securirty original Application .. complete authentication and to use one or two services (like check balance service by using "raja" user) and do not logout

http://localhost:3030/BootSecurity03-FORM-DB-SpringDataJPA/bank/



step3) create another normal Dynamic web project having the following code in any html page

index.html

```
<form action="http://localhost:3030/BootSecurity04-FORM-DB-SpringDataJPA-CSRF/bank/balance"
method="POST">
      <input type="submit"  value="Click here to win lottery"/>
   </form>
```

step4) Run new Application along with old App using r old browser window and feel the csrf or pishing problem by clicking othe hyerlink

Solving csrf Problem

=>GET mode rquest is read only request i.e it does not any thing in the server where as POST mode request is Update request i.e they change data of the server env..

=>csrf problem can be solved only for the request urls that are expecting POST requests.

step1) make csrf is not disabled in SecurityConfig class

```
//http.csrf().disable();
        //By default csrf potection is
        enabled in spring boot
```

step2) add csrf related token to the login form page and to the other form page by taking the support hidden box

In custom_login.html

```
<input type="hidden" th:name="${_csrf.parameterName}" th:value="${_csrf.token}"/>
```

step3) In Another web application's index.html (any html page) try to send the request

```
<form action="http://localhost:3030/BootSecurity04-FORM-DB-SpringDataJPA-CSRF/bank/balance"
method="POST">
    <input type="submit" value="Click here to win lottery"/>
</form>
```

Make sure this URL based handler method is @PostMapping in the controller class as shown below

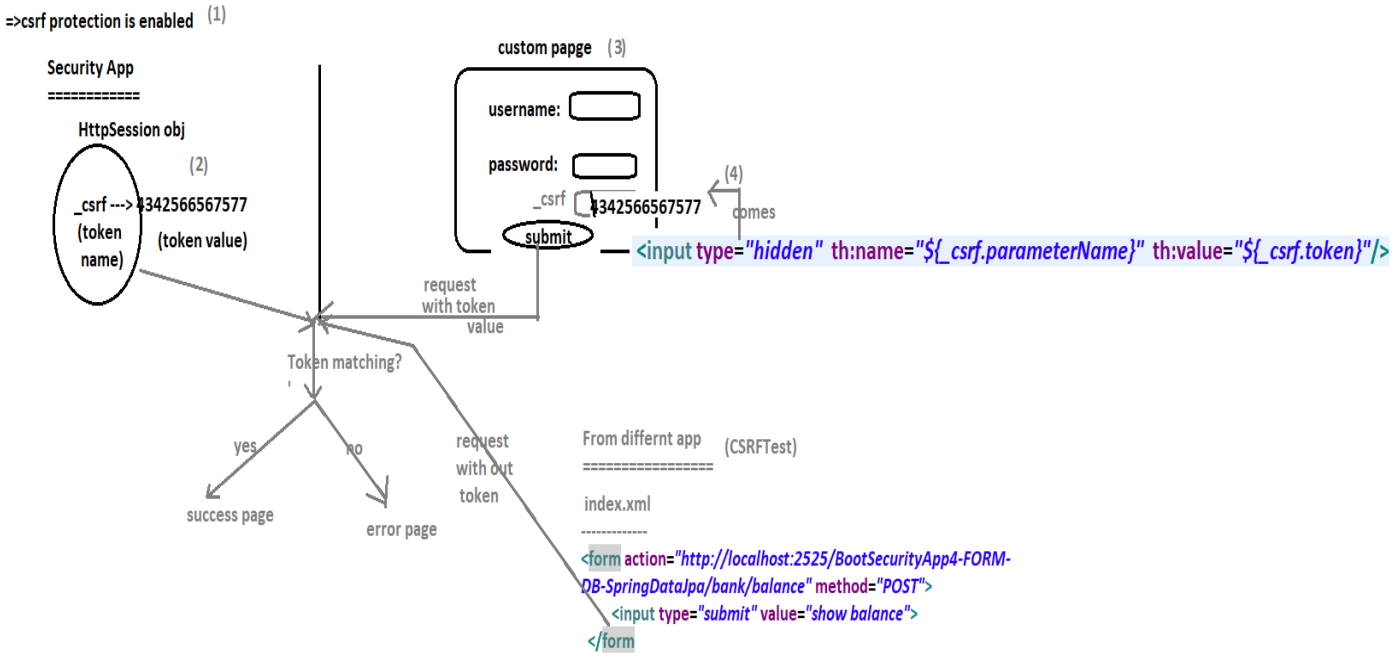
```
@PostMapping("/balance")
public String checkBalnace(Map<String,Object > map) {
    map.put("balance_amt",new Random().nextInt(1000000));
    return "show_balance";
}
```

How does csrf protection works internally

=>when csrf protection is enabled (it is by default spring secuirty /spring boot security app) one session token will be created as session attribute having "_csrf" as token name and and "32" digits hexa decimal number token value.. Using the following hidden box

```
<input type="hidden" th:name="${_csrf.parameterName}" th:value="${_csrf.token}"/>
```

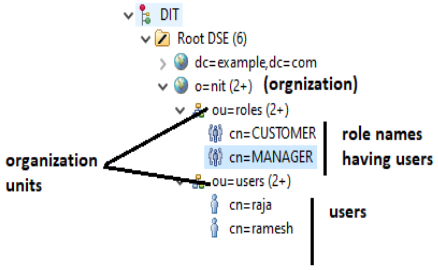
we try to get crsf token name and token value to the form page and we send them along with form submission.. The Security env... of server side takes token name and value coming from browser and validates with already avaiable session token value .. if matched furhter activities will be allowed.. if request comes with invalid token or no token then error page will be shown



Configuring LDAP Server as Authentication provider in spring boot securirty App

step1) Install apache Directory studio and LDAP Server and creates users having roles by following given pdf document

Spring-security LDAP-converted.pdf (uploaded FB Group)



Apaache Directory Studio s/w installation allows to creates Apache DS server nothing but LDAP server..

step2) Try go gather LDAP Server url and admin user name and password

ou=system , uid=admin ,userPassword=secret

ldap://localhost:10389

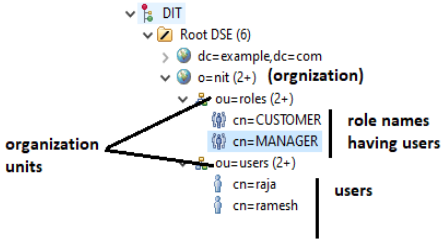
44.NTSPBMS615- Aug 25th -26th-2022-Spring Boot Security - Working with LDAP Server

Configuring LDAP Server as Authentication provider in spring boot securirty App

==========

step1) Install apache Directory studio and LDAP Server and create. users having roles by following given pdf document

Spring-security LDAP-converted.pdf (uploaded FB Group)



Apaache Directory Studio s/w installation allows to creates Apache DS server nothing but LDAP server..

LDAP :: Light weight directory access protocol

note:: The Speciality of LDAP server , once we forgot the password .. we can not get the password except resetting the password.

note:: we can create users and roles in LDAP server either using manual process or using Open LDAP java api

step2) Try go gather LDAP Server url and admin user name and password

gather from LDAP browser | ou=system , uid=admin ,userPassword=secret

ldap://localhost:10389

gather from log messsages

On Ldap api of spring env..

refer

http://ldaptemplate.sourceforge.net/quickstart.pdf

[Spring Ldap api gives LdapTemplate class obj through AutoConfiguration having connectivity with LdapServer using which we can create users, list users, delete users and etc..]

step3) keep any spring Boot security App ready

(take InMemory Auth application's copy as seperate project)

step4) add the following dependency (jar file) to pom.xml

```
<!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-ldap
-->
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-ldap</artifactId>
</dependency>
```

step5) make sure that Apache directory studio's Apache DS server is in running mode

step6) write the following in the 1st confugure(-) SecurityConfig class to make LDAP server as Authentication provider..

SecurityConfig.java

==========

```
package com.nt.config;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.session.SessionInformationExpiredEvent;
import org.springframework.security.web.session.SessionInformationExpiredStrategy;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
import org.springframework.security.web.util.matcher.RequestMatcher;
```

```
@Configuration
@EnableWebSecurity   // makes the Normal @Configuration class Spring Security Configuration class
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.ldapAuthentication().contextSource().url("ldap://localhost:10389/o=nit")
        .managerDn("uid=admin,ou=system").managerPassword("secret")   //for connecting LDAP server
        .and()
        .userSearchBase("ou=users").userSearchFilter("(cn={0})")  //for authentication
        .groupSearchBase("ou=roles").groupSearchFilter("(uniqueMember={0})")
        .groupRoleAttribute("cn").rolePrefix("ROLE_");  // For Authorization
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        //authorize requests
       http.authorizeRequests().antMatchers("/").permitAll()  //Not authetication an no authorization
       .antMatchers("/offers").authenticated()  //only authentication
       .antMatchers("/balance").hasAnyRole("CUSTOMER","MANAGER")  // authentication + authorization for  "CUSTOMER","MANAGER" role uses
                               .antMatchers("/loanApprove").hasRole("MANAGER")  //// authentication + authorization for
                                 .anyRequest().authenticated()  //remaing all requests url mus be authtenticated

        //specify  authentication mode
                .and().formLogin()
          // add remember filter
                .and().rememberMe()

        //add Logout Filter
        //.and().logout()
                 .and().logout().logoutRequestMatcher(new AntPathRequestMatcher("/signout"))


        //exception/error handling
       .and().exceptionHandling().accessDeniedPage("/denied")

      // add  SessionMaxConcurrency count
                .and().sessionManagement().maximumSessions(2).maxSessionsPreventsLogin(true);


    }
}
```

we can collect all thi info from apache directory studio

In security in spring apps

->using spring boot security
->using JWT
->using oauth 2.0
->using okta

will be learned after spring rest

step7) run the Application

note:: Instead of installing LDAP server seperately as Apache Directory studio .. we can also get it as eclipse plugin .. try this

eclipse LDAP plugin link steps:
https://directory.apache.org/studio/installation-in-eclipse.html

---

(spring security 5.7.0)
org.springframework.security.config.annotation.web.configuration.~~WebSecurityConfigurerAdapter~~ is deprecated
So alternate is developing SecurityConfig class as @Configuration with out extending from WebSecurityConfigurerAdapter
i.e In SecurityConfig class , we take all neccessary configurations of authentication auhorization in the form of @Bean methods
(this alternate is called work with SecurityFilterChain object)

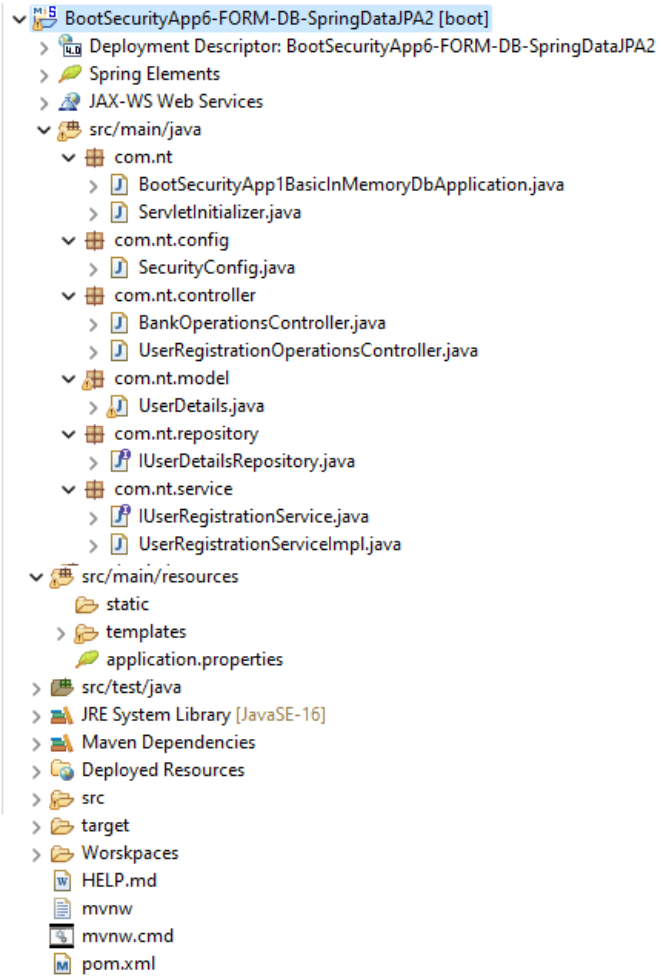
step1) keep Spring Boot Security App ready that uses SpringDataJpa as the Authentication Info Provider
(Here we using FORM Authentication with the support of Spring data JPA based Authentication Info Provider)
step2) Modify the Security Configration class show below with out extending from WebSecurityConfigurerAdapter

```
//SecurityConfig.java
package com.nt.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import com.nt.service.IUserRegistrationService;

@Configuration
public class SecurityConfig {
        @Autowired
         private IUserRegistrationService  userService;
        @Autowired
         private  BCryptPasswordEncoder encoder;


        @Bean
        public DaoAuthenticationProvider authenticationProvider() {
          DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();

          authProvider.setUserDetailsService(userService);
          authProvider.setPasswordEncoder(encoder);

          return authProvider;
        }
```

Configuration of UserDetailService(I) style spring data jpa env.. AuthenticationProvider.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

    //autheticate and authorize urls
    http.authorizeRequests().antMatchers("/bank/").permitAll()
                            .antMatchers("/user/register","/user/showLogin").permitAll()
                            .antMatchers("/bank/offers").authenticated()
                            .antMatchers("/bank/balance").hasAnyAuthority("MANAGER","CUSTOMER")
                            .antMatchers("/bank/approveLoan").hasAuthority("MANAGER")
                            .anyRequest().authenticated()

       // To enable BASIC Authentication (uses thr browser supplied dialog box)
      // .and().httpBasic()
          // To enable FORM based Authentication
       .and().formLogin().defaultSuccessUrl("/bank/",true)
       .loginPage("/user/showLogin")
       .loginProcessingUrl("/login")
       .failureUrl("/user/showLogin?error")
       // Enables  remember ME option in Form Based Authentication
             .and().rememberMe()

       // enable logout
       //.and().logout()
       //enable logout with custom url
       .and().logout().logoutRequestMatcher(new  AntPathRequestMatcher("/bank/signout"))
       .logoutSuccessUrl("/user/showLogin?logout")
```

```
                // cofigure custom error page for 403 error
                .and().exceptionHandling().accessDeniedPage("/bank/denied")
                // session max concurrency control
                .and().sessionManagement().maximumSessions(2).maxSessionsPreventsLogin(true);

                //specify AuthenticationProvider
            http.authenticationProvider(authenticationProvider());
            return http.build();    //gvies SecurityFilterChain object having both AuthenticationManager,
                                      AuthenticationProvider Information
        }
}
```



45.NTSPBMS615- OAuth2.x Intro -- Aug 27th-2022

Spring Security With OAuth2.x

==========

(OAuth :: Open Authorization)

=>Oauth 2.x is an open standard and framework for providing 3rd party application services to Client Apps i.e security to Client Apps using third party Applications.

=> Our Projects or web application behaves like Client Apps trying to use the services of Thrid party Applications for security while

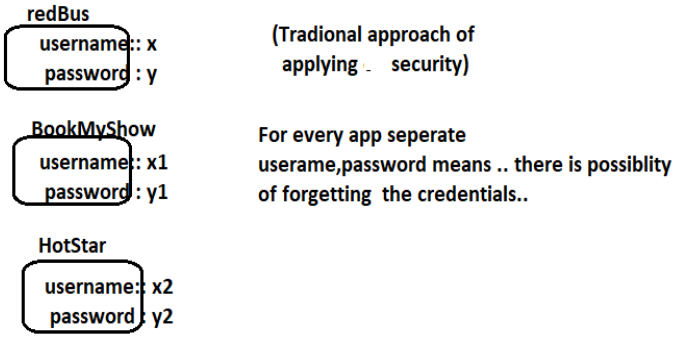
=>Examples for Client Apps are swiggy , zomoto , redbus, abhibus, MakeMyTrip , IMobile, shadi.com, BookMyShow , amazon prime , zerodha , netflix ,zepto and etc.

=> The third Party Applications are technically called Authorization & Resource Serves.

=>Examples for Authorization and Resource Servers are like Facebook ,gmail, twitter , Instagram , linkedin, GitHub and etc..

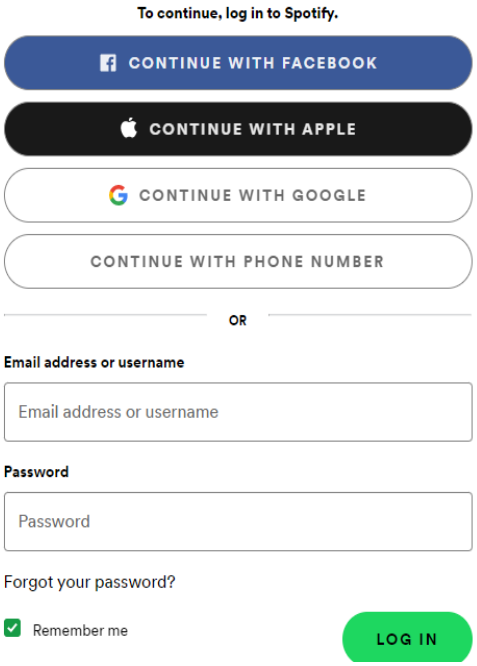
=> Different Client Apps that are listed above tries to use Third party Applications that are listted above for simplifying Login and Authentication activities

Problem::

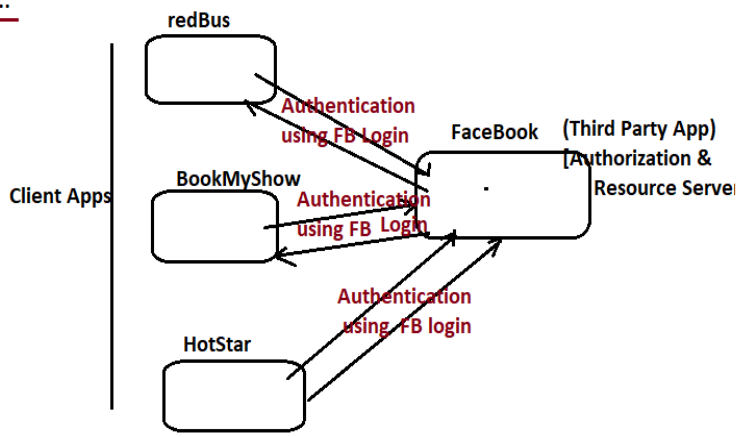


(Tradional approach of applying security)

For every app seperate userame,password means .. there is possiblity of forgetting the credentials..



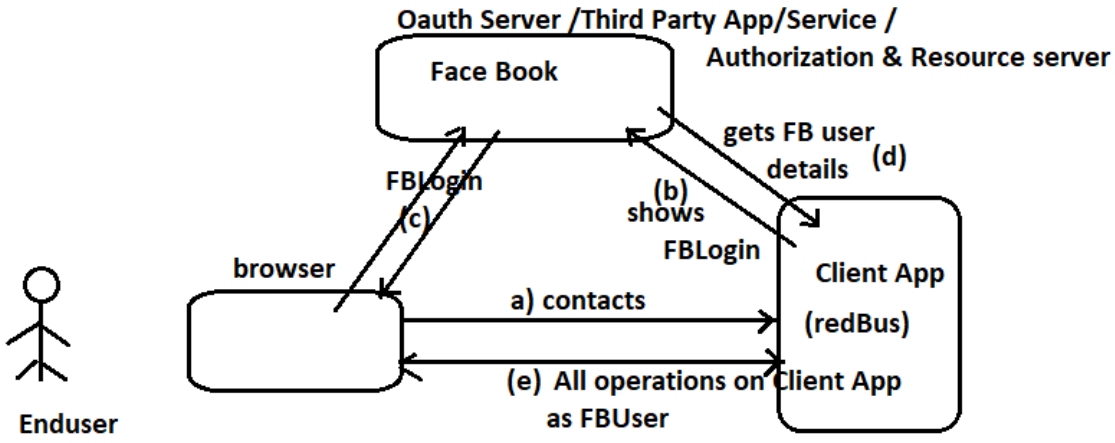
Solution ::



Oauth 2.x supports SSO (Single Singon Feature) i.e by login to one third party App FB , Google and we can login multiple other Apps like redbus , bookmyshow and e

=> by login to gmail account we can start accessing youtube , google drive , google google services..

Basic Architecture diagram of OAuth2.x



Oauth 2.x features

=> It is very popular open standard for securing Apps
=> It can be used and implemented in any techinal domain like java ,.net and etc..
=> Spring boot gives lots of abstraction towards using and implementing oauth 2.x service
=> Very much impleneted in Day to day activity Apps like tickBooking Apps, e-commerces Apps and etc..
=> Integration of Client App with Outh2.x is very easy towards connecting and using Third party Service
=> This is not much recomanded to financial services Applications like Bank Account creation , Credit card issuing and etc.. becoz some Third party Services like FB contains lots of fake identities.
=> Allows to implement SSO on Applications as discussed above
=> if we enable oauth 2.x based Login activity in our Client App then there is no need of implementing Ldap server concepts seperately in client app development.

[if oauth 2.x is used fully in our Client Apps development then
there is no need of implementing spring security forms,
spring security ldap , spring security with JWT seperately)

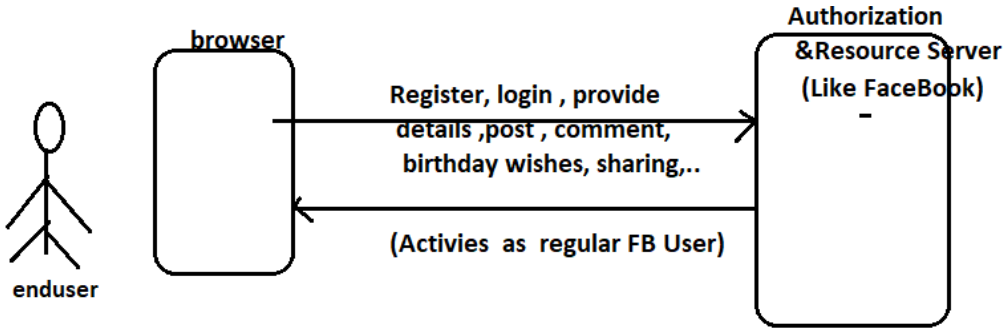
All Thri rd party Apps or services like FB ,gmail provides two ways of interactions

a) Direct interaction for enduser
[To use the services of Third Apps directly as enduser --> eg:: we using FB directly]

b) Interaction as developer
[To make other Client Apps like redBus using Third Party App services]
eg:: Oauth 2.x impl

Oauth 2.x Implementation

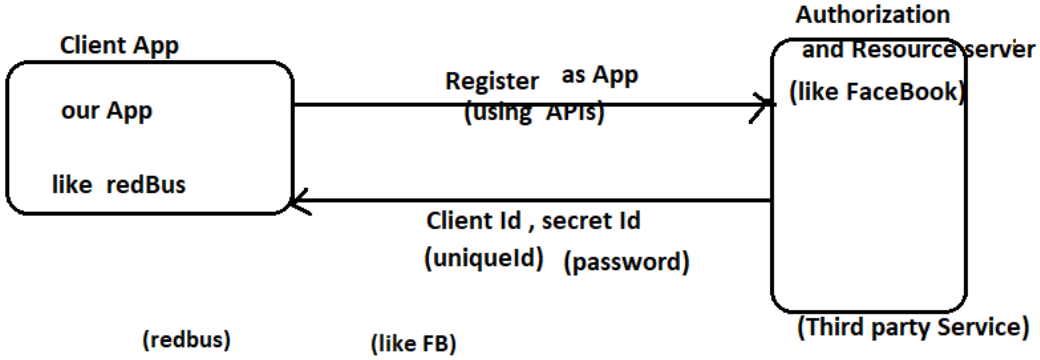
1) Register and provide user Details with Authorization and Resource Server..
(create account in FB as enduser)



(Here we need to do some cfgs in Third party app as developer)

2.) Register our Client App (redbus) with Auhorization and Resource server (Third Party App or Service) (like FB) to get Client ID and SecretId

[For this we need to use Third Party App or service provided API as Developer)



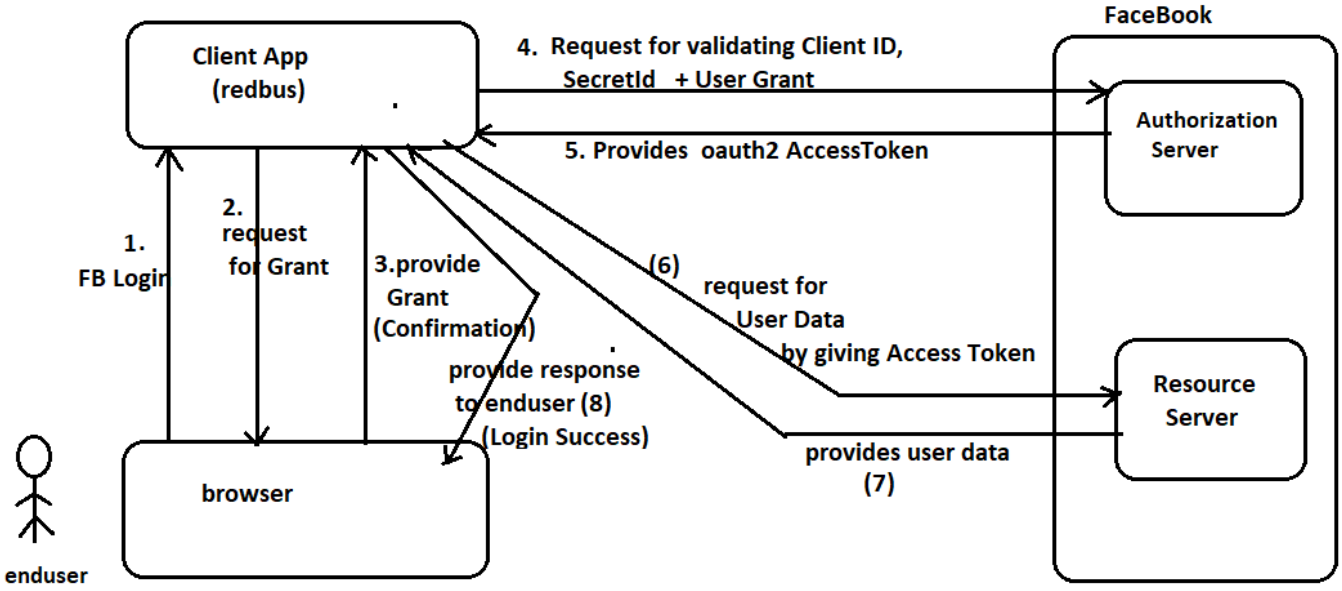
3) Valid Enduser and Client App (redbus) ate in Authorization and Resource server (like FB)

FB , Gmail , Linked In and etc.. maintains two parts

a) Authorization Server :: Contains authentication logic (FB Login)

b) Resource Server :: gives access to various details of authenticated user (getting FB User details)

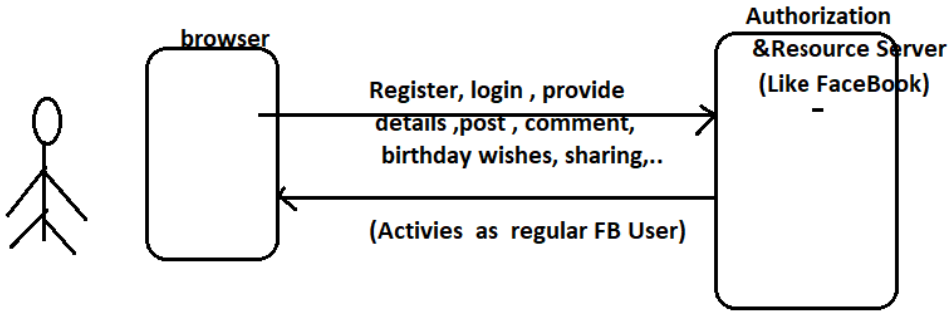
like friends info,posts info, user dob and etc..)



46.NTSPBMS615- OAuth2.x Implementation -- Aug 30th-2022

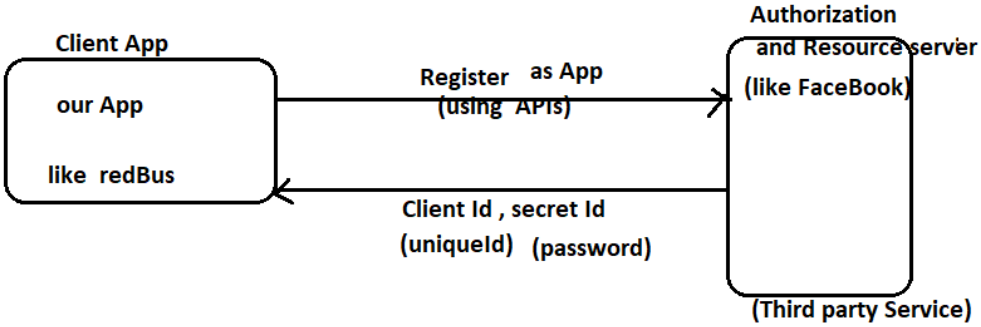
Oauth 2.x Implementation
==========

1) Register and provide user Details with Authorization and Resource Server..



2.) Register our Client App with Auhorization and Resource server (Third Party App or Service) to get Client ID and SecretId

[For this we need to use Third Party App or service provided API as Developer)

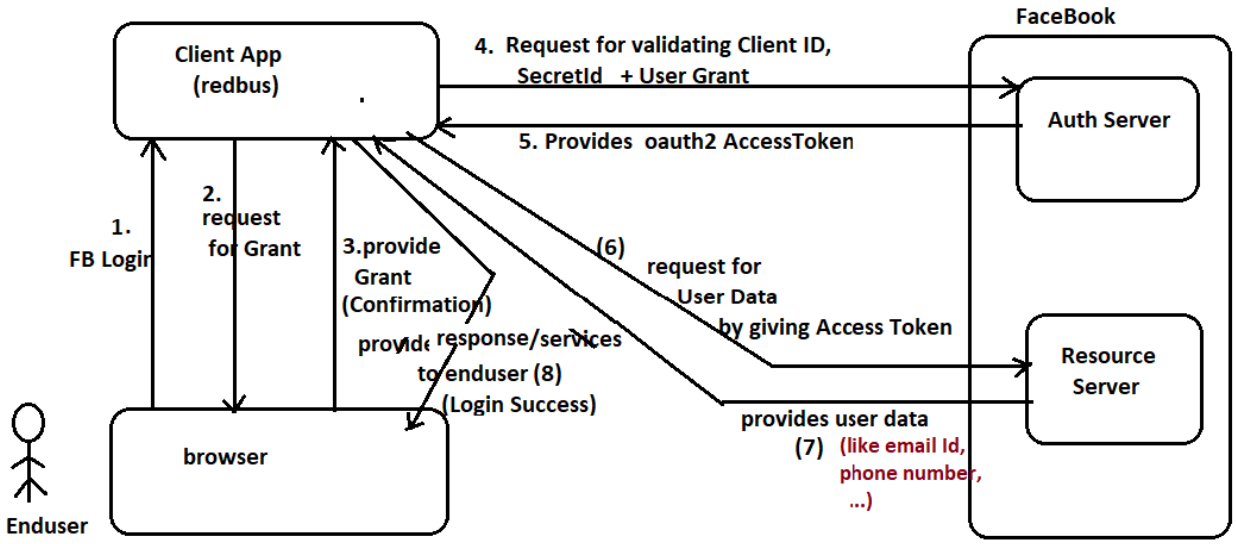


3) Valid ate Enduser and Client App in Authorization and Resource server

FB , Gmail , Linked In and etc.. maintains two parts

a) Authorization Server :: Contains authentication logic (FB Login)

b) Resource Server :: gives access to various details of authenticated user (getting FB User details)



with respect diagram of step3

---

(1) In browser that is pointing client App screen (redbus screen) enduser clicks on FB Login

(2) The Client App (redbus) makes you provide login Credentials of FB to provide permission to use FB user details (request for access grant)

(3) Once we complete FB Login and "continue as certain <user> " we can say Access Granted (Permission provided)

(4) Client App(red Bus) goes to Auth Server of FB for Client App , user details Validation by carrying Client id, scretId , login details

(5) FB Auth Server validates the Client App and provides one Access Token (ID) which is valid for current server user and current Client App to perform certain operations in the Resource (if enduser tries to change it then it will not work)

(6) Client App (redbus) makes a request Resource Server of FB having that access Token (ID)

(7) Resource Server of FB validates the access Token provides the required and permitted UserData to Client App

(8) Finally Client App gets UserData and uses that data to provide various services to Enduser.

Registering our web application (Client App) with FB to get Client Id and SecretId

a) Go to FB developers url (https://developers.facebook.com)
(if not logged into FB account .. u need to login now)

b) Create FB Application by clicking "create Application Button"

FB Developers home page ---> my aps ---create App

c) Choose "Bussines" Type App --->next

d) provide the following details

App name :: BootMsApp1
email Id :: _____

create App ---> submit the FB password

e) Gather Client App/App Id , secret Id of the App ..

Dashboard ----> settins ---> basic --->

App ID: 651124969397572 (like userame)

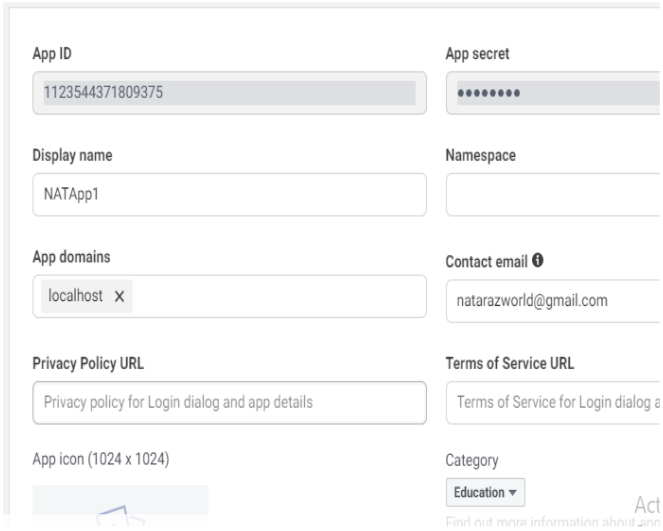
App secret: 984d4ab13983dfacf13315de98eb4046 (like password)

Oauth Security Implmentation using spring boot *(developing single sign on applicaiton using FB)*

step1) make sure that u have FB Account and u logged into that FB Account

step 2) Register Client app (spring boot MVC/spring RestApp) With Auth And Resource Server to collecte Client Id And Secret Id

client id :: 651124969397572
secrete id :: 984d4ab13983dfacf13315de98eb4046



step3) Develop Spring boot Rest/Spring boot MVC application as Client App having spring security Oauth Support.

a) create spring boot starter Project having the following Dependencies
i) spring security , ii) OAuth2 Client ,iii)Spring web iv) thymeleaf

b) add the following entries in application.properties file

application.properties

```
# Applicaiton name
spring.application.name=SpringSecuirtyOauth2.xApp

# server Port number
server.port=4041

# Cfg FB AuthServer Credentials (Client Id , Secret Id)
spring.security.oauth2.client.registration.facebook.client-id=531577848170095
spring.security.oauth2.client.registration.facebook.client-secret=ebbe102f51e507100ea43b3973e5a96b
spring.security.oauth2.client.registration.facebook.scope=email,public_profile
```

c) Develop Security Configuration class.

SecurityConfig.java

```
package com.nt.config;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
```

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
	@Override
	public  void configure(HttpSecurity http) throws Exception {
		http.authorizeHttpRequests().antMatchers("/","/login","/home").permitAll()
		.anyRequest().authenticated()
		.and().formLogin()
		.and().oauth2Login()   //  we develop  custom login form having hyper link to login as FB user.
		.and().csrf().disable();
	}
}
```

d) Develop RestController / MVC Controller Comp

#RestController

```
package com.nt.controller;

import java.security.Principal;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class RedBusUserOperationsController {

	@GetMapping("/home")
	public   String showHome() {
		return " Hello ,  WELCOME  TO HOME PAGE of  RedBus.com";
	}

	@GetMapping("/after")
	public   String  afterLoginPage() {
		return " Hello ,  Successfully Logged into  RedBus.com";
	}

	@GetMapping("/user")
	public Authentication  showUserDetails(Principal  principle) {
		System.out.println("logged in details ::"+principle.getName());
		Authentication auth=SecurityContextHolder.getContext().getAuthentication(); //gives more info
		return  auth;                                                                about logged in user.
	}

}
```

** Principal object hold currently logged user nanme*
** Authentication object holds multiple details currently logged in user like credentials (username, password) , addtional details and etc..)*

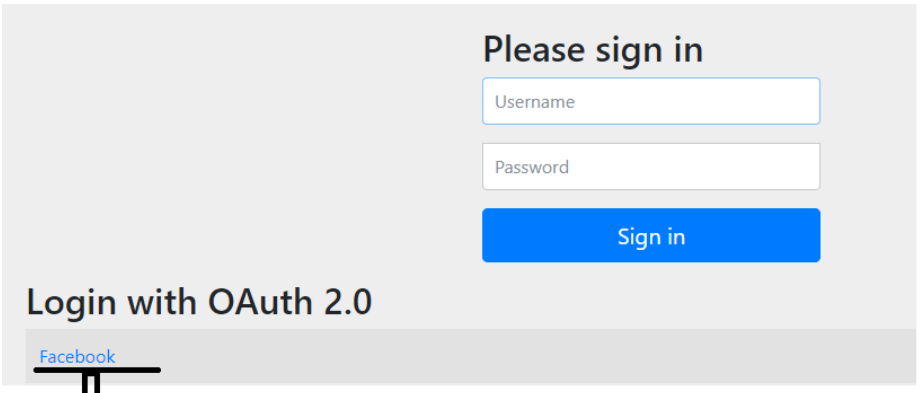
e) add login.html having oauth facebook hyperlink as show below
(collect this from spring security app of custom login page)

```
<html xmlns:th="https://thymeleaf.org">

 <h1 style="color:blue;text-align:center"> Login Page </h1>
<form th:action="@{/login}"  method="POST">
  <table border="0" bgcolor="cyan"  align="center">
    <tr>
        <td>  username :: </td>
        <td>  <input type="text"  name="username" /> </td>
    </tr>
```

```
        <tr>
            <td> password :: </td>
            <td>  <input type="password" name="password" /> </td>
        </tr>
        <tr>
            <td> <input type="submit"  value="Login"> </td>
            <td> <input type="reset"  value="cancel"> </td>
        </tr>
    </table>
     <input type="hidden"  th:name="${_csrf.parameterName}"  th:value="${_csrf.token}"/>
      <span style="color:red;text-align:center"  th:if="${param.error}">Invalid  Login detials (authentication failed)</span>
      <span  style="color:red;text-align:center"  th:if="${param.logout}">User Logged succefully </span>
</form>
    <br>_____(or) <br>

 <br>  <h1 style="color:red;text-align:center"><a href="/oauth2/authorization/facebook">FaceBook Login</a>
```

f) run the Spring boot App

http://localhost:4041/home ---> home page comes directly
http://localhost:4041/after



either login page of facebook
(or)
continue as already Logged in user will com

output will come

http://localhost:4041/user

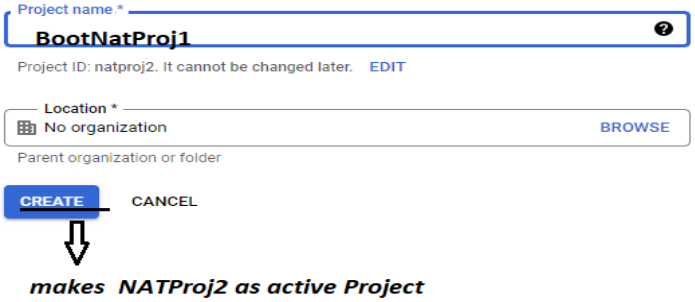
gives current fb user infomration in the form Json content

(developing single sign on applicaiton using Google)

step1) Register Client App with Google api

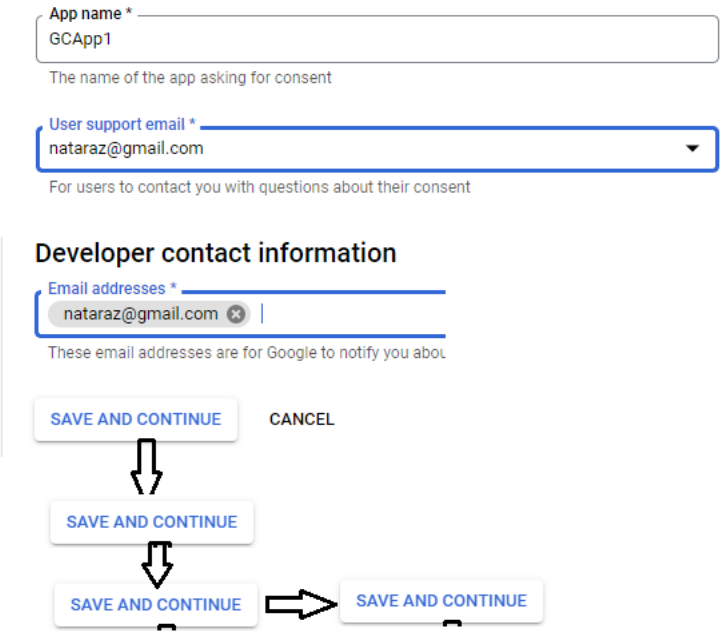
go to https://console.developers.google.com/

Go to projects section select it ---> create new project

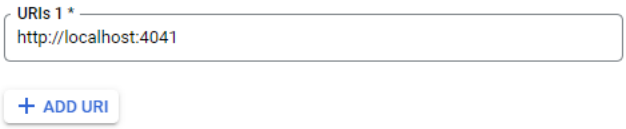


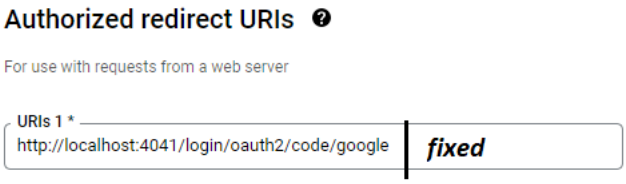
makes NATProj2 as active Project

Go to credentials ----> creaate credentials ---> oauth Client Id ---> configure consent screen ---> select external --->



Credentials ----> create credentials ---> select oauthClient Id ----> application type :: web application





======
create
=====

step2) add the following addtional entries in application.properties file

spring.security.oauth2.client.registration.google.client-id=961105121882-l1pgfjt34ifo906pr38e0gj5gfg8p61f.apps.googleusercontent.com
spring.security.oauth2.client.registration.google.client-secret=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

step3) add the following addtional hyperlink in login.html

```
<br>                                    (or) <br>
<h1 style="color:red;text-align:center"><a href="/oauth2/authorization/google">Google Login</a>
```

step4) run the Client App as spring boot application

http://localhost:4041/home (gives home page)
http://localhost:4041/after

⇩

show login page

⇩

select google login

⇩

select user

.
.
.

=>if authorized redirect url is not added then to the google developer Project creation then we get this error

⇩

Google

Authorization Error

Error 400: redirect_uri_mismatch

You can't sign in to this app because it doesn't comply with Google's OAuth 2.0 policy.

If you're the app developer, register the redirect URI in the Google Cloud Console.

Learn more

Request Details

The content in this section has been provided by the app developer. This content has not been reviewed or verified by Google.

If you're the app developer, make sure that these request details comply with Google policies.

- redirect_uri: http://localhost:4041/login/oauth2/code/google

⇩

Take this url and use it authorized redirect url

47.NTSPBMS615- Messaging Intro - sept2nd-2022

Messaging (Communication using Messages)

==========

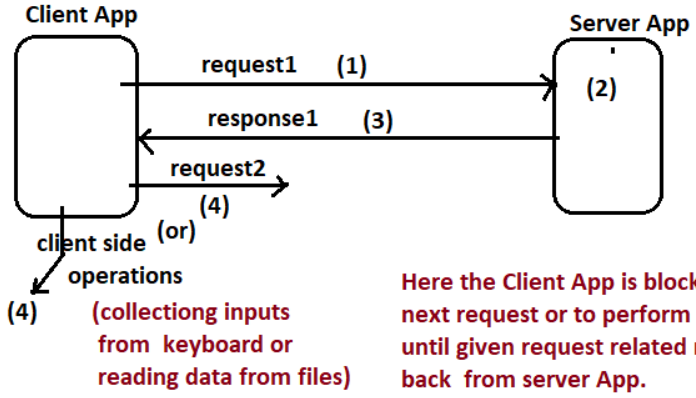
Messaging using JMS with active Mq/rabitmq
Messaging usuing kafka..

The Client - Server Communication is of two types

a) Synchronous communication
b) Ashynchronous Communication

a) Synchronous communication

In this communication, the client App can send next request to Server App only after getting the response for the already given request i.e The Client App should wait for given request related response from server App in order to make it next request to server App or to perform some client side operations.



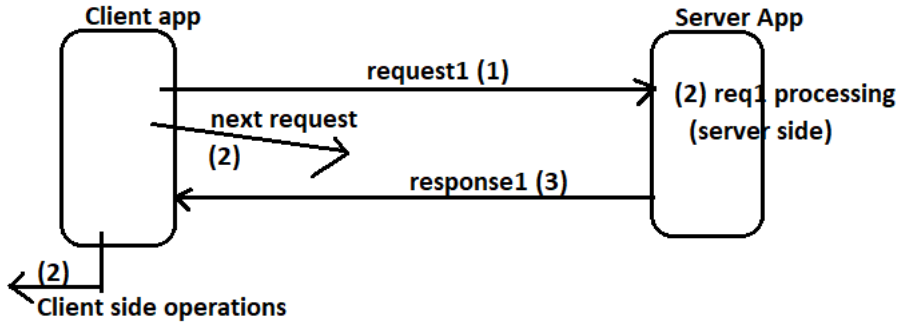
in synchrnous communication .. both Client App and server App must be active at time

Here the Client App is blocked to generate next request or to perform client side operations until given request related response comes back from server App.

note:: By default all Client - server communications are synchronous communincations
(browser to web application communication is synchronous communication by default)

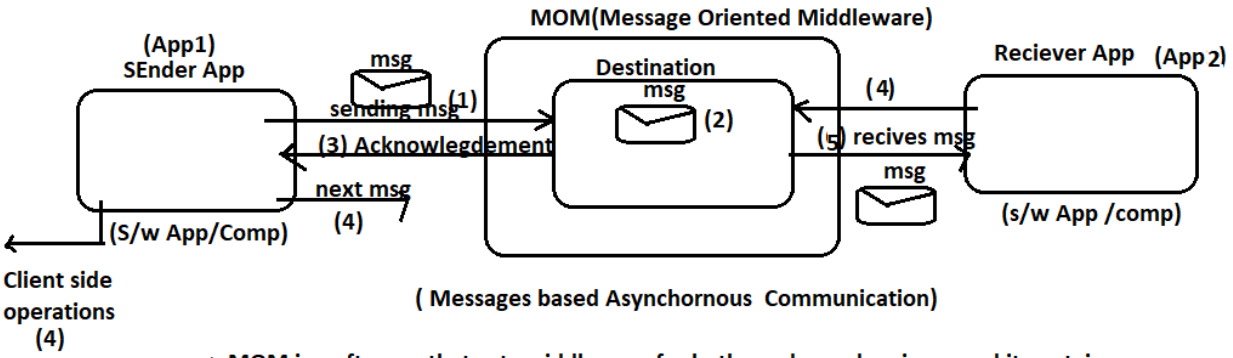
b) Ashynchronous Communication

=>Here Client App is free (not blocked) to generate the next request or to perform client side operations with out waiting for the given request related response from server App..



=>In web applications we can use AJAX (Asynchronous Java Script and Xml) to get asynchrnonous communication b/w browser(Client) and web application(server) (now ajax is part of UI Technologies)

=> we can use "Messaging concept" (Messages based communication) to get Asynchronous communication b/w two software Apps /comps.



(Messages based Asynchornous Communication)

=> MOM is software that acts middle man for both sender and reciever and it contains memory called Destination to hold messages sent by sender App and to deliver the same messages to the Reciver App
eg: active mq, rabbit mq and etc..

note :: In Messaging the messages will go from one to App to another App in continuos flow like stream.

note :: SMS messaging, whatup messaging , mailing does not come under messages based communication becoz that deals with person to person communication.. the actual messaging takes place b/w two software Apps or comps.

use-cases for Messaging :: (Communication b/w softwre apps or comps using messages)

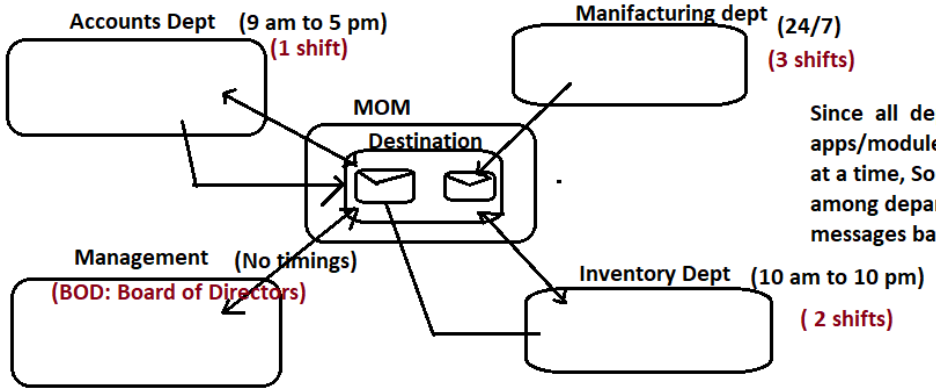
a) Data streaming based on Scheduled /continous Flow
(eg:: uber cab availability status , Goods delivery App store availabiity status , Live Train Running status , Live cricket score status and and etc..)

b) Web activities and serach results
(eg: The advertisement agecy App gathers google search queries continuosly to arrggate search queries and to show direct advetisements to endusers)

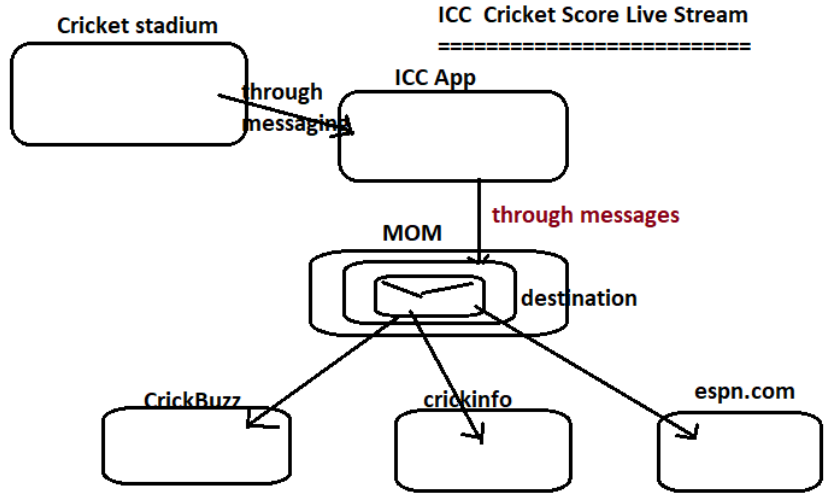
c) Log Aggregation
(eg:: Collecting log messages generated by Production env. App and aggregrating those messages)

note:: method calls based interaction b/w the apps is called synchronous communication
messages based interaction b/w the apps is called asynchronous communication

Car manifacturing company



Since all departments related apps/modules can not be active at a time, So synchronous communication among departments is not possible.. So prefer messages based Asynchronous communication.



note: text streaming or video streaming or some content streaming takes place becoz of messaging concepts..

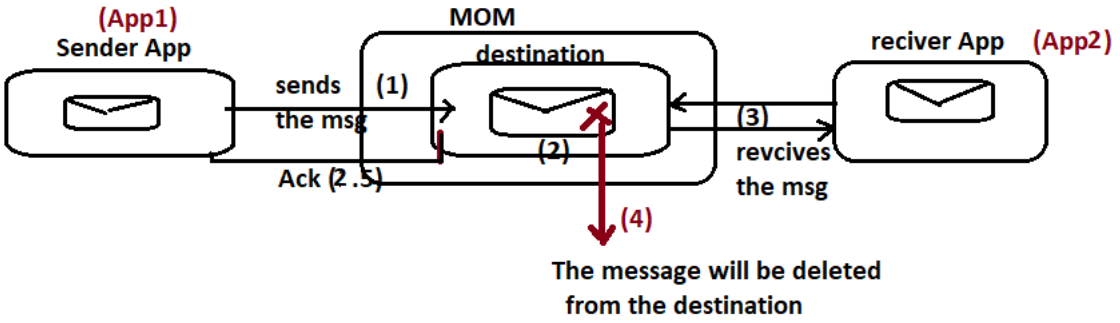
This Messaging based communication can be done in two ways

a) as Point to Point (PTP) communication

b) as Publisher and subscriber (Pub -sub) communication)

a) as Point to Point (PTP) communication

=>Here each message send to destination by sender will have only one consumer/Reciver i.e once consumer consumes the messages the message will be removed from the destination.



=> App1 is sending messages to App2 through MOM then App1 is called SenderApp and App2 is called Reciever App

=>if App2 is sending messages to App1 through MOM then App2 is called Sender App and App1 is called Reciever App

real cases for undering PTP model

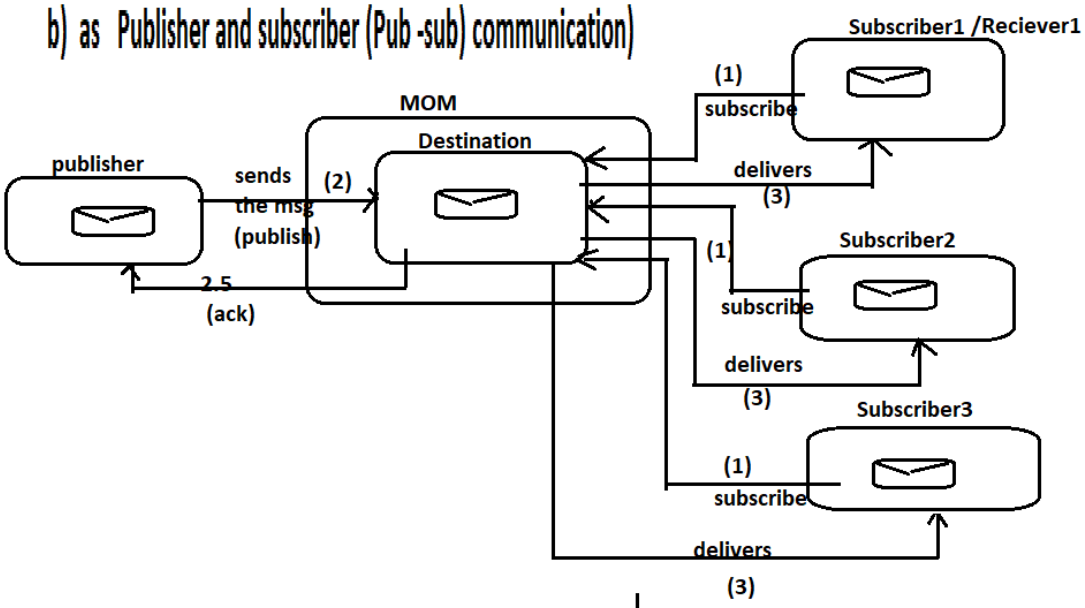
(sender) MOM (reciever)

-> customer food order ----> resturant -----> chef taking order to prepare the food

-> patient doctor appointment ----> hospital ----------> Doctor treatement to patient

(sender) (MOM) (reciever)

b) as Publisher and subscriber (Pub -sub) communication



=> Here each message sent by the publisher will go multiple subscribers who have done their subscription to destination before publisher publishes the message.

(each message can have 0 or more subscribers)

after consuming current messages the subscribers will stay connect to MOM to cosume further messages that are goinging come

realtime scenarios to understand pub-sub model

=> The TV Programs broadcasted by Zee TV will be go multiple subscribers (audions) who had done their subscription before Zee TV broadcasts the programs (only current and future programs can be watched) (Dish tV scription)

=> Joining in naresh it course

48.NTSPBMS615- Messaging using JMS - sept 3rd 2022

=> By Default all Client app -server app communication is synchronous communication
eg:: browser to web application , Rest Client to Rest Service , Ms Client to Ms , MS to MS.
=> In web application to get asynchronous communication b/w browser and web application take the support of AJAX .
=> Between two Java Apps or two MS or Rest Client And RestService if u r looking for Messages based asynchronous communication then go for Messaging/Message Queue with the support of MOM software.

MQ : MessageQueue

=> Messaging /Message Queue can be implemented using two types of porotocols
a) Using Basic MQ Protocol (BMQP) :: For this we need to use JMS
b) Using Adavanced MQ protocol (AMQP) :: For this we need to use RabitMq, Apache kafka and etc..

==========
Messaging /Message Queue using JMS (BMQP Impl)
==========

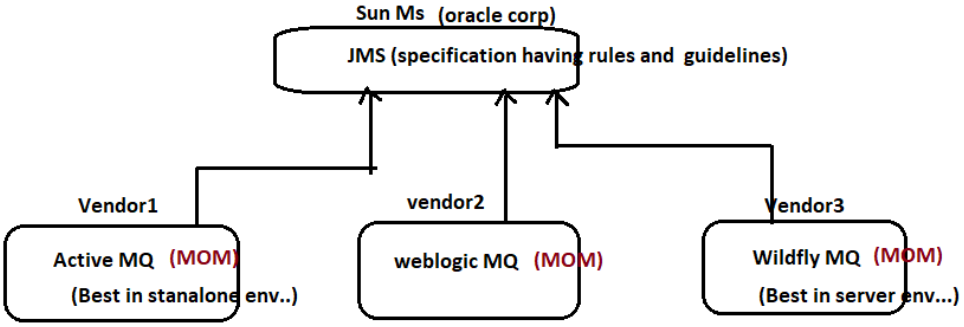
(JMS :: ~~Java~~ jakarta Messsaging Service)

Java Mail Api is different from JMS

java mail api (for mail operations)
(javax.mail and its sub pkgs)

JMS (Java Messaging Service)
(javax.jms and its sub pkgs)

=> JMS is the software specification given by Sun Ms in JEE module having set of rules and guidelines to provide messsages based communication b/w java comps or Apps.



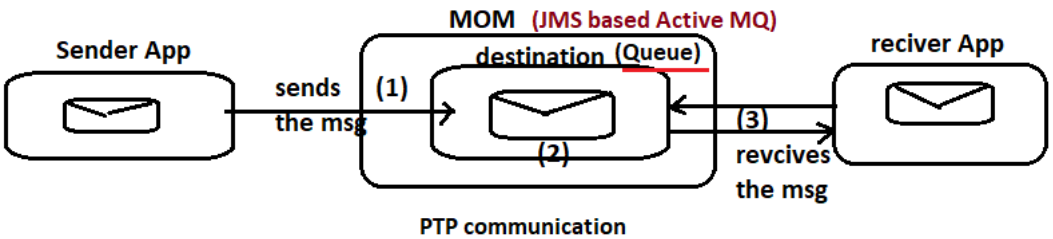
=> Each Implementation software of JMS given by different vendors is called one MOM software

=> Every Application server s/w like weblogic , wildfy and etc.. gives one built-in MOM software
=> Tomcat is not providing any MOM software..so if u r application is using tomcat server then prefere working with Active MQ as the MOM software.

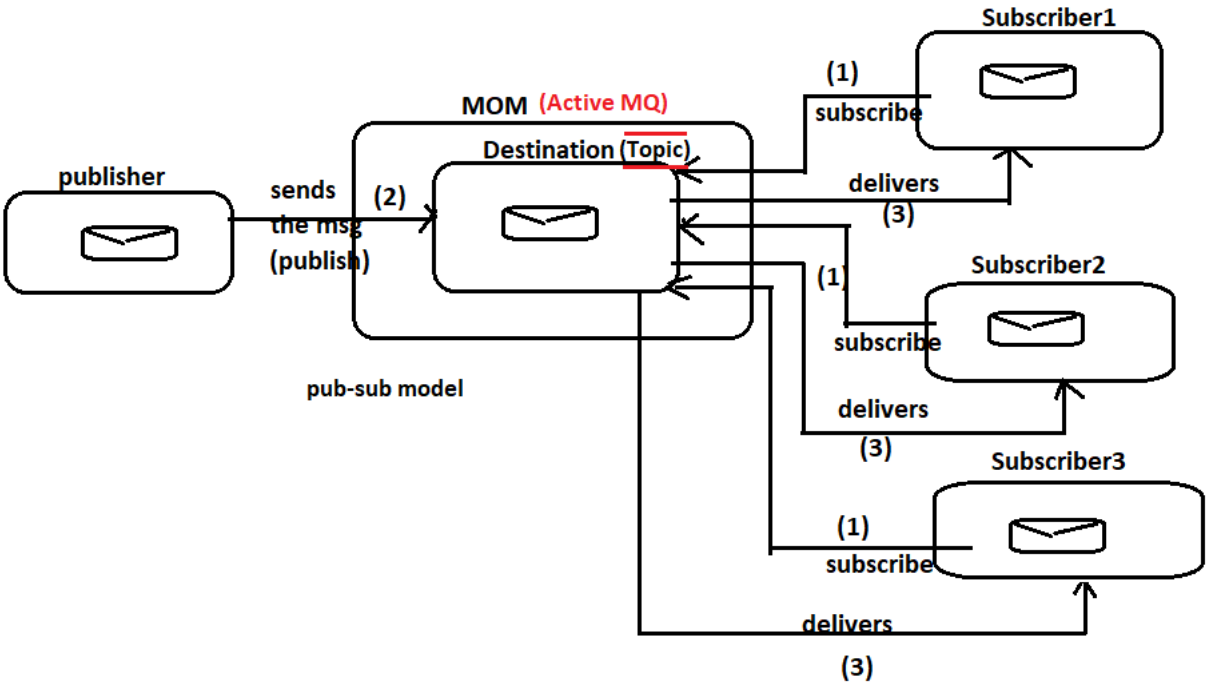
-> In stanalone Apps ,prefer using ActivMQ as JMS based MOM software
-> In web applications, Rest Apps, MicroServices Apps prefer using underlying server based MQ as the JMS based MOM Software .. if the underlying Server software is not supporting that MQ then go for Active MQ

=> JMS supports both PTP and pub-sub Models of Messaging
=> The MiddleMan software between sender and reciver/consumer who takes messages given by sender and holds them for Reciver to come and cosume is called MOM
=> MOM contains destinations as the logical mememories to recive and hold the messages..
In PTP model this destination is called "Queue" and in pub-sub model this destination is called "Topic"

note: Based on Destination type Queue/Topic .. the indurstry identifies Messaging Model name
Queue Destion means the Messaging Model is PTP Model , Topic Destination means the messaging model is PUB-SUB Model



PTP communication



=> JMS provides api representing rules and guidelines in the form of interfaces,classes placed in javax.jms or jakarta.jms and its subpkgs. (It is like JDBC API)

=> Active MQ, weblogic MQ, WildlfyMQ and etc.. are MOM softwares which are internally providing impl classes for JMS API intefaces (These are like JDBC DRIVERS)

=>Programmers take impl classes through JMS interrfaces .. to create objects and to consume services [The Sender and reciver Apps will be developed using JMS Api referring one or another MOM Impl sofwares) (These are like Java Apps using JDBC drivers through JDBC API)

---

sample reference code to understand API , Impl software and App development using API

==========

(Like JMS API)

```
interface Operation1{
  public void process();
}
interface Operation2{
   public String process(String msg);
}
```

It is like JMS API given by Sun Ms (JMS specification)

(JMS Impl software -MOM software)

(Active MQ)

```
public class Operation1Impl implements Operation1{
  public void Process(){
     ....
   }
}
public class Operation2Impl implements Operation2{
  public String Process(Stirng msg){
     ....
   }
}
```

=>mpl classes provided by the Vendor

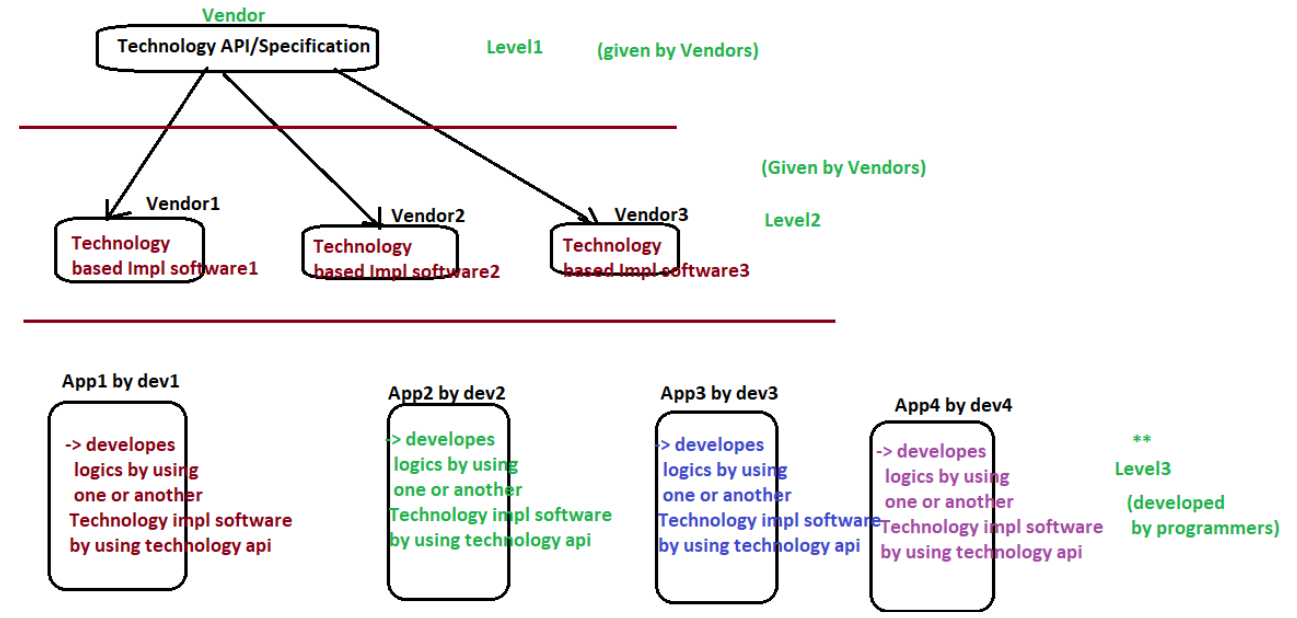
(It is like active Mq , weblogic mq and etc.. MOM softwares give by different Vendors

---

(JMS - Sender or Recieve App)

```
Operation1 op1=new Operation1Impl();
 op1.process();
Operation2 op2=new Operation2Impl();
String result= op2.process("hello");
```

It Like JMS application devleoped by the Programmer



(Core Java Recap)

=>The interface that contains only one abstract method direclty or indirectly is called functional Interface.

```
@FunctionalInterface
interface Operation1{
   public void process(String msg);
}
```

Functional interface

Impl1 ::

```
public class Operation1Impl implements Operation1{
    public void proess(String msg){
       ...
       ..
    }
}
```

Normal Impl class

Impl2 :: Anyonomous inner class (inline impl)

```
Operation op1= new Operation1(){
                    public void process(Stirng msg){
                        ...
                        ...
                    }
                };
```

Here Anonymous inner class is created implementing Operation1 interface and process(-) is implemented in that class.

(a) anonymous inner class is created implementing Operation1(I) ..
(b) In the above class process(-) method Operation1(I) is implemented
(c) Object is created for Anonymous inner class and that refered Operation1(I) ref variable

Impl3 :: Lamda based anonymous inner class (anonymous inner clss Impl class obj + method impl)
(InLine impl)

```
Operation op1= (msg)->{ ...,
                        }....
```

(or)

```
Operation op1=msg->{ ....  }
```

LAMDA based anonymous inner class obj creation for interface is possible only when that interface is functional interface.

---

spring

The JMS api is providing one Funcational Interface called "MessageCreator" as shown below

```
@FunctionalInterface
public interface MessageCreator {
    Message     createMessage(Session session);
    }
```

represents the connectivity with MOM software from Sender /Reciever App
It is like JDBC Connection object

=>Spring JMS is part of spring JEE module and provides abstraction on plain JMS api to simplify the process of developing sender and reciever app.

=> spring boot JMS = Spring JMS ++

Impl1 (Using anynomous innner class)

```
MessageCreator mc=new MessageCreator(){
                public Message createMessage(Sesssion ses){
                    .... // logic to create MEssage object
                    ....
                    return message obj;
                    }
    }
```

Impl2 (Using Lamda anonymous inner class) (Best from Java 8 onwaords)

```
MessageCreator mc=(ses)->{ .....
                            ...
                            return message obj;
                           }
```

(or)

```
MessageCreator mc=ses-> message obj;
```

(if the code is single line

SpringBoot JMS/spring JMS provides abstraction on plain JMS to simplify the messages based communication with the support of "JMSTemplate" (template method DP)

=>TemplateMethod DP says i will give one algorithm or plan to complete a job where common things will be taken care by supper class and specific things will be left to sub classes to implement.

==========
Procedure to keep ActiveMq as MOM software
==========

step1) Dowload ActiveMQ software as zip file from Internet

https://activemq.apache.org/components/classic/download/

ActiveMQ 5.17.2 (Sep 2nd, 2022)

Release Notes | Release Page | Documentation | Java compatibility: 11+

Windows	apache-activemq-5.17.2-bin.zip	SHA512	GPG Signature
Unix/Linux/Cygwin	apache-activemq-5.17.2-bin.tar.gz	SHA512	GPG Signature
Source Code Distribution:	activemq-parent-5.17.2-source-release.zip	SHA512	GPG Signature

step2) Extract the zip file to a folder.

step3) start the ActiveMQ software

E:\ActiveMQSoft\apache-activemq-5.16.3\bin\win64\activemq.bat file

Make sure that java 11 and above versions are added CLASSPATH env.. variable

step4) open admin console page of Active MQ software

http://localhost:8161

username : admin
password : admin

step5) Observe Topic and Queue sections in admin console page

home page ---> manage active mq --->

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

---

Procedure to develop JMS PTP (Queue) application using active MQ

=>In any JMS App (Producer /Sender or Reciver /Subscriber/consumer) once we spring -boot-starter-activemq starter as dependency we get JMSTemplate class object through AutoConfiguration that can be injected to Sender / Reciver App to send /recieve message effectively/easily

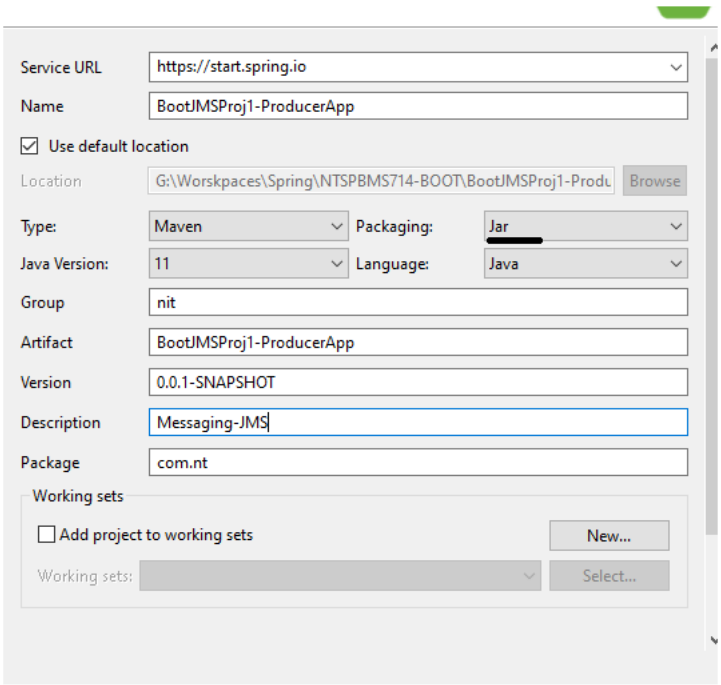
Procedure to develop JMS PTP (Queue) application using active MQ

=>In any JMS App (Producer /Sender or Reciver /Subscriber/consumer) once we spring -boot-starter-activemq starter as dependency we get JMSTemplate class object through AutoConfiguration that can be injected to Sender / Reciver App through Autowiring.

In any JMS One Sender/Publisher App and Reciever / Subscriber App will be there..

For Producer App

step1) create spring boot project adding active mq starter..




```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-activemq</artifactId>
    </dependency>
```

step2) add the following properties in application.properties file

```
# MOM connectivity Details      #8161 for admin console , 61616 for actual MOM service
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin

#enable PTP communication
# true enables pub-sub model and false enables ptp model
spring.jms.pub-sub-domain=false
```

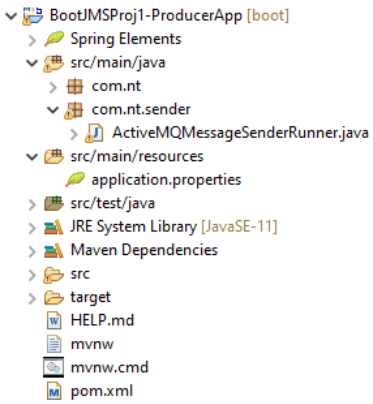
step3) Develop runner class as the Message sender

=>JmsTemplate class is having send(-,-) method taking destination (queue/topic) logical name(generally new name) and MessageCreator(I) (Functional interface) impl class obj.

```
public void send(Destination destination,  MessageCreator messageCreator)  throws JmsException
```

For this we can pass either anonymous inner class obj or Lamda style inner class obj

Runner class having sender logic

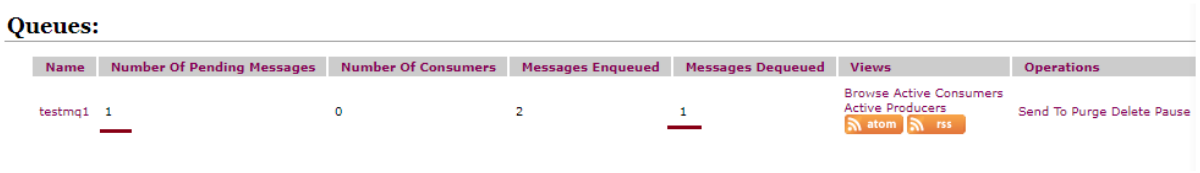



```
package com.nt.sender;

import java.util.Date;

import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.Session;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.jms.core.MessageCreator;
import org.springframework.stereotype.Component;

@Component
public class ActiveMQMessageSenderRunner implements CommandLineRunner {
    @Autowired
    private JmsTemplate template;

    @Override
    public void run(String... args) throws Exception {
  /*    //using anonymous inner class logics
        template.send("testmq1", new MessageCreator() {
            @Override
            public Message createMessage(Session ses) throws JMSException {
                Message message=ses.createTextMessage("From Sender at ::"+new Date());
                return message;
            }
        }); */

    /* using LAMDA style anonyomous inner class
     template.send("testmq1",ses->{
         return ses.createTextMessage("From sender at"+new Date());
    });*/

        //using LAMDA style anonyomous inner class
        template.send("testmq1",ses-> ses.createTextMessage("From sender at"+new Date()));
        System.out.println("Message sent");
    }//run
}//class
```

step4) Make sure that Active MQ started

E:\ActiveMQSoft\apache-activemq-5.16.3\bin\win64\activemq.bat

step5) Run the Sender App and ActiveMQ Server console

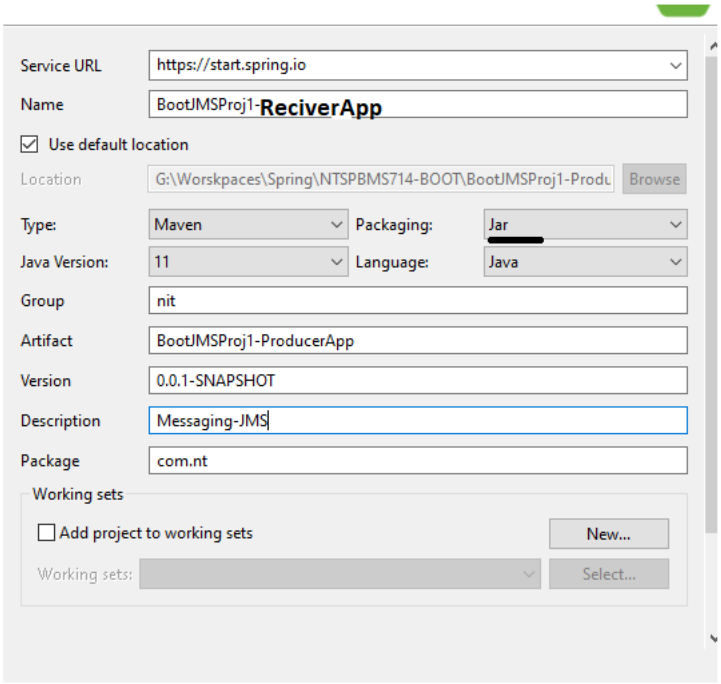


==========

Procecedure to develop Consumer/Reciever App of PTP model using ActiveMQ MOM software

==========

step1) create spring boot project adding active mq starter..




```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-activemq</artifactId>
    </dependency>
```

step2) add the following properties in application.properties file

```
# MOM connectivity Details      #8161 for admin console , 61616 for actual MOM service
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin

#enable PTP communication
# true enables pub-sub model and false enables ptp model
spring.jms.pub-sub-domain=false
```

step3) Develope the java class having @JMsListener method as shown below.

```
package com.nt.reciever;

import org.springframework.jms.annotation.JmsListener;
import org.springframework.stereotype.Component;

@Component
public class JmsMessageConsumer {

    @JmsListener(destination = "testmq1")
    public  void  readMessage(String text) {
        System.out.println("Recived Message::"+text);
    }

}
```

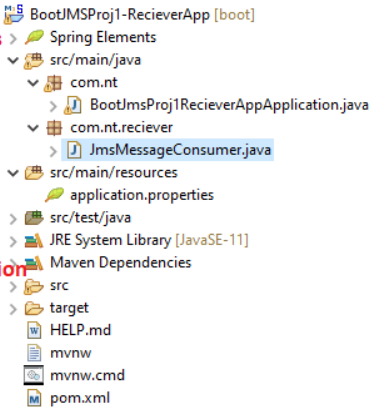
readMessage(-) executes automactically to read message from queue destination

becomes callback method to read the message from queue destination

This method name can be any thing

@JmsListener(destination="....")

Annotation that marks a method to be the target of a JMS message listener on the specified destination



step4) Run the App Consumer App

(reads the message from Quue destinaiuon)

step5) Observe the console

Queues:

Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views
testmq1	0	0	2	2	Browse Active Consumers Active Producers atom rss

on

Keypoints PTP model messaging

(a) The destination name is "Queue". (FIFO rule)

(b) Both Sender and reciver need not be active at a time

(c) One Message will have only one Reciever/Consumer

(d) The Queue Destination can send message to reciever App who connected to the Destination before Sender sends the message and will also only the recieve the message when the sender sends the messsage.

(e) if multiple consumers are waiting for a message sent by the SEnder then only first reciver the recieves the message.
(firstly started reciever)

(f) The queue destination delivers the message to Reciever App and deletes the message from queue destination

usecase:: our car factory usecase needs this model (PTP) communication

How to make Sender of App of PTP Model sending message continuosly to MOM software

=> we can take the support of scheduling concept.. For that we need to add @Scheduling annotation on the b.method of Sender App.

note:: @Scheduling can not be applied on the method with args.. So make sure that u r b.method is designed having no args/params.

```
@Component
public class ActiveMQMessageSender    {
    @Autowired
    private  JmsTemplate  template;
```

```
    @Scheduled(cron="*/10 * * * * *")
    public void sendMessage() {
        //using LAMDA  style anonyomous inner class
         template.send("testmq1",ses-> ses.createTextMessage("From sender at"+new Date()));
         System.out.println("Message sent");
    }//run
}//class
```

note: also place @EnableScheduling on the top of main class

note:: while developing sender and Reciver Apps .. placing @EnableJms on the top of main class is optional.

==========

Developing ActiveMQ Pub-sub model App

==========

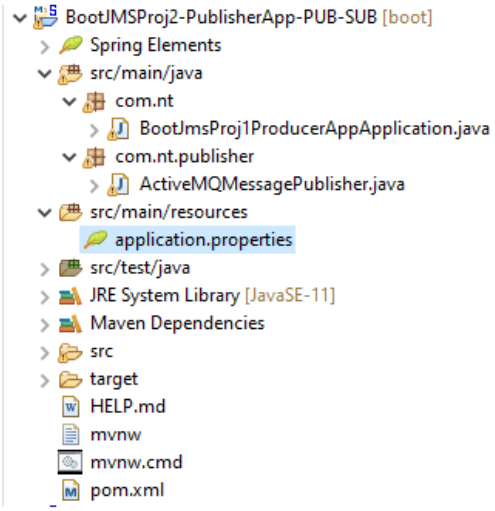
=>All are same in both Sender/Publisher App and also in Reciever/Subscriber App but change

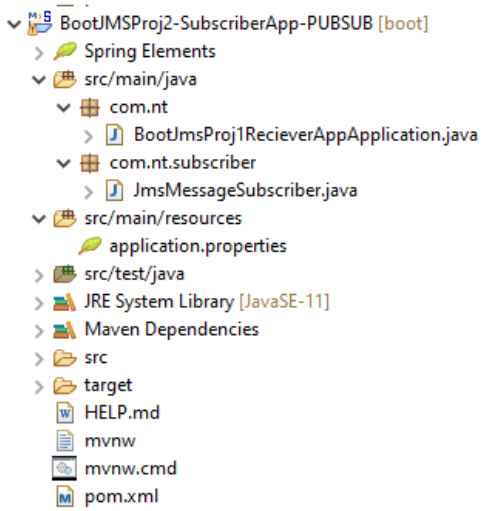
spring.jms.pub-sub-domain value to true

in application.properties

---

spring.jms.pub-sub-domain=true





Order of execution

==========

=>Run subscriber App for multiple times (more than 1 time) (To give the feel of more subscribers)

=>Run publisher App

=> Check the console window of subscribers application

(web console screen /page of active MQ)

Keypoints on pub sub model

==========

(a) One message published by publisher can be consumed by more than one subscriber

(b) The subscribers must done their subscription before publisher publishes the message

(c) The message published by publisher will be duplicated and will be delivered to multuiple subscribers

(d) The Publisher and Subscribers all must be in active mode at a time.

(e) In this model the Destination name is Topic

(f) Once the subscriber consums the message the message from Topic destination of MOM s/w will not be deleted. from TOPIC

Can we send java object data over the network ?

a) yes , by making the object as Serializable object.

note: Serializing object means the converting object data into stream of bits -bytes that are required to send data over the network or to write to destination file.

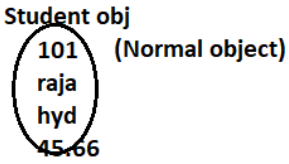
How does normal object become Serilizable object?

=> By making the class of the object as the Serializable class .. (class must implement java.io.Serializable(I))

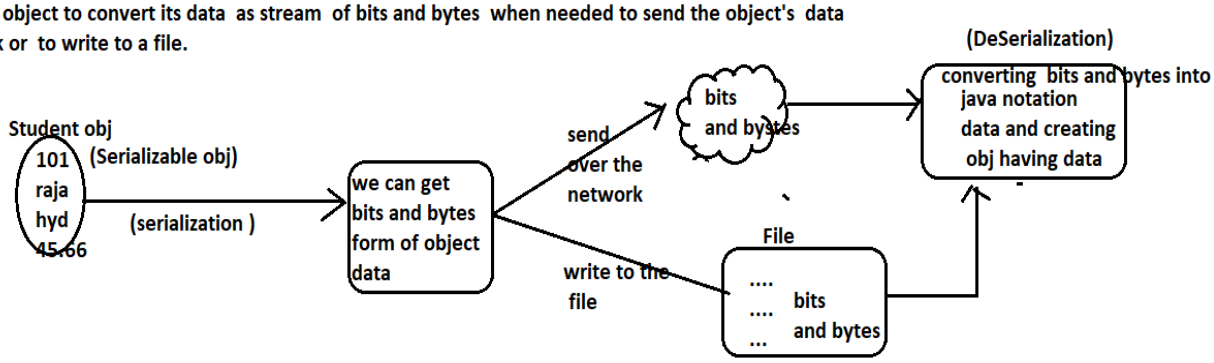
java.io.Serializable(I) is empty interface (marker interface) then how does it makes its impl class object as the Serializable object?

Ans) Markers Interface does nothing directly .. By seeing the marker interface implementation the underlying JVM /Server /Framework /Container provides special runtime capabilities to implementation class objs they create.

=> if JVM is create object for normal class .. that object data is in java format and can not be converted to stream of bits and bytes i.e we can not send/write normal object over the network or to a file.



=> if JVM is creating object for Serializable class (class implementing java.io.Serializable(I)) then the JVM provides capability to the object to convert its data as stream of bits and bytes when needed to send the object's data over the network or to write to a file.



=>Most of marker interfaces are empty interface.. but we can not say every empty interface is marker interface.

=> We can develop our own custom marker interface .. but we need to create custom Container to provider runtime capabilities to the impl class objs of custom marker interface.

other marker interfaces are

==========

java.io.Serializable(I)
java.lang.Cloneable(I)
java.rmi.Remote(I)
Repository(I) of spring data jpa
java.lang.Runnable (not empty)
and etc..

To decide wheather interface is marker or not ...
do not take emptiness as the criteria.. Take wheather the underlying JVM/container/server/Framework/... providing special runtime capabilities to the impl class object or not.

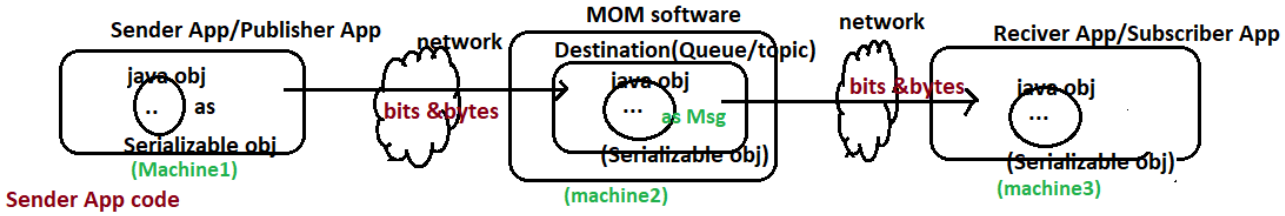
49.NTSPBMS615- Messaging using JMS for sendign Object as Message - sept6th 2022

==========

Developing ActiveMq PTP application to send Object as message

==========

=> since Sender and Reciever Apps can be there in two different machines of a network or there is possibility of running MOM software on different machine so we need to take the object as Serializable object ..



Sender App code

==========

step1) create spring boot project adding activemq , lombok dependencies

step2) Develop the Java bean class as the Model class.

```
//Model class
============
package com.nt.model;
import java.io.Serializable;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ActorInfo implements Serializable {
        private  Integer actorId;
        private String actorName;
        private String actorAddrs;
}
```

step3) Develop the Sender App by Enabling Scheduling

```
//Sender class
----------------------
package com.nt.sender;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import com.nt.model.ActorInfo;

@Component
public class ObjectMessageSender {
        @Autowired
        private JmsTemplate template;

        @Scheduled(cron = "0/20 * * * * * ")
        public  void  sendObjectDataAsMessage() {
                //prepare object
                 ActorInfo actor=new ActorInfo(1001, "ranveer", "mumbai");
                //send object as the message
                 template.convertAndSend("obj_mq1", actor);
                 System.out.println("Object is send  as Message ");
        }

}
```

The SimpleMEssageConvertor class contains logic to convert given Serializable object into JMS style Message object.

main class

```
@SpringBootApplication
@EnableScheduling
@EnableJms
public class BootJmsProj3SendingObjectSenderAppApplication {

	public static void main(String[] args) {
		SpringApplication.run(BootJmsProj3SendingObjectSenderAppApplication.class, args);
	}

}
```

step4) Add properties in application.properties file

```
MOM connectivity Details
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin

#enable PTP communication
# true enables pub-sub model and false enables ptp model
spring.jms.pub-sub-domain=false
```

```
# make all packages as the trusted packages (especically model class pkg)
# to send/recieve model class object data as message to MOM software
spring.activemq.packages.trust-all=true
 # or
#spring.activemq.packages.trusted=com.nt.model
```

if we do not place this entry them there is a possibility of getting the following error

This class is not trusted to be serialized as ObjectMessage payload. Please take a look at http://activemq.apache.org/objectmessage.html for more information on how to configure trusted classes.

step5) Makesure Activem MQ software (MOM software) is in running mode

use E:\ActiveMQSoft\apache-activemq-5.16.3\bin\win64\activemq.bat file

step6) Run the Send App ...

Reciver App Code
==========

step1) create spring boot project adding activemq , lombok dependencies

step2) Develop the Java bean class as the Model class. (copy from sender App)

```
//Model class
============
package com.nt.model;
import java.io.Serializable;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ActorInfo implements Serializable {
	private Integer actorId;
	private String actorName;
	private String actorAddrs;

}
```

step3) , Add properties in application.properties file

```
MOM connectivity Details
spring.activemq.broker-url=tcp://localhost:61616
spring.activemq.user=admin
spring.activemq.password=admin

#enable PTP communication
# true enables pub-sub model and false enables ptp model
spring.jms.pub-sub-domain=false

# make all packages as the trusted packages (especically model class pkg)
# to send/recieve model class object data as message to MOM software
spring.activemq.packages.trust-all=true
 # or
#spring.activemq.packages.trusted=com.nt.model
```

step4) Deveop the ReciverApp as JMSListener...

```
package com.nt.reciever;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.jms.core.JmsTemplate;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import com.nt.model.ActorInfo;

@Component
public class ObjectMessageReciever {

	@JmsListener(destination = "obj_mq1")
	public  void  consumeObjectDataAsMessage(ActorInfo  actor) {
		System.out.println("Recived Object Data ::"+actor);
	}
}
```

step5) Run the Reciver App ...

Limitation of JMS based Messaging

---

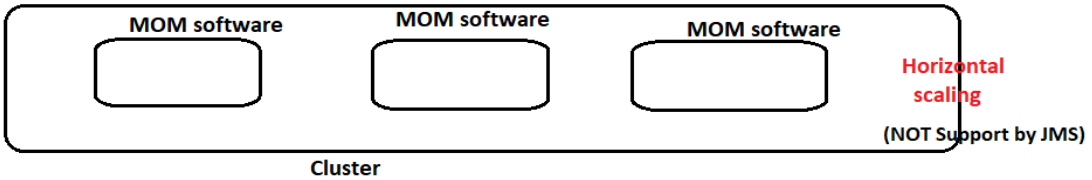
a) JMS is java language dependent technology i.e we need to develop both sender and reciver Apps in same java language .. (JMS can not be used outside of Java Domain) (no support of interoperability)

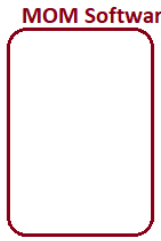
b) JMS based MOM softwares can recieve and send messages only by using protocol TCP i.e we can not use other than protocol TCP like http , smtp and etc..

c) There is a possibilty of loosing data /message if the MOM software is down or MOM software is not responding while publisher or sender is sending the messages

d) if the Message is very big / large scale then MOM software behaves very slow (i.e gives the performance issue)

e) There is no ability of creating multiple instances of single MOM software .. if multiple senders /publishers are sending messages simultaenously the Performance of single copy MOM software may not suitable for industry needs.
(It indirectly say JMS style MOM software does not support horizontal scaling.. it supports only vertical scaling)





Veritical scaling

(Supported by JMS)

(Increase its capacity
by adding more RAMs,
CPUs, HDD to computer)

Conclusion:: use JMS style messaging only when no.of message are less and no.of senders/publishers are less towards sending messages. otherwise go for AMQP based Kafka/rabbit Mq messaging.

50.NTSPBMS615- kafka Intro - sept 7th

Apache Kafka (It is based on AMQP -- Advanced MessageQueue Protocol)

==========

=> It is java based messaging technique .. which can be implemented in different languges

=> It supports to use different technologies and concepts for messaging activity like hadoop, spark, scala and etc..

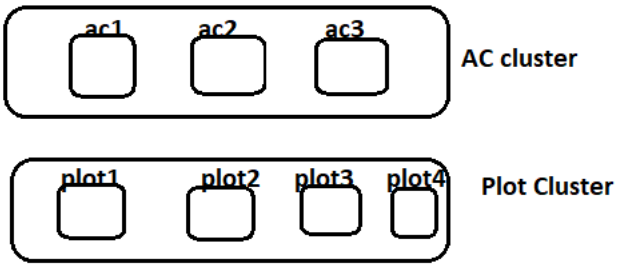
(kafka integration is possible with multiple technologies of java and non-java env..)

=> Apache kafka is all about

Store + Process + Integrate (Transfer)

=>Kafka is basically used to transfer (integrate) data /messsages between multiple complex Apps/systems using Cluster design.

cluster :: set of similar items is called cluster.



=>Kafka intergration /interaction with non-java applications is possible with the support of REST calls.

=> Ka fka is protocol Independent ... we can write code to send messages across the applications using http ,tcp ,ftp and etc .. protocols. (JMS supports only tcp)

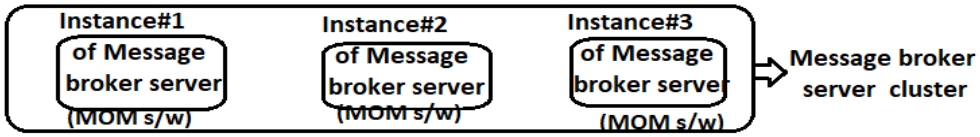
=> Kafka allows to take multiple message broker s/ws (MOM s/ws) at a time as cluster having support for Horizontal scaling giving the following advantages.. (JMS supports only Vertical scaling)

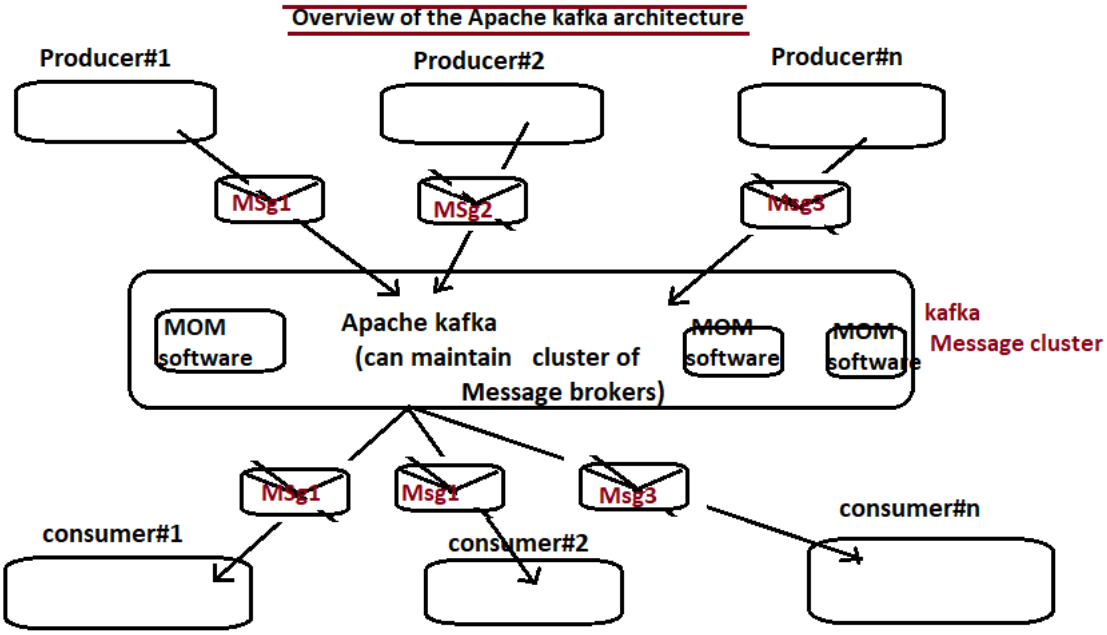
a) Fast data transfer for target data set /messagess

b) No data loss even one messsage broker s/w or MOM s/w is down becoz other broker s/w or MOM s/w takes care of messages.

c) It takes the support of apache zookeeper to handle load balancing among the multiple instances of message broker s/ws or MOM s/w

apache zookeepr is like netflix eureka server..





Kafka Message Broker / Broker server / Mom Server

==========

It behaves like MOM/Broker to recieve messages/data from sender , to hold them as needed and to deliver to Consumer ..

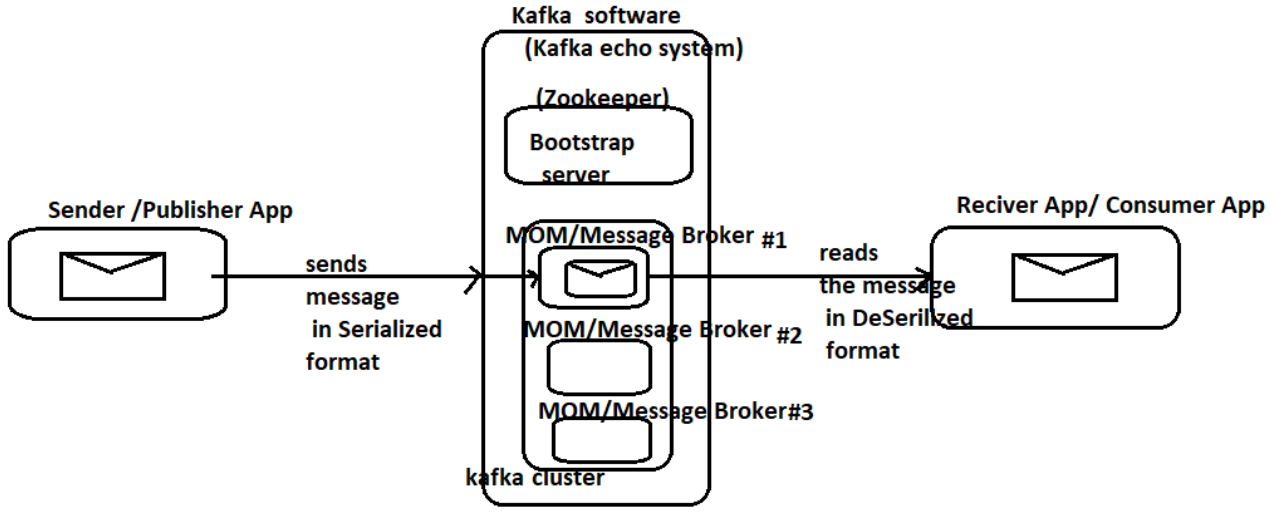
=>When kakfa s/w is started one instance of Message broker will be created automatically.. and it can be increased as needed.

=> collection of message broker instances togather is called kafka message broker cluster .. and we can add any no.of instances in one cluster .. i.e there is no limit to add instances.

=>Apache Zookeepr is resposible to manage message broker instances of cluster by applying Load Balance support and it is also called Bootstrap server becoz it is responsbile to create cluster having single instance and increasing the instances as needed.

=>Apache kafka Eco System = (Zookeeper) Boostrap server + message broker cluster. (with one or more MOM s/ws)

=> The apache kafka Message broker gets Message from Sender /Publisher App in Serialized format and delivers to consumer /subscriber in DeSerialized format.



=> kafka do not support PTP model messaging i.e the message broker/MOM can not have Queue as the Destination

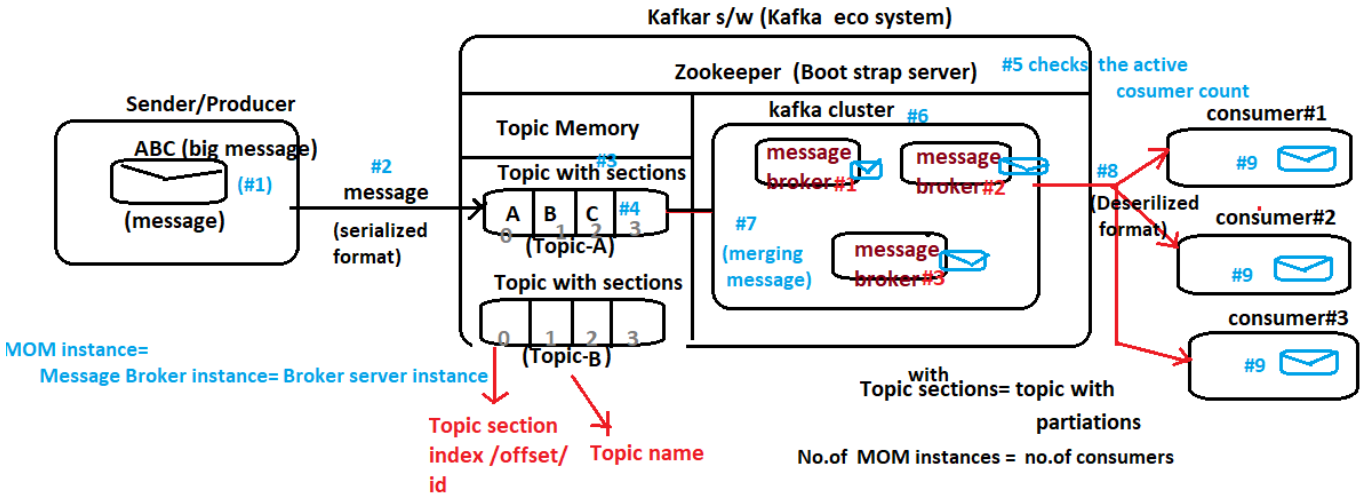
=> kafka supports only Pubsub mode messaging i.e the message broker /MOM can have only Topic as the destination.

=> To send message only to one consumer .. we take the support of TOPIC Destination.

=>For one message one topic destination (Memory) is created/allocated ..that stores given data /message in partiations (huge messsage will be divied into parts/packets). Every paritiation of the message is indentified with an index like 0,1 ,2,3 .. These indexes are technically called as offsets.

=> In kafka messaging , For one consimer one message broker instance will be allocated at a time.
So to send 1 message to 5 consumers we need
1 Sender ---> 1 message --> 1 Topic with sections/partiations --> 5 message broker instanaces -- 5 cosumers (MOM instances)

=> At a time , the Message broker reads one paritiation data , takes the data , replicates data (cloned data) and sends to consumer i.e each large message/data will be stored in multiple partiations of topic and the messge broker reads data from all partiations and sends to its respective consumer.



=>Each message broker instance is capable of reading message from multiple sections/partiations of Topic destinations... replicates the message parts -- merges the message parts --sends the message to respective consumer. Reading content from topic sections , merging the sections into single message , creating message cluster with MOM instance and sending message to that MOM instance is taken care by bootstrap server ...

Sender sends the Message -----> TOPIC with sections will be created having message partiations (indexes/ offsets) --> Zookeeper reads from the content topic destiations and merges into single message ----> Zookeper creates kaflk Message Broker Cluster with default instance .. and writes the message to that Broker /MOM instance ---> Cosumer consumes the message from respective MOM instance

kafka use-cases :: To send and recive big messages/complex message between publisher/sender and subscriber /consumer we take the support of kakfa

a) Live Game scoring b) stock share values streaming c) Live game broadcasting d) Digital movie screening market

b) Live corona reports across the world

concluson :: use JMS style messaging .. if the message are simple and short lived
eg:: messaging b/w different departments of manifacturing company
eg:: Passing Live attendenc details every to university from colleges between 9am to 10am

use apache kafka style message if the messages are complex and long lived

eg: Live train details streaming , FB Live streaming , Live game streaming , Digital movie screening and etc...
Zoom Live classes

Apache Kafka terminologies

Message Broker /MOM :: The mediator /Middle man that will transfer message to consumer by reading message/data from Topic /Topic partiotions

kafka cluster :: collection /group of message broker instances which are created based cosumers count is called kafka cluster.

Topic :: The memory that holds message sent by the Producer in parts

Producer :: The App/comp that sends the message to Topic Memory in Serialized format

Consumer :: The App/comp that reads the message from Topic Memory through Message Broker in DeSerialized format

offset/index :: The id or index given to partiation of the message/data in Topic

Zookeeper /BootstapServer :: Handles Message Broker Cluster and Topic Memory .. While managing the Message broker cluster it will do Load Balancing.

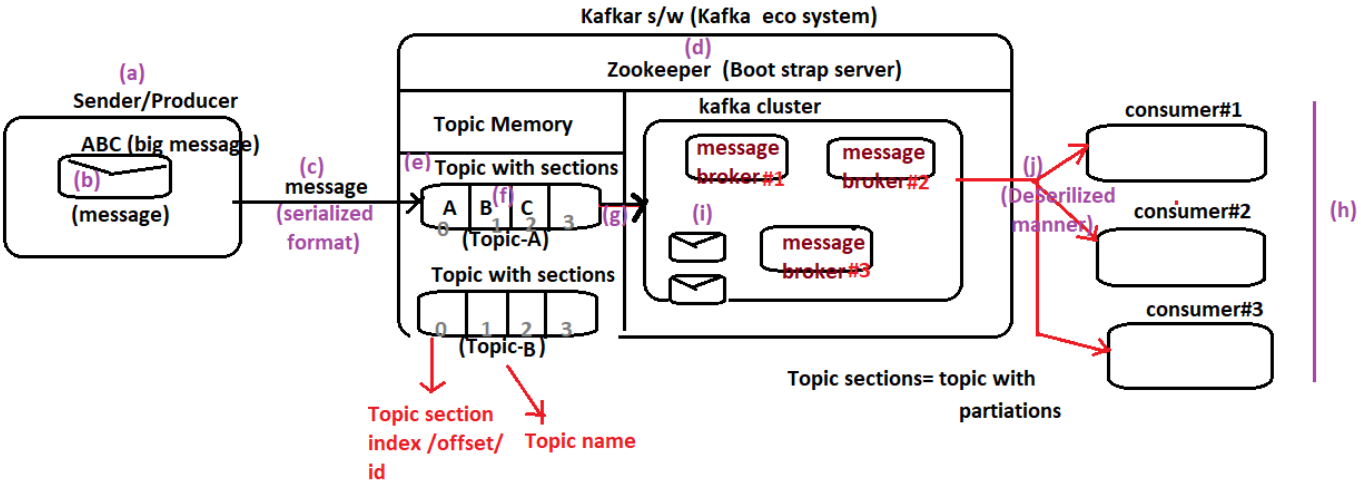
Replica-Factor :: The no.of cloned/duplicate messages that should be created in order to send the messages to Consumers through message brokers .. Generally it is decided based consumer count.

=>if there are 10 cosumers then Replica-factor is 1/10 (For 1 message 10 clonnded copies are required to send to 10 consumers through 10 Message brokers

==========

Apache kafka Work Flow Diagram

==========



=>Each message broker instance is capable of reading message from multiple sections/partiations of Topic destinations... replicates the message parts -- merges the message parts --sends the message to respective consumer.

(a) Define one Producer App

(b) Create either Static message or dynamic message that can be given at runtime

(c) Send message from Producer in the form fo key=value pair format

here "key" is topic name (With the given name topic available then uses it other wise new topic will be created)

"value" is the message/Data

note:: Producer App sends this data/message in Serilized format.

(d) Start zoo keeper (Boot starp server)

(e) Topic Section will be created in Topic Memory to read the message sent by the producer App..

note: New Topic section will be created (if not already available) otherwise it will use the existing Topicsection

(f) Topic section reads message and stores message in different partiations of Topic which will be created as needed.

(note:: These partiations are identified with indexes /offsets)

(g) creates link b/w Topic section/Topic and Message broker

(h) Define one or more consumer Apps as needed by specifying topic name
(note:: Based on the given Topic name in cosumer App(s) the link b/w Topic , message broker and Cosumer App will be created as needed)

(note:: The zookeeper takes the responsbility of creating messagebroker instances as cluster based on the Consumer count and takes care of LB -Load Balancing)

(i) The Message Broker(s) reads the data/messages from Topic Partiations creates cloned/replicate copies of these partitaion messages.

j) The message broker(s) sends the cloned message to partiation s one or more consumer(s) part by part. (In DeSerialized format)

Arranging Apache Kafka Software

Go to https://kafka.apache.org/downloads

3.1.0

- Released January 24, 2022
- Release Notes
- Source download: kafka-3.1.0-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-3.1.0.tgz (asc, sha512)
  - Scala 2.13 - kafka_2.13-3.1.0.tgz (asc, sha512)

gives kafka_2.13-3.1.0.tgz
==> Copy this file (E: drive) and extract it

---

and Recieving
Sending Message using apache kafka infrastructure (No java coding here.. All operations will be done by using kafka commands)

step1) start Zookeeper as Boot strap server..

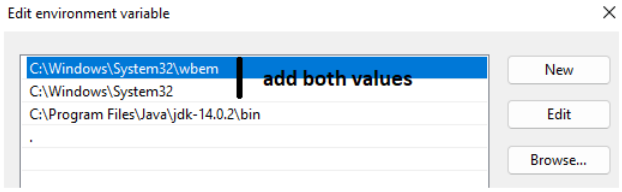
```
E:\kafka_2.13-3.1.0\bin\windows>zookeeper-server-start E:\kafka_2.13-3.1.0\config\zookeeper.properties
```

command as .bat file

(input value)

step2) Do apache kakfa s/w setup and start apache kafka server

i) make sure that "C:\Windows\System32\wbem" is added PATH env.. variable (User Variable)
"c:\windows\System32" should also be add
Mycomputer ---> proeprties ----> advanced system settings --->



env.. variables ---> user vairables
-> new --->

variable name :: PATH
value :: C:\Windows\System32\wbem; C:\Windows\System32

ok --->ok --->ok

ii) `E:\kafka_2.13-3.1.0\bin\windows>kafka-server-start E:\kafka_2.13-3.1.0\config\server.properties`

step3) Create new topic specifying reflection factor (no.of copies) , partintiations count (offset count)

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-topics --create --bootstrap-server localhost: 9092
                                --replication-factor 1 --partitions 1 --topic nit-tpc
```

step4) Create a producer and link with message broker with support of bootstrap-server

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-console-producer --bootstrap-server localhost:9092 --topic nit-tpc hello
```

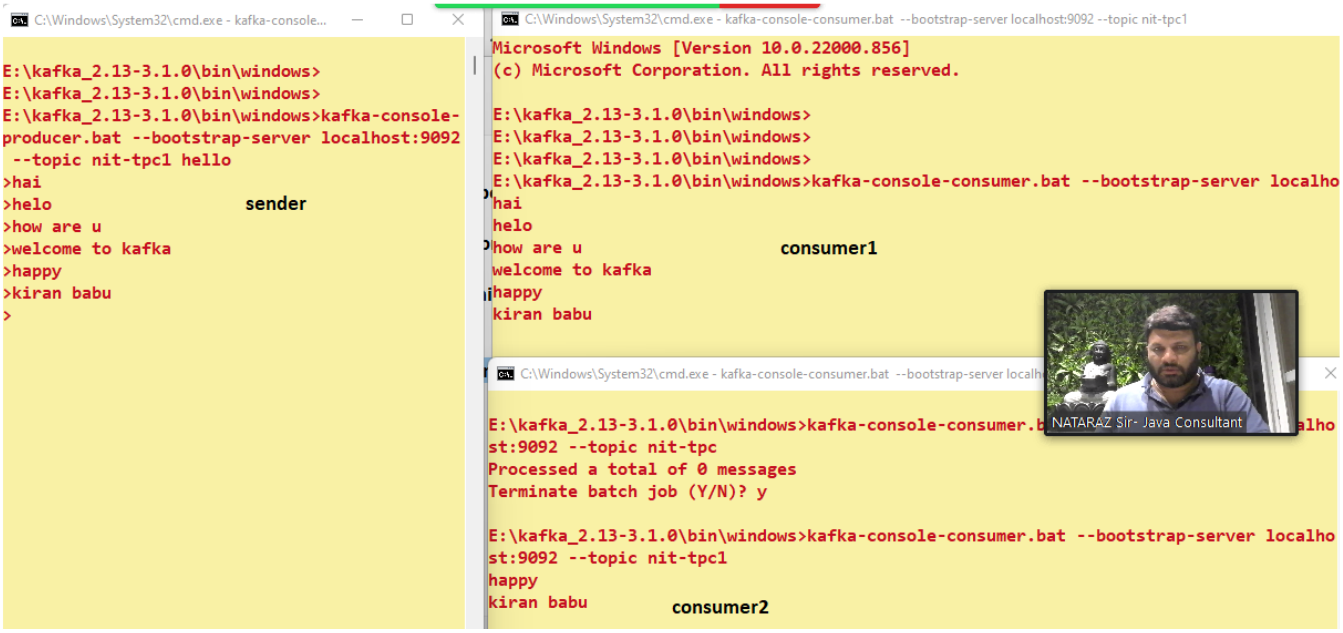
messages

```
>hai
>123
```

send these messages
after starting the consumer

step5) create a consumer and link with message broker

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-console-consumer.bat --bootstrap-server
                                localhost:9092 --topic nit-tpc
hello
hai
123
```



51.NTSPBMS615- Developing Kafka Clients using Java APIs - sep9th-2022

Developing Kafka Client Apps using Java API to

=> Both Producer /Sender and reciver/Consumer Apps that are taking kafka software are called kafka Client Apps.

kafka Client App Sender App/ Consumer App

Producer /Sender App Development (Legacy Style - spring style) with

=> First we need to establish a link between Producer App and kafka setup (mainly bookstrap server) by using the following details /properties

#1
bootstrap-servers = localhost:9092
key-serializer = pkg.StrngSerializer
value-serializer = pkg.StringDeSerializer

(Kafka Bootstarp server default port is 9092)
we generally keep this info in Properties class obj we develope app in legacy style (not spring boot style)

java.util.Properties is the sub class of jav.util.Hashtable which is a map collection allowing only Strings in the keys and values of the elements

note1:: any type of data /message given by Producer will be converted to String message/data using StringSerializer class that is specified above. if we give java object as message/ data then it will be converted into json String content .

KafkaProducer<String,String> producer=new KafkaProducer<String,Stirng>(props);

=> Take the support of "ProducerRecord" class to sepcify Message object details and topic name

#2 ProduderRecord<String,String> record=new ProducerRecord("topicname", message object);
(ProducerRecord class obj indirectly represeing message/data to send from producer App to topic of kafks setup)

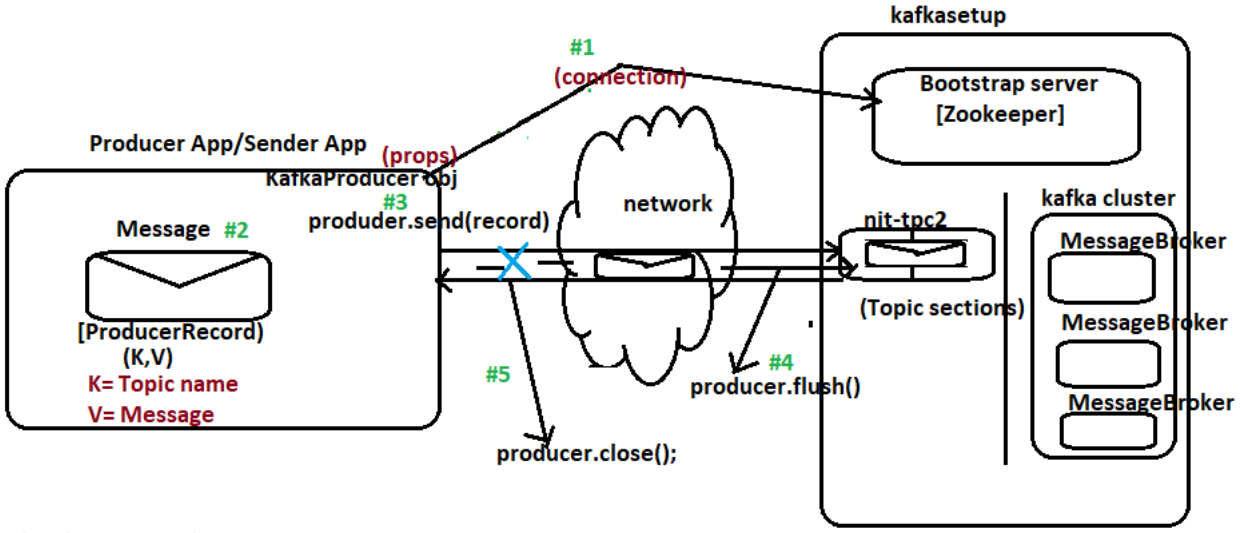
=> To send Message (ProducerRecord) created in the Producer App we need to use KafkaProducer<String,String> class as shown below

KafkaProducer<String,String> producer=new KafkaProducer<>(props);

#3 producer.send(record); //puts data in connection stream in queue

#4 producer.flush(); //sends data to Topic based on the given topic name

#5 producer.close(); // Closes the link b/w producer and kafks setup (Bootstarp server)

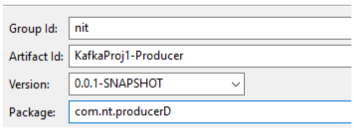


Code developnent and execution

step1) create maven project (not sprng boot starter Project) using maven-archetype-quickstart

adding the following dependencies (kafkaclients , slf4j-simple , jackson-dataformat-xml)

File menu ---> maven project ---> next --->
select maven-archetype-quickstart --->next -->



In pom.xml

```
<!-- https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients -->
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>3.1.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>1.7.35</version>
    <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml -->
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
    <version>2.13.1</version>
</dependency>
```

step2) Develop the Producer App

```
package com.nt.test;
import java.util.Properties;
import java.util.Scanner;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.StringSerializer;

public class KafkaSenderApp {
        public static void main(String[] args) {
          //prepare Kafka Properties using java.util.Properties class obj
              Properties props=new Properties();
              props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
              props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,StringSerializer.class.getName());
              props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,StringSerializer.class.getName());
              //create KafkaProducer object
              KafkaProducer<String, String> producer=new KafkaProducer<String, String>(props);

          ProducerRecord<String,String> record=null;
          String msg=null;
          do {
                Scanner  sc=new Scanner(System.in);
                System.out.println("Enter message::");
   msg=sc.nextLine();
         //  Create ProducerRecord object having topic name and Message
          record=new ProducerRecord<String, String>("nit-tp1", msg);
          // send the message
           producer.send(record);
           //flush the message
           producer.flush();
          }
          while(!msg.equalsIgnoreCase("stop"));
                        //close the link with  Bootstrap server..
           producer.close();
    }

}
```

step3) Execute things in the following order

a) start zookeeper b) start kafka setup c) create topic d) Run the above Producer App
e)start Consumer (readmade)

a)start zookeeper

```
E:\kafka_2.13-3.1.0\bin\windows>zookeeper-server-start.bat E:\kafka_2.13-3.1.0\config\zookeeper.properties
```

b) start kafka setup

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-server-start.bat E:\kafka_2.13-3.1.0\config\server.properties
```

c) create Topic and topic sections

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-topics --create --bootstrap-server localhost:9092
                                --topic nit-tpc-feb --replication-factor 1 --partitions 1
  Created topic nit-tpc-feb.
```

d) start the consumer App

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-console-consumer
                                --bootstrap-server localhost:9092 --topic nit-tpc-feb
WELcome to Apache kafkas messaging
```

e) run producer App from eclise ide

sends message

Developing kafka consumer as kafka client in legacy style (spring style)

#1] We need to create communication link b/w kafka setup and consumer(s) by supplying multiple connection details as key=value pairs in the form of Properties object

```
bootstrap-server = localhost:9092
key-deserializer = StringDeSerializer
value-deserilizer = StringDeSerializer
groupid = grp-listeners [anything can be given here]   [optional]
```

(given by kafka api)

=> if multiple consumers are taken reading same message from topic thorugh message broker then grouping multiple consumers to single group by providing groupid make replicate /cloning operation on messages faster. (Performence will be improved)

#2] create KafkaConsumer class object specifying the above connection properties to get link /connection between consumer and kafka setup (Indirectly with Bootstarp server (zookeeper) that manages Topic memories and message brokers.

```
KafkaConsumer<String,String> consumer=new KakfaConsumer<>(props);
```

java.util.Properties class obj having the above connection properties

#3] Link Consumer with MessageBroker by specifying the topic name
[Here MessageBroker will be created dynamically on 1 per Consumer basis)

```
consumer.subscribe(Arrays.asList("nit-tpc-feb"));   // we can give multiple topic names
                                                    // to read messages from muliple topic sections
```

#4] Do polling (pinging the message broker continuosly having certain gap) wth Message broker the to check message(s) avaiable or not and read message.

Expected process :: (but not possible in legact style of kakfa consumer development)

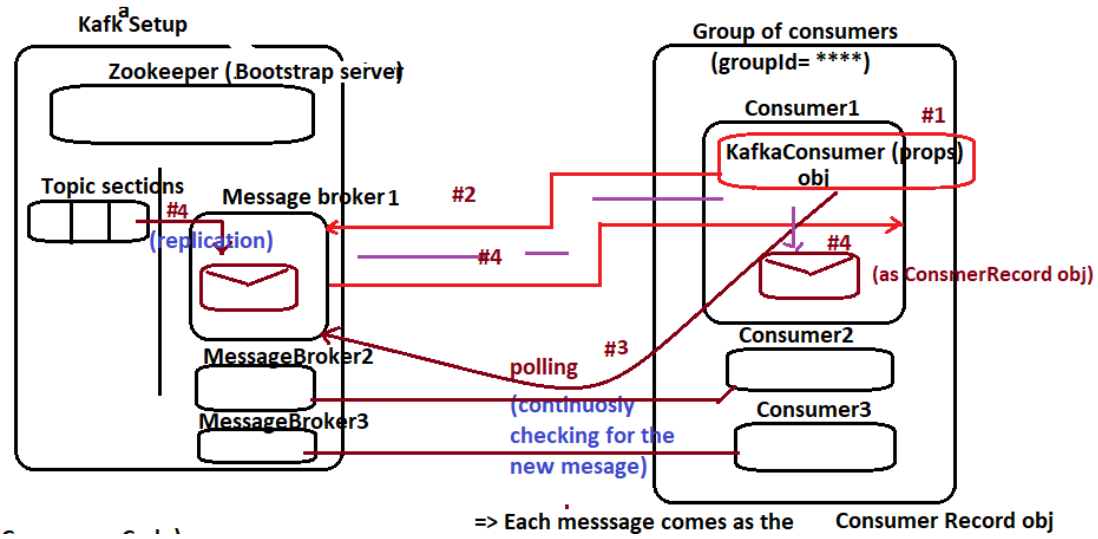
On arrival of message to topic memory , the Message broker reads the messages and sends the message to all the subscribed consumers automatically .. but In legacy style kafka consumer devleopment that is not happening . So polling has to be done

Actual process
==========

=> Consumer pings the Message Broker having scheduling (having certain continuos time gap) for new message .. if came to topic memory then the message broker gives that message to message consumer (this process is polling)

```
while(true){  //infinite loop

 ConsumerRecords<String,Stirng> records=
                    consumer.poll(Duration.ofMillis(1000));

  for(ConsumerRecord<String,String> record: records){
       System.out.println("message is ::"+record.value());
  }//for
 } //while
```

=>Each consumed message will be represented by ConsumerRecord object
Multiple comsumed messages will be represented by ConsumerRecords object.



=> Each message comes as the ConsumerRecord obj

Code (Kafka Consumer Code)
==========

step1) create maven project adding the following dependencies
kafka clients , slf4j-simple , jacson-data format-xml (same producer App)

```
<!-- https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients -->
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-clients</artifactId>
  <version>3.1.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.7.35</version>
  <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml -->
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
  <version>2.13.1</version>
</dependency>
```

step2) Develop the Consumer App

MessageConsumer.java

```
package com.nt.consumer;

import java.time.Duration;
import java.util.Arrays;
import java.util.Properties;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.serialization.StringDeserializer;

public class MessageConsumer {

    public static void main(String[] args) {
        // create Connection properties as K=V in java.util.Properties class obj
        Properties props=new Properties();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,StringDeserializer.class.getName());
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,StringDeserializer.class.getName());
        props.put(ConsumerConfig.GROUP_ID_CONFIG,"grp1_consumers");

      //create KafkaConsumer object
      KafkaConsumer<String, String> consumer=new KafkaConsumer<String, String>(props);
      //Subscribe to topic Destination through MEssagebroker
        consumer.subscribe(Arrays.asList("nit-tpc-feb"));
       //Performing polling to check and read the messages
        while(true) {
            //poll and get consumer records (messages)

         ConsumerRecords<String,String> records=consumer.poll(Duration.ofMillis(2000));
          // read and display messages
           for(ConsumerRecord<String,String> record:records) {
                System.out.println("message is ::"+record.value());
```

```
                }//for
            }//while
        }
}
```

step3) Run The services in the following order

a) start zookeeper as boostrap server

```
E:\kafka_2.13-3.1.0\bin\windows>zookeeper-server-start.bat   E:\kafka_2.13-3.1.0\config\zookeeper.properties
```

b) start apache kafka setup

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-server-start.bat    E:\kafka_2.13-3.1.0\config\server.properties
```

c) note:: since Topic name "nit-tpc-feb" is already created.. So we need not to create again

d) Run the consumer App using eclipse

e) Run the Producer App using eclipse (previous class App)

Spring Boot + Apache Kafka API

=> Spring Boot gives built-in support for apache kafka.. It even gives certain objects automatically through AutoConfiguration process are added

=> if spring -boot- starter-apache-kafka depenecies to app then we get KafkaTemplate<K,V> class object through autoConfiguration.. which internally takes care of creating KafkaProducer, ProducerRecord objects that are required to send messages/data.

note: KafkaTemplate<K,V> is like JdbcTemplate,JMSTemplate, MongoTemplate, RestTemplate and etc.. which is given based on TemplateMethod DP

In Producer class or comp

```
@Autowired
private    KafkaTemplate<String,String>  template;
```

=> we can place @KafkaListener(topicName="....." , groupId=".....") annotation on the method of Listener class to make Message broker to collect the message recived to topic section i.e it internally takes care of creating KafkaConsumer<String,String> obj and ConsumerRecord object.

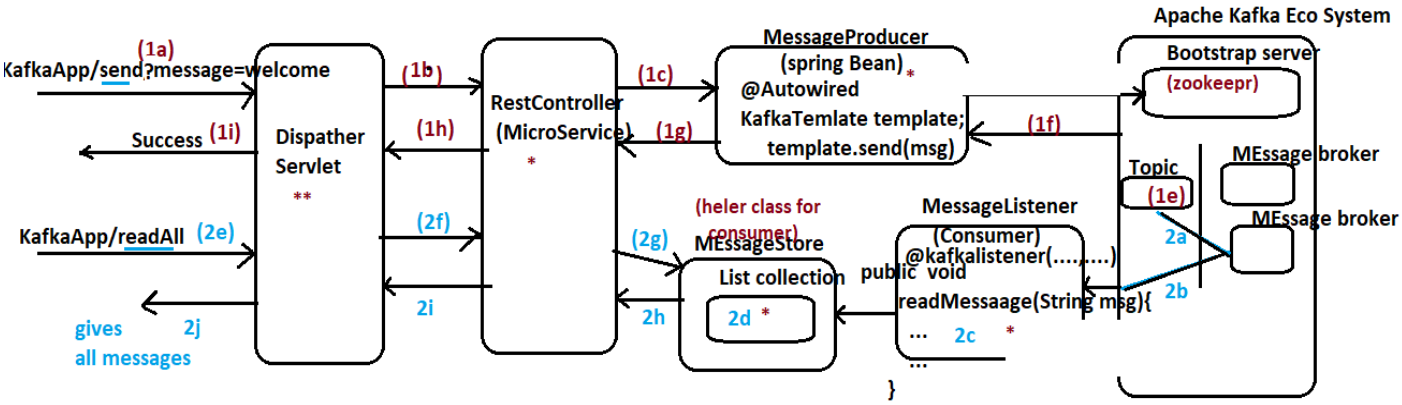
In Listener class

```
@KafkaListener(topicName="nit-tpc-fri", groupId="nit-grp1")
public  void   readMessage(String msg){
    ....
     ....  //logic to read the message.
  }
```

=> We need to add @EnableKafka on the top of starter class (main class)
(this will created Topic sections dynamically with the given names)

=> we can get message from endusers as request parameter through RestController/MicorService of spring MVC/Spring Rest App , So we need to make the Producer App taking message from through RestController to send to kafka setup.
Similarly we need to read the message from kafka setup using Consumer or MEssageListener to send to endusers through RestController.

http://localhost:4041/KafkaApp/send?message=welcome
http://localhost:4041/KafkaApp/send?message=quotation value 2000
http://localhost:4041/KafkaApp/send?message=how are u



(1a) --- 1i :: 1st request -response (for sending messages)
(2a) --- 2j :: 2nd request-response (To reeciving messages all togather)

* -->user-defined development
** -->pre-defined class

order of development
==========

of

=>MessageProducer having injection KafkaTemplate obj
=> Restcontroller with handler method with "/send" request path having injection of MessageProducer object

for Sending messages

=> MessageStore
=>MessageListner injected with MessageStore
=> above Restcontroller with another handler method with "/readAll" request path having injection of MessageStore object

For recieving all messages togather

step1) Create spring boot starter Project adding the following dependencies web , kafka , lombok api, devtools

step2) add @Enablekafka on the top of main class/starter class.

```
@SpringBootApplication
@EnableKafka
public class BootKafkaProj2RestWithKafkaApplication {

	public static void main(String[] args) {
		SpringApplication.run(BootKafkaProj2RestWithKafkaApplication.class, args);
	}
}
```

step3) add the following properties in application.properties file

applicaiton.properties

```
#server port
server.port=4041
#context path
server.servlet.context-path=/RestKafkaApp
#topic name
app.topic.name=nit-tpc-sat1
#Producer properites
spring.kafka.producer.bootstrap-servers=localhost:9092
spring.kafka.producer.key-serializer=org.apache.kafka.common.serialization.StringSerializer
spring.kafka.producer.value-serializer=org.apache.kafka.common.serialization.StringSerializer

#Consumer properites
spring.kafka.consumer.bootstrap-servers=localhost:9092
spring.kafka.consumer.key-deserializer=org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.consumer.value-deserializer=org.apache.kafka.common.serialization.StringDeserializer
```

step4) MessageProducer.java

```
package com.nt.producer;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Component;

@Component("msgProducer")
public class MessageProducer {
	@Autowired
	private KafkaTemplate<String,String>  template;
	@Value("${app.topic.name}")
	private  String topicName;

	public  String   sendMessage(String message) {
		template.send(topicName,message);
		return " message delivered ";
	}
}
```

step5) Restcontroller

```
package com.nt.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.nt.consumer.MessageStore;
import com.nt.producer.MessageProducer;


@RestController
public class KafkaMessageHandlingController {
	@Autowired
	private  MessageProducer producer;
	@Autowired
	private MessageStore  store;

	@GetMapping("/send")
	public  String   sendMessage(@RequestParam("message")String message) {
		String status=producer.sendMessage(message);
		return "<h1>"+status+"</h1>";
	}

	@GetMapping("/readAll")
	public   String   fetchAllMessage() {
	  return  "<h1>"+store.getAllMessages()+"</h1>";
	}
}
```

step6) MessageStore.java

```
==================
package com.nt.consumer;
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Component;

@Component
public class MessageStore {
	private List<String> listMessages=new ArrayList();

	public void   addMessage(String message) {
		listMessages.add(message);
	}

	public   String   getAllMessages() {
		return  listMessages.toString();
	}
}
```

step7) MessageConsumer.java

```
package com.nt.consumer;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Component;

@Component
public class MessageConsumer {
	@Autowired
	private MessageStore store;

	@KafkaListener(topics = "${app.topic.name}",groupId = "grp1")
	public void  readMessage(String message) {
		//add  message to store
		store.addMessage(message);
	}

}
```

Exection order

i) start bootstrap server (zookeeper)

```
E:\kafka_2.13-3.1.0\bin\windows>zookeeper-server-start.bat E:\kafka_2.13-3.1.0\config\zookeeper.properties
```

ii) start kafka server setup

```
E:\kafka_2.13-3.1.0\bin\windows>kafka-server-start.bat E:\kafka_2.13-3.1.0\config\server.properties
```

note:: No need of creating topic seperately .. @EnableKakfa will take care of creating topic dynamically

iii) Run the application as spring boot App or on server
and give requests

```
http://localhost:4041/RestkafkaApp/send?messsage=raja
http://localhost:4041/RestkafkaApp/send?messsage=rani
http://localhost:4041/RestkafkaApp/send?messsage=hello
```

for sending messages

```
http://localhost:4041/RestkafkaApp/readAll
 [raja,rani,hello]
```

for reading messages

52.NTSPBMS615- MiniProject using MicroServices -sept13th-2022

Mini Project using MicroServices (Spring Cloud)

==========

=>EurekaServer (Project1) --> R & D Server (Registry and Discovery Server)
=>Zull Proxy Server (Project2) --> API Gateway server
=>Admin server (Project3) ---> Contains spring boot Actuators
=>IPLPlayer SErvice (Project4) --- > MicroService
=>IPLTeam Service (Project5) --> Microservice
=>ConfingServer (Project6) ---> To collect jdbc properties from GITUB/GITLab

MicroSErvices in Communication using spring data jpa based persistence operations

Project1:: (EurekServer)

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

Name :: SpringBootMsProj10 -MiniProject-EurekaServer
Dependencies :: EurekServer

At Main class :: @EnableEurekaServer

application.properties

```
#Server Port number
server.port=8761

# Registration details
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

ConfigServer (Project6)

==========

Name :: SpringBootMSProj10-MiniProject-ConfigServer
Dependencies :: ConfigServer

AT main/starter class :: @EnableConfigServer

in application.properties

```
#Server port
server.port=8888
spring.cloud.config.server.git.uri=https://gitlab.com/nataraz/csinfoproj.git
```

In GItLab -- create CsInfoProj -->
add applicaiton.properties having the following information

```
#DataSource Properties
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=manager

# JPA Properties
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.Oracle10gDialect
```

AdminServer (Project3)

==========

name :: SpringBootMSProj10-MiniProject-AdminServer
dependencies :: web , spring boot actuators , admin server

At starter class :: @EnableAdminServer

in application.properties

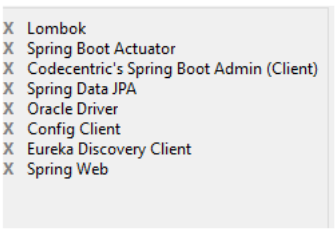
---

server.port=9999

MicroService#1 (Project5) ***(Producer Ms)***

==========

Name :: SpringBootMSProj10-MiniProject-IPLTeamService



At starter class :: @EnableEurekaClient ,

in application.properties

```
#server port
server.port=8081
# MS name
spring.application.name=TEAM-SERVICE
#To register with EurekaServer
eureka.client.service-url.default-zone=http://localhost:8761/eureka
#To provide random instance id
eureka.instance.instance-id=${spring.application.name}:${random.value}
#To get all actuators
management.endpoints.web.exposure.include=*

#To link MS with  Admin server
spring.boot.admin.client.url=http://localhost:9999/

#To link with config server
spring.config.import=optional:configserver:

##Logging Information
logging.file.name=team_info.log
#logging.file.max-size=15MB    //deprecated
#logging.file.clean-history-on-start=true  //deprecated
logging.logback.rollingpolicy.max-file-size=15MB
logging.logback.rollingpolicy.clean-history-on-start=true
logging.level.root=info
```

bootstrap.properties

```
spring.cloud.config.uri=http://localhost:8888
```

note::

```
@Slf4j
public class LogExample {
}
```

will generate:

```
public class LogExample {
   private static final org.slf4j.Logger log = org.slf4j.LoggerFactory.getLogger(LogExample.class);
}
```

//write u r ms logics like model class , repository(I) , serivice interface, service impl class, controller class

...
...
..

IPL Player MS (Project 4) Cosumer Service

Name :: SpringBootMSProj10-MiniProject-IPLPlayerService

Dependencies::

- [x] Lombok
- [x] Spring Boot Actuator
- [x] Codecentric's Spring Boot Admin (Client)
- [x] Spring Data JPA
- [x] Oracle Driver
- [x] Config Client
- [x] Eureka Discovery Client
- [x] Spring Web
- [x] OpenFeign

AT Stater/main class :: @EnableFeignClients

=>Develop model class, repository (I) , service class , consumer class (feign client) ,RestController and etc..

....
... ...

ZuulProxyServer (API Gateway)

=> use spring boot version 2.3.6.RELEASE
=> name :: SpringBootMSProj10-MiniProject-ZuulGateway
=> in pom.xml add the following properties in <properites> tag

```
<properties>
    <java.version>11</java.version>
    <spring-cloud.version>Hoxton.SR9</spring-cloud.version>
</properties>
```

=>dependencies :: web , eureke discovery client , netflix zuul
(collect from mvnrepository.com)

=> At starter class or Main class :: @EnableZuulProxy @EnableEurekaClient

=>In application.properties

```
#zuul server port
server.port=9100
# application name
spring.application.name=ZUUL-Server
#register with Eureka server
eureka.client.service-url.default-zone=http://localhost:8761/eureka
#Provide  common url /Gateway url
zuul.routes.player.path=/player-api/**
zuul.routes.player.service-id=PLAYER-SERVICE
```

Order of execution

==========

=>Run Eurekserver App

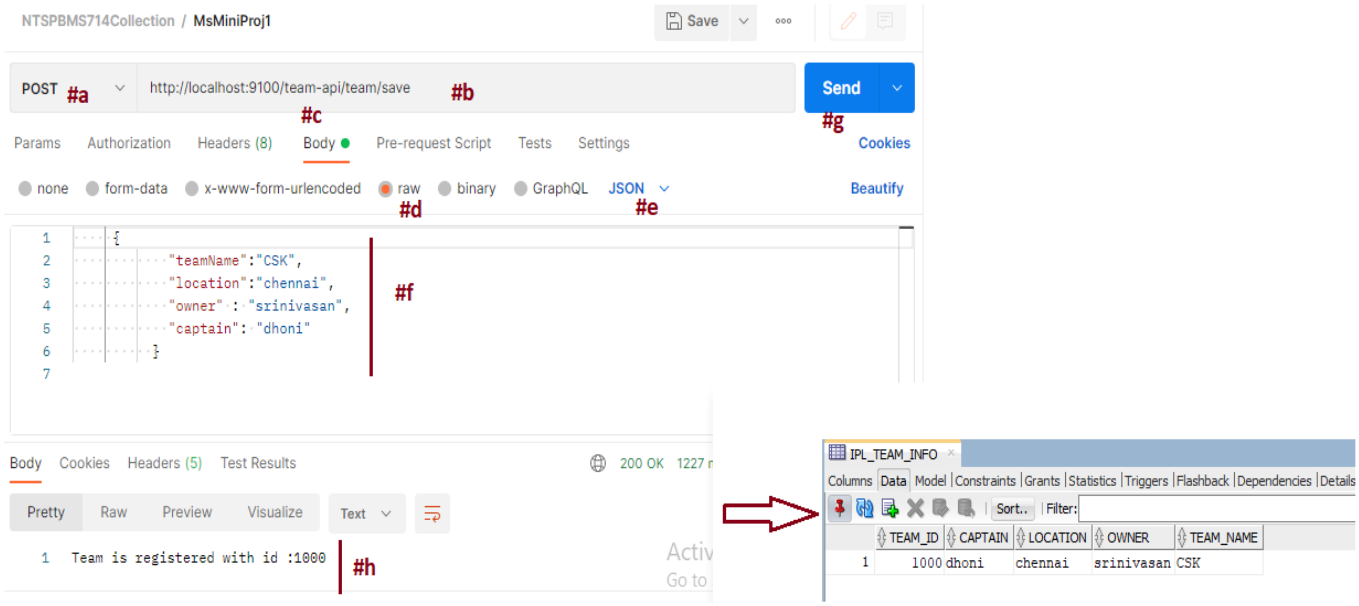
=>run Config Server App

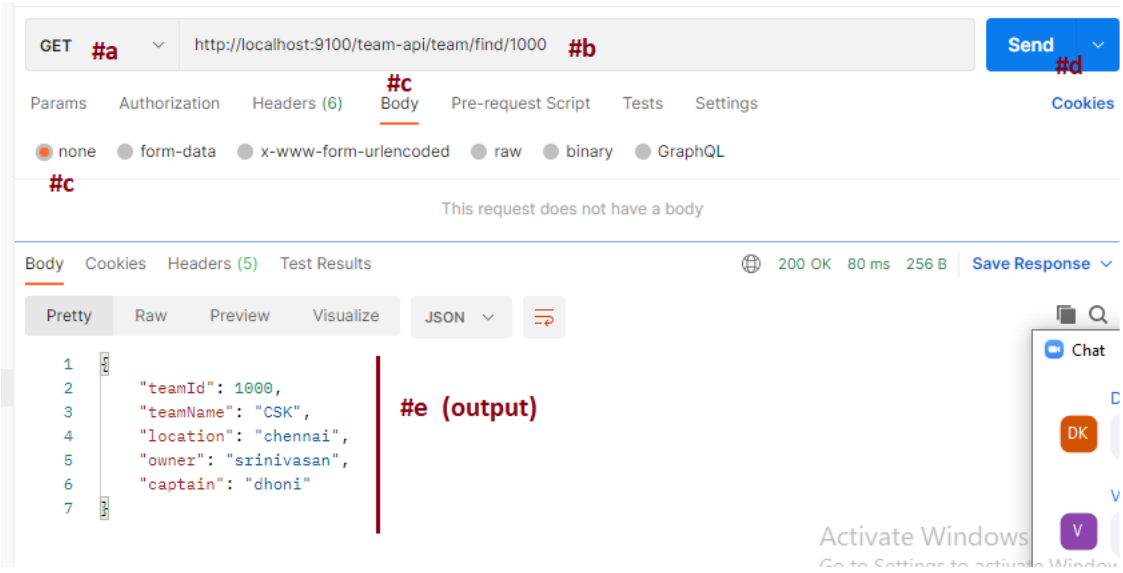
=> run admin Server App

=> Run Product Ms (TeamService)

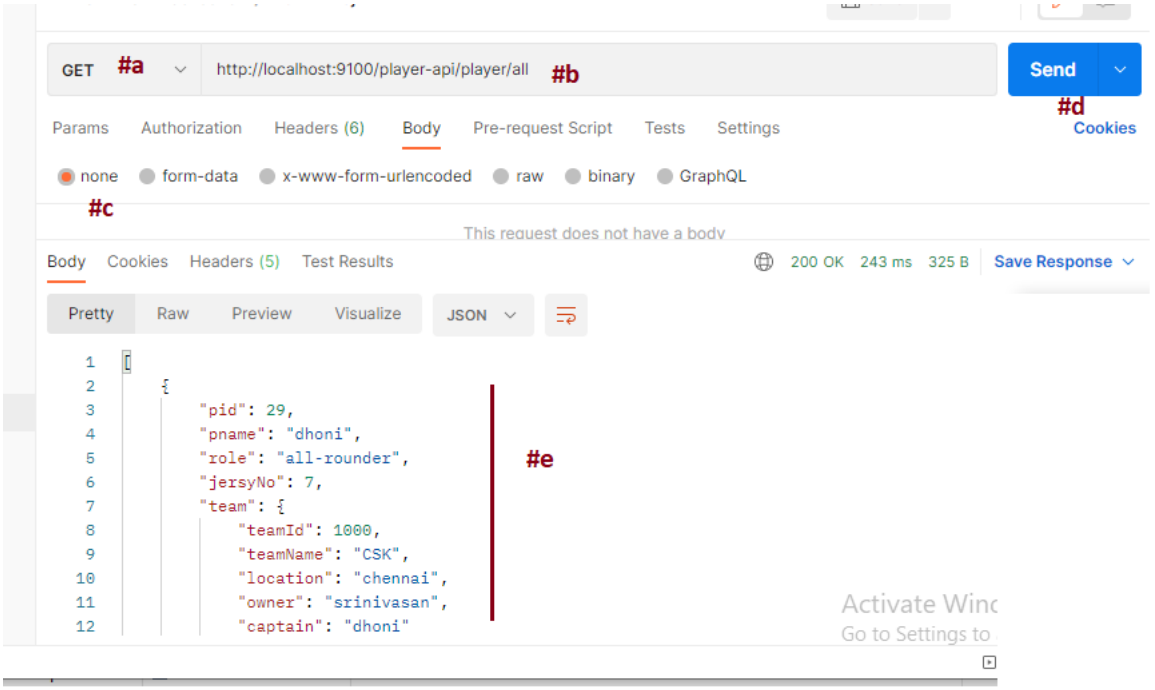
=> Run Cosumer MS (PlayerService)

=> Run ZullServer






POST
http://localhost:9100/player-api/player/save
#b
Send
#g
#a
#c
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
#d
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
#e
{
"pname":"dhoni",
"role" : "all-rounder",
"jersyNo": 7,
"team" : {
"teamId": 1000
}
}
#f
Body
Cookies
Headers (5)
Test Results
200 OK
313 ms
230 B
Pretty
Raw
Preview
Visualize
Text
Player and his team details saved having the player id::29
#h


GET
#a
http://localhost:9100/player-api/player/all
#b
Send
#d
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
#c
This request does not have a body
Body
Cookies
Headers (5)
Test Results
200 OK
243 ms
325 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"pid": 29,
"pname": "dhoni",
"role": "all-rounder",
"jersyNo": 7,
"team": {
"teamId": 1000,
"teamName": "CSK",
"location": "chennai",
"owner": "srinivasan",
"captain": "dhoni"
#e
Activate Winc
Go to Settings to