Testing Provider /Producer /Rest API App/service with POSTMAN tool

note:: since ResponseEntity<String> is taken and handler method type is "GET" (in this controller) we can send that request even from browser (otherwise not possible from browser)

Using browser (Not recomanded to use becoz it can send only GET or POST mode requests where as @RestController can have GET ,POST, DELETE,PUT,PATCH and etc.. modes handler methods)

a) Run the Application using Run as server option
b) use the following from the browser

TO overcome this problem, take the support of POSTMAN tool

Using POSTMAN Tool (Recomanded to use) (A good tool to Test the Restfull web services /service providers developed in any language/technology /framework having capability generate different modes of request and ability recieve text/JSON/XML content based response .can send and recieve inputs in diffrent formats like text or XML or JSON and etc..)

a) download and install postman tool (by skipping the registration)

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

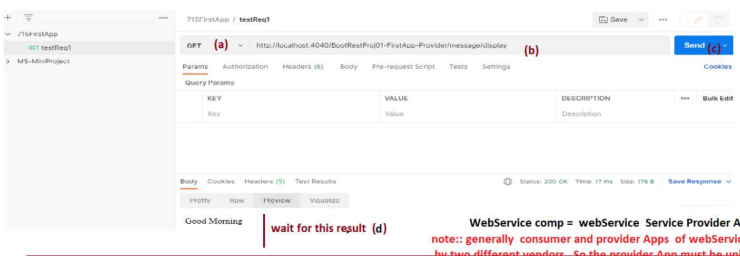
b) Create a new Collection
=> Postman home page ===> skip registration and go to the app (look at botton the page)===> use (+) symbol to create the collection.

note: In Postman collection is a tool where set of test requests are placed

c) Send request by using the request url

=>POSTMAN just provides Env.. to test the Service Provider App
=> Swagger provides Test env.. + document about the Service Provider App

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



wait for this result (d)

=> POSTMAN UnitTesting tool for Restfull webSErvice comp developers
=> In POSTMAN , we keep track all the tests that are done with different inputs and instructions by adding them to collection

WebService comp = webService Service Provider App
note:: generally consumer and provider Apps of webService are developed by two different vendors , So the provider App must be unit tested using POSTMAN by Provider App developers.

Q) What is the difference between @Controller and @ResstController ?

@Controller	@RestController
a) Given in the spring 2.5 version (bit old annotation)	a) Given in the version spring 4.0 (relatively new annotation)
b) Specialization of @Component	b) Specialization of @Controller
c) makes the java class as webController class having capability to handle http requests by taking them through DispatcherServlet	c) makes the java class as Rest Controller or Restfull service provider class having capability to handle http requests by taking them through DispatcherServlet
d) Can be used in both spring MVC and spring Rest Apps	d) Recomanded to use only in spring Rest Apps (Restful Apps)
e) Every Handler method does not get @ResponseBody automtically, So we need to add it expicitly if needed	e) Every handle method automatically gets @ReponseBody So there is no need adding it seperately
f) Based on the return type of handler method it decides wheather view comp should be invovled or not (if the return type is otherthan ResponseEntity<T> then it involves ViewResolver and view comp)	f) Makes the handler method sending its output/results as response to client directly through DispagcherServlet without involving ViewResolver and View comp

@RestController=@Controller + @ResponseBody

browser ---> flipkart.com -----> gpay
(web applicaiton) (websevice comp)
(@Controller) (@RestController)

note:: Instead of comparing @RestController with @Controller .. plese use it as convient annotation given over @Controller providing easyness to develope Restfull Service providers (@Contrller + @ResponseBody)

Http request methods/modes

- GET
- POST
- PUT
- DELETE
- OPTIONS
- TRACE
- PATCH
- HEAD
- CONNECT (Resrved for Future)

Generally we use the following Http request methods/modes in Restful applications while performing CURD Operations

GET ---> for Read Operations (R) ->selecting records
POST ---> for Create opertions (C) -> inserting records
DELETE ---> for DELETE opertions (D) -> deleting records
PUT ---> for Update opertions (U) -> modifying records (complete record modification)
PATCH ---> for Update opertions (U) -> modifying records (partial record modification)

note: PUT for complete modification of the record.
PATCH for partial modification of the record.

```
public <RT> updateEmployee(Employee emp){
   ....
   ...
}
```
use PUT mode request

```
public <RT> updateEmployeeEmail(String newEmail){
   ....
   ...
}
```
use PATCH mode request

related

=>Since HEAD request mode HttpResponse does not contain body /output/results , So it can not be used for Read /Select Opertions . This can be used only for Debugging especially to check whether certain handdler method is there or not
=> Since TRACE is given to trace/debug componets involved for the SUCCESS or FAILURE of request processing so it can not be used for CURD Operations
=>Since OPTIONS mode request gives the possible Http request methods/modes that can be used to generate request to web comp .. its HttpResponse contains purely list of modes/methods that are possible to give request to web comp.. So it also can not be used for CURD Operations

```
@WebServlet("/testurl")
public class TestServlet{

  public void doGet(req,res){
     ...
     ...
  }
}
```
if we give OPTIONS mode request to this web comp we get response as
Allow : GET,HEAD,OPTIONS

```
@WebServlet("/testurl")
public class TestServlet{

  public void doPost(req,res){
     ...
     ...
  }
}
```
if we give OPTIONS mode request to this web comp we get response as
Allow : POST,OPTIONS,

In @RestController we can use the follwing annotation for handler/operation methods
@GetMapping , @PostMapping, @DeleteMapping
@PutMapping, @PatchMapping

@OptionsMapping,
@HeadMapping
@TraceMapping
annotations are not there

=>if the request url is not valid then we get 404 error (requested resource not found)
=> if the request url based web comp is not ready to process given mode request then we get 405 error (method not allowed)
=> if the request fails in Authentication then we get 401 error
=> if the request fails in Authorization then we get 403 error (resource is forbidden)
=> if the web comp(servlet/jsp/producer) fails to instantiate for the given request then we get 500 error

=>if the return type of producure method is other than <String> generic in ResponseEntity object then the producer methods send JSON data along with the HttpResponse body

=>if the return type of producer method is <String> generic in ResponseEntity object then the producer methods sends text data along with the HttpResponse body

Every Restfull application /Project contains
a) server App /prodcuder App/ Service provider App (spring MVC App with @RestController with methods) (also called Rest API)
[The request url of @RestController and other required infomation for sending request are called End points]

b) Client App /Service Cosumer/ Consumer App

- Angular
- ReactJS
- PHP
- IOS
- IOT Devices
- .net
- python
- andriod

(Programable client Apps)

POSTMAN, Swagger | Tools for Testing

spring RestTemplate (spring basedClient) (Programmable Client Apps)

request headers vs req pameters (query params)

=>req headers are Client generated inputs that go along with request automatically having fixed names(header names)
eg:: accept , accept-language, user-agent,referer ,contentType and etc.
These request headers carry technical data along with the request given by client/ cosumer app

=>req params are enduser supplied values as query String /form data ..req param names are not fixed .. and they are user-defined
sno=101&sname=raja&sadd=hyd
req params /query params
request params carry non-technical data given by endusers/visitors

==> API development means Developing Spring Rest Server App/Service provider App

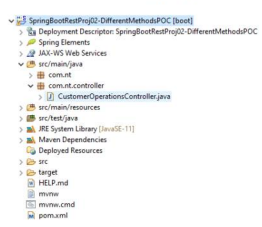
=> Giving API End points means providing request url and other related information to developers for developing client Apps/service consumer Apps for cosuming the services offered by Service provider

note:: The API/ services developed in SOAP based webservices can not consumed using Rest Client and vice-versa.

note: The Restful webSErvices developed using one kind of Rest API/framework (like jesry /spring rest, ja x-RS,Restlet and etc..) can be consumed using same Rest API or different Rest APIS (becoz both are in Restfull webService env..)

eg:: There are multiple open (public apis) /free apis /service providers devleoped in different technologes/frameworks can be of Restfull webservices and they consumed in our Apps using our choice rest apis..

eg:: weather report api
Google Maps api
Gpay APIs
ICC API
Covid APIs
paypall APIs



RestController class

```java
package com.nt.controller;

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

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

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

    }

    @PostMapping("/register")
    public ResponseEntity<String> registerCustomer(){
        return new ResponseEntity<String>("From POST-RegisterCustomer Method" , HttpStatus.OK);

    }

    @PutMapping("/modify")
    public ResponseEntity<String> updateCustomer(){
        return new ResponseEntity<String>("From PUT-UpdateCustomer() Method" , HttpStatus.OK);

    }

    @PatchMapping("/pmodify")
    public ResponseEntity<String> updateCustomerByNo(){
        return new ResponseEntity<String>("From PATH-UpdateCustomerByNo() Method" , HttpStatus.OK);

    }

    @DeleteMapping("/delete")
    public ResponseEntity<String> deleteCustomer(){
        return new ResponseEntity<String>("From DELETE-deleteCustomer Method" , HttpStatus.OK);

    }


}
```

GET http://localhost:3030/SpringBootRestProj02-DifferentMethodsPOC/customer/report Send

POST http://localhost:3030/SpringBootRestProj02-DifferentMethodsPOC/customer/register Send

DELETE http://localhost:3030/SpringBootRestProj02-DifferentMethodsPOC/customer/delete Send

PUT http://localhost:3030/SpringBootRestProj02-DifferentMethodsPOC/customer/modify Send

PATCH http://localhost:3030/SpringBootRestProj02-DifferentMethodsPOC/customer/pmodify Send