SpringRest = SpringMVC++

=> Developing SpringMVC Handlermethods of Controller class without returning the logical viewName and making it to return
ResponseEntity<T> object would make the handler method ready for SpringRestEnvironment.(infact it becomes indirectly buisness method of RestFul WebServices)

=> To send response to the browser/client without involving view, we need to place @ResponseBody on the top of handler method in @Controller class.

=> Instead of Keeping 2 annotations like
- a. @GetMapping
- b. @ResponseBody

=> We can attain the same functionality using single annotation called "@RestController".

@RestController = @Controller + @ResponseBody

eg#1.

```
  @Controller
  @RequestMapping("/customer")
  public class CustomerController{

      @ResponseBody
      @GetMapping("/display")
      public ResponseEntity<String> displayMessage(){
            return object of ResponseEntity
      }
  }
```

eg#2.

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


      @GetMapping("/display")
      public ResponseEntity<String> displayMessage(){
            return object of ResponseEntity
      }
  }
```

Note:
In SpringMVC/SpringRest the configuration takes palce automatically
- a. DispatcherServlet would act like a frontcontroller with "/" url pattern.
- b. HandlerMapping is configured autoamtically(Default is RequestMappingHandlerMapping)
- c. ErrorFilters display the white label error pages.
- d. ViewResolver is configured autoamtically(Default is InternalResourceViewResolver)

ResponseEntity<T> object contains 2 parts
- a. ResponseBody(String/object/Collection)
- b. ResponseStatus code(We use ResponseStatus enum constants like

ResponseStatus.OK,,....)

ResponseStatus
      -> it indicates that the generated response/result/ouput should be given to client as success(200-299) or clientside error(400-499)
          or servererror(500-599)
ResponseEntity
        -> It indicates that the generated ouptut/result should go to client/browser directly through DispatcherServlet without involving
         ViewResolver and ViewComponents.
      -> If ResponseEntity object is having other than <String> type as generic type(Object/collection) then internally it converts and holds
           the output as "JSON(key-value) pair",where as for "String" type it contains normal text.

FirstRestApI
++++++++++++
pom.xml

```xml
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

MessageRenderController.java
++++++++++++++++++++++++++++

```java
package in.pwskills.nitin.restcontroller;

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("/message")
public class MessageRenderController {

      @GetMapping("/generate")
      public ResponseEntity<String> generateMessage() {

            LocalDateTime ldt = LocalDateTime.now();

            int hour = ldt.getHour();

            String body = null;

            if (hour < 12) {
                  body = "Good Morning";
            } else if (hour < 16) {
                  body = "Good Afternoon";
            } else if (hour < 20) {
                  body = "Good Evening";
            } else {
                  body = "Good Night";
            }
            HttpStatus status = HttpStatus.OK;

            ResponseEntity<String> entity = new ResponseEntity<>(body, status);
            return entity;

      }

}
```

```
application.properties
  server.port = 9999
```

Run the application and send the request to the api
      http://localhost:9999/generate/message

Response(200-OK)
  Good Evening.

+++++++++++++++++++++++
Working with JSON Data
+++++++++++++++++++++++
HTTPRequest Methods are
  a. GET
  b. POST
  c. PUT
  d. DELETE
  e. OPTIONS
  f. TRACE
  g. PATCH
  h. HEAD

Generally we use the following HTTPRequest methods/mode in RestFul application while performing CURD operations

```
      GET    ---> for Read operation   ---> Selecting Records
        POST   ---> for Create Operation ---> Creating Record
        PUT    ---> for Update Operation ---> Updating the Record
        DELETE ---> for Delete Operation ---> Deleting the Record
```

HEAD request mode HttpResponse does not contains body/output/results, so it can't be used for select operations.
Trace request is given to trace/debug the components involved for SUCCESS of FAILURE of request processing so can't be used for curd operation
OPTION request mode request gives the possible HTTP request methods/modes that can be used to generate the request to the webcomponent.

Note:
  From the browser we can send only request for "GetMapping", to send the request to the application in other modes we need to use
  a tool called "POSTMAN".

SecondRestApi
+++++++++++++
pom.xml

```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

```
++++++++++++++++++++++++++++++++
CustomerOperationController.java
++++++++++++++++++++++++++++++++
package in.pwskills.nitin.restcontroller;

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.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 CustomerOperationController {

      @GetMapping("/report")
      public ResponseEntity<String> showCustomerReport() {
            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);
      }

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

application.properties
server.port = 9999

request(from postman tool)
a. get   ---> http://localhost:9999/customer/report   :: FROM GET-ShowReport Method
b. post  ---> http://localhost:9999/customer/register :: FROM POST-RegisterCustomer
Method
c. put   ---> http://localhost:9999/customer/modify   :: FROM Put-UpdateCustomer
Method
d. delete --> http://localhost:9999/customer/delete   :: FROM Delete-deleteCustomer
Method


How to represent java object as JSON?
 {
      "cno" : 10,
      "cname" : "sachin",
      "caddress" : "MI",
      "billAmount" : 45000
 }

public class Customer{
      private Integer cno;
      private String cname;
      private String caddress;
      private Integer billAmount;
}
```

Sending the response in the form of JSON
++

```
package in.pwskills.nitin.entity;

public class Customer {
      private Integer cid;
      private String cname;
      private String caddress;
      private Float billAmount;

      public Integer getCid() {
            System.out.println("Customer.getCid()");
            return cid;
      }

      public void setCid(Integer cid) {
            this.cid = cid;
            System.out.println("Customer.setCid()");
      }

      public String getCname() {
            System.out.println("Customer.getCname()");
            return cname;
      }

      public void setCname(String cname) {
            this.cname = cname;
            System.out.println("Customer.setCname()");
      }

      public String getCaddress() {
            System.out.println("Customer.getCaddress()");
            return caddress;
      }

      public void setCaddress(String caddress) {
            this.caddress = caddress;
            System.out.println("Customer.setCaddress()");
      }

      public Float getBillAmount() {
            System.out.println("Customer.getBillAmount()");
            return billAmount;
      }

      public void setBillAmount(Float billAmount) {
            this.billAmount = billAmount;
            System.out.println("Customer.setBillAmount()");
      }

      public Customer(Integer cid, String cname, String caddress, Float billAmount)
{
            super();
            this.cid = cid;
            this.cname = cname;
            this.caddress = caddress;
            this.billAmount = billAmount;
      }

      @Override
      public String toString() {
            return "Customer [cid=" + cid + ", cname=" + cname + ", caddress=" +
caddress + ", billAmount=" + billAmount
                        + "]";
      }

}
```

RestController
+++++++++++++

```
package in.pwskills.nitin.restcontroller;

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 in.pwskills.nitin.entity.Customer;

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

      @GetMapping("/report")
      public ResponseEntity<Customer> showData() {
            Customer body = new Customer(10, "sachin", "MI", 54500.5f);
            HttpStatus status = HttpStatus.OK;

            ResponseEntity<Customer> entity = new ResponseEntity<>(body, status);
            return entity;
      }

}
```

request (use POSTMAN tool)
 a. GET ----> http://localhost:9999/customer/report
     response

```
      {
            "cid": 10,
            "cname": "sachin",
            "caddress": "MI",
            "billAmount": 54500.5
      }
```

Note: In order to send JSON as the reponse, SpringRest internally uses JacksonAPI
      While Sending the response, JacksonAPI calls getXXXX() on the fields of POJO

class and it creates a JSON Data.

eg#2. Sending complext json response from the API

```
package in.pwskills.nitin.restcontroller;

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

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 in.pwskills.nitin.entity.Address;
import in.pwskills.nitin.entity.Customer;

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

      @GetMapping("/showReport")
      public ResponseEntity<Customer> showReport() {

            Customer body = new Customer(
                              10, "sachin", 2345.5f,
                              new String[] { "blue", "black", "purple" },
                              List.of("10", "10+2", "B.E"),
                              Set.of(5454545L, 9998887776L, 23456728L),
                              Map.of("adhar", 234567, "pan", 23456),
                              new Address("INDIA", "Maharashtra", "Bombay")
                        );

            HttpStatus status = HttpStatus.OK;

            ResponseEntity<Customer> entity = new ResponseEntity<>(body, status);
            return entity;
      }
}
```

```
request
  GET --> http://localhost:9999/customer/showReport
```

```
response
{
    "cid": 10,
    "cname": "sachin",
    "billAmount": 2345.5,
    "favColours": [
        "blue",
        "black",
        "purple"
    ],
    "studies": [
        "10",
        "10+2",
        "B.E"
    ],
    "phoneNumber": [
        23456728,
        5454545,
        9998887776
    ],
    "idDetails": {
        "adhar": 234567,
        "pan": 23456
    },
    "address": {
        "country": "INDIA",
        "state": "Maharashtra",
        "city": "Bombay"
    }
}
```

++++++++++++++++++++++++++++++++++++
Working with @RequestBody Annotation
++++++++++++++++++++++++++++++++++++

```
package in.pwskills.nitin.restcontroller;

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 in.pwskills.nitin.entity.Customer;

@RestController
@RequestMapping("/customer")
public class CustomerOperationController {
      @PostMapping("/register")
      public String registerCustomer(@RequestBody Customer customer) {
            return customer.toString();
      }
}
```

request(use postman tool)
 a. post --> http://localhost:9999/customer/register
      click on body section, click on raw radiobutton, select text as "JSON", enter the json data

```
            {
                  "cid":7,
                  "cname":"dhoni",
                  "billAmount":4567.5,
                  "caddr":"CSK"
            }
```

response

```
  Customer [cid=7, cname=dhoni, billAmount=4567.5, caddr=CSK]
```

Note:
  Whenever we send the json data in the request body automatically SpringRest uses Jackson api and it binds the json data to POJO using
      1. Creates the object using Zero param constructor(fill the default values to instance variables)
      2. Inject the values from JSON to instance variables by calling setter methods.

SendingJSON with Collection Type
++++++++++++++++++++++++++++++++

```
package in.pwskills.nitin.restcontroller;

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 in.pwskills.nitin.entity.Customer;

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

      @PostMapping("/register")
      public ResponseEntity<String> registerCustomer(@RequestBody Customer
customer) {
            return new ResponseEntity<String>(customer.toString(), HttpStatus.OK);
      }
}
```

```
application.properties
server.port = 9999
server.servlet.context-path=/JsonToJavaObject
```

```
request : POST -> http://localhost:9999/JsonToJavaObject/customer/register
{
    "cid":7,
    "cname":"dhoni",
    "compDetails":[
        {"name": "iNeuron","location":"BGLR","size":250},
        {"name": "pwskills","location":"Noida","size":350},
        {"name": "IGATE","location":"HYD","size":300}
    ],
    "dob":"1991-01-03",
    "purchaseDate":"2022-06-05 19:01:23",
    "familyDetails":[
        {"adharNo": 123456,"pan":123456},
        {"adharNo": 234234,"pan":4443335}
    ]
}
```

```
response
 Customer [
 cid=7, cname=dhoni,
 compDetails=[
      Company [name=iNeuron, location=BGLR, size=250],
      Company [name=pwskills, location=Noida, size=350],
      Company [name=IGATE, location=HYD, size=300]
 ],
 dob=1991-01-03,
 purchaseDate=2022-06-05T19:01:23,
 familyDetails=[
      {adharNo=123456, pan=123456},
      {adharNo=234234, pan=4443335}
```

```
          ]
]
```