

SpringRest

=====

### JSON :: JavaScript Object Notation

=> It is a way of representing the Object using key:value format  
=> Key must be represented in "" and value can be anything, if the value is in String format then also it should be in "".  
=> One {} bracket indicates one Object or sub Object  
=> [] represents array/list/set elements values  
=> In Json Array/list/set collection will be treated as array only.  
=> In Json Array/list/collection are called "1-D" array.  
=> In Json Map collection is called "2-D" array.  
=> In Json Map collection elements and HAS-A property elements will be represented using sub/object node  
{"Key": "value", "key": "value", .....}

Java

=====

```
Customer customer = new Customer(10, "sachin", "IND", 53.4f);
```

JSON(light weight)

=====

```
{
    "cno": 10,
    "cname" : "sachin",
    "country": "IND",
    "Avg": 53.4
}
```

XML(Heavy weight)

=====

```
<customer>
    <cno>10</cno>
    <cname>sachin</cname>
    <country>IND</country>
    <avg>53.4</avg>
</customer>
```

What is the difference b/w HTML and JSON?

HTML/CSS => It is given to display data on browser by applying styles

JSON/XML => It is given to describe data(To construct data having structures)

To convert the data from Json/Java or from Java/JSON we take the help of api called "Jackson" api.

#### 1.jackson-databind

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.14.2</version>
</dependency>
```

sample-lite.json

=====

```
{
    "id": 10,
    "firstName": "Sachin",
```

```

        "lastName": "Tendulkar",
        "active": true
    }
}
public class TestApp {
    public static void main(String[] args) {
        try {
            // Create Object Mapper
            ObjectMapper mapper = new ObjectMapper();

            // using that Object read the data from json file and convert
into pojo object
            Customer customer = mapper.readValue(new File("data/sample-
lite.json"), Customer.class);
            System.out.println(customer);

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

**Note::**

To convert JSON to POJO Object internally jackson-api calls "setXXXX()" and binds the value to java Properties.

```

{
    "id" : 10  ===setXXXX()=====> setId(10)
}

```

To Convert Pojo to JSON, internally jackson-api calls "getXXXX()" and binds to Json Object(K,V)

```

Customer customer = new Customer("sachin");

```

```

    |
    |
customer.getXXXX()
    |
    |
{
    "name":"sachin"
}

```

ComplexJson Object[ Array, HAS-A property]

```

=====
{
    "id": 10,
    "firstName": "Sachin",
    "lastName": "Tendulkar",
    "active": true,
    "address": {
        "street": "25/1",
        "city": "Mumbai",
        "state": "Maharashtra",
        "zip": "560026",
        "country": "IND"
    },
    "languages" : ["Java", "C#", "Python", "Javascript"],
    "Company" : "MI"
}

```

```

@Data
@JsonIgnoreProperties(ignoreUnknown = true) //Any unknown properties coming into
java object for binding to ignore we use the annotation
public class Customer {
    private Integer id;
    private String firstName;
    private String lastName;
    private Boolean active;

    // HAS-A property
    private Address address;

    // Array property
    private String[] languages;
}

```

### Sending the response from the application as JSON

```

=====
@Data
public class Product {
    private Integer pid;
    private String pname;
    private Double price;
    private String[] types;
}

TestApp.java
=====
public class TestApp {
    public static void main(String[] args) {
        try {
            // Create Object Mapper
            ObjectMapper mapper = new ObjectMapper();

            Product product = new Product();
            product = getObjectData(product);
            System.out.println("In java format :: "+product);

            //Converting java object into jsonString
            String jsonData = mapper.writeValueAsString(product);
            System.out.println("In json format :: "+product);

            // Writing the data to json file
            mapper.writeValue(new File("product-list.json"), product);
            System.out.println("Wrote the data to json file");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static Product getObjectData(Product product) {
        product.setPid(10);
        product.setPname("fossil");
        product.setPrice(24556.0);
        product.setTypes(new String[]
{"chronography", "simplified", "automatic"});
    }
}

```

```

        return product;
    }
}
In java format :: Product(pid=10, pname=fossil, price=24556.0, types=[chronography,
simplified, automatic])
In json format :: {"pid":10,"pname":"fossil","price":24556.0,"types":
["chronography","simplified","automatic"]}
wrote the data to json file

```

product-list.json

```

=====
{"pid":10,"pname":"fossil","price":24556.0,"types":
["chronography","simplified","automatic"]}

```

Note::

In realtime coding/projectcoding conversion of json-pojo and pojo-json would be a boiler plate code,so to avoid this boiler plate code we need to use "Spring-Rest" module which takes care of all these internal conversions.

Syntax::

```

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

    @GetMapping("/id/{theId}")
    public ResponseEntity<Customer> getCustomerById(@PathVariable("theId")
Integer id){
        ;;;;;;
        return new
ResponseEntity<Customer>(customer,HttpStatus.OK);
    }

    @PostMapping("/save")
    public ResponseEntity<String> saveCustomer(@RequestBody Customer
customer){
        ;;;;;;
        return new ResponseEntity<String>(body,HttpStatus.OK);
    }
}

```

Working with RestApi to send Json as the response

=====

```

@Data
public class Customer {
    private Integer cno;
    private String cname;
    private Float billAmount;
    private String[] teamNames;
    private List<String> studies;
    private Set<Long> phoneNumbers;
    public Map<String,Object> idDetails;

    public Company company;
}

```

```

@Data
@AllArgsConstructor

```

```

@NoArgsConstructor
public class Company {

    private String cname;
    private String ctype;
    private String cddress;
    private Integer size;

}

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

    @GetMapping("/report/{id}")
    public ResponseEntity<Customer> showAllCustomer(@PathVariable Integer id) {

        // get from database
        System.out.println("Customer data for the id :: " + id);

        Customer customer = new Customer();
        customer.setCno(id);
        customer.setCname("sachin");
        customer.setBillAmount(54.5f);
        customer.setTeamNames(new String[] { "IND", "MI", "AsiaXI",
"Mumbai" });
        customer.setStudies(List.of("10th", "12th", "Engineering"));
        customer.setPhoneNumbers(Set.of(9994445556L, 994349845L, 98765678L));
        customer.setIdDetails(Map.of("adhar", 99453123432L, "panNo",
"DOOPQRCL12"));
        customer.setCompany(new Company("MI", "IPL", "Mumbai", 45));

        ResponseEntity<Customer> entity = new
ResponseEntity<Customer>(customer, HttpStatus.OK);

        return entity;
    }
}

```

Output

=====

```

{
    "cno": 10,
    "cname": "sachin",
    "billAmount": 54.5,
    "teamNames": [
        "IND",
        "MI",
        "AsiaXI",
        "Mumbai"
    ],
    "studies": [
        "10th",
        "12th",
        "Engineering"
    ],
    "phoneNumbers": [
        98765678,

```

```

        994349845,
        9994445556
    ],
    "idDetails": {
        "panNo": "D00PQRCL12",
        "adhar": 99453123432
    },
    "company": {
        "cname": "MI",
        "ctype": "IPL",
        "cddress": "Mumbai",
        "size": 45
    }
}

@PostMapping(value= "/save")
public ResponseEntity<String> saveCustomer(@RequestBody Customer customer) {
    System.out.println(customer);
    Integer id = 10;
    String body = "customer registered with the id :: " + id;
    return new ResponseEntity<String>(body, HttpStatus.OK);
}

```