Sending JSON Data as Http Response body to Client from service Provider App (Rest API)

==========

==========

=> if the method of Service Provider returns ResponseEntity object having

other than <String> generic (can be any other object) then given object will be converted

to JSON data and will be placed in HttpResonse as body automatically..

JSON:: Java Script Object Notation

=>It is a way of representing Object data using key: value format

note :: other than String generic means it could be

=> Key must be in "" and value can be any thing.. if the value is "string" content then it also should be in ""

any class obj or array or collection or collection of objects

=> one {} (flower bracket) represents one object or sub object (In json terminology {} represents document or sub document)

=> "[]" represents array/list/set element values

=> In JSON, array/list/set collection will be treated as array only,so their

elements will be represented using [---]

=> In JSON, array/list/set collection are called 1D arrays

=> In JSON, map collection is called 2D array

=> In JSON, Map collection elements and HAS-A property elements

will be represented using sub object/nodes {"key":value, "key":value,....}

Customer cust=new Customer(1001,"raja","hyd",67877);

In Json::

cust (Customer obj)

cno:1001

cname:λaja

cadd:hyd

billAm67877

{

"cno": 1001,

}

"cname":"raja",

"cadd" : "hyd",

"billAmt": 67877

the

=>HTML (Hyper Text Markup Language) is given to display data on browser by applying styles => JSON/XML are given to describe data (To construct data having structure)

What is the difference b/w JSON and XML?

Student st=new Student(101, "jani",67.88);

st(Student obj)

sno:101

sname: jani

Both are tag based, so these are called markup lanaguges.

YML, MongoDB docs (BSON docs) are inspired from JSON

avg: 67.88

JSON way of representing data

Xml way of representing Data

==========

{

"sno":101,

"sname":"jani",

"avg":67.88

}

(simple structure)

JSON

==========

(Both are Global

format for representing Data)

(a) It is Java Script Object Notation

(b) It is the way of represnting object data using key, value pairs

XML

//schema or DTD import (optional)

<student>

<sno>101</sno>

<sname>jani</sname>

<avg> 67.88 </avg> </student>

==========

(complex structure)

(a) It is extensible Markup language

of

(b) it is way constructing/descring data using tags

(c) maintains the data as "key": "value" pairs (c) maintains the data using tags {"key":"value", "key" : "vlaue" }

(d) Does not Provide namespaces

to validate the data

(e) This format is given by Java Script laguage (Sun Ms + Netscape)

(f) JSON files are easy to read and process

(g) To handle JSON Data we use the support of JACSON api and others...

(h) To covert JSON data to Object

and vice-versa we can use GSON, JSON-B and etc.. api

of

(i) The Process converting object

to JSON Data is called serialization and reverse is called Deserilziation

(j) JSON fomat/ files are ligh weight compare to XML format/files

(k) Very much used is Restfull web serices as alternate to XML

to send and recive data

(l) JSON is less structured compare to Xml

as the tags body and attributes [<> </>] namespaces

to validate data (namespace is a library that contains

(d) provides

(e)

the

set tags, attributes and rules to construct xml data) This is given according to w3c recomadations and inspired from SGML

(Standard Generalized Markup Language)

(f) Xml files are bit complex to read and process.

(g) To handle Xml data we take support of

xml apis like Jaxp (java api for xml processing) Jaxp deals with SAX api, DOM api, JDOM api and etc..

(i) To convert Xml doc to objects and vice-versa we use JaxB api (Java api for Xml Binding)

(i) The Process of converting object to xml data

is called marshalling and reverse is called unMarshalling

j) XML fomat/ files are bit heavy weight compare to JSON format/ files

(k) Very much used in SOAP based web services

to send and recieve data

(SOAP message are xml messages)

(1) Xml is more structured..

XSD,DTD based

namespaces are avaialable

SAX:: Simple API for XML processing DOM :: Document Object Model JDOM :: Java Document Object Model

To read and process we use Javax-p apis..where as for Marshalling and unmarshallin activities we use Jax-B apis

JSON Serialization is no way related to

FILE IO Serialization concept

(m) does not support commenting

(n) Less secured becoz it just contains keysand values

(0) Allows only UTF-8 Characters

(m) supports commenting

(n) More secured becoz data is having

hierachy strcuture and namespaces protected

(0) Allows lots of Charsets including UTF-8

UTF :: Unicode Transformation Format

Provider

Making RestController methods (service methods) sending Java Object data as JSON data

==========

==========

Service API methods =Rest Operations = Rest methods = API operations EndPoints =Rest End Points

```
//RestController
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.RestController;
import com.nt.model.Customer;
@RestController
public class CustomerOperationsController {
@GetMapping("/report")
public ResponseEntity<Customer> showData(){
Customer cust=new Customer(1001,"raja",54566.66f); HttpStatus status=HttpStatus.OK;
return new ResponseEntity<Customer>(cust,status);
}
}
//model class
==========
package com.nt.model;
import lombok.AllArgsConstructor; import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Customer {
private Integer cno;
```

private String cname;

private Float billAmt;

}

from POSTMAN

==========

GET

http://localhost:3030/SpringBootRestProj03-SendingJSONDataAsResponse/report (SEND)

response body

==========

{

}

"cno": 1001,

"cname": "raja",

`"billAmt": 54566.66`

Sending Complext obj data as the complex JSON Data to Client from RestController

// In RestControler class

==========

Model class

==========

@GetMapping("/report1")

public ResponseEntity<Customer> showData1(){

//body

Customer cust=new Customer(1001,"raja",54566.66f,

new String[] {"read","green","blue"},

List.of("10th","10+2","B.Tech"), Set.of(544535345L,7576575L,6465654L),

Map.of("aadhar", 35345435L, "panNo", 354353534L), new Company("SAMSUNG","hyd","Eletronics",4000));

//status

HttpStatus status=HttpStatus.OK;

return new ResponseEntity<Customer>(cust,status);

}

POSTMAN Tool

==========

GET http://localhost:3030/SpringBoot RestProj03-SendingJSONDataAs Response/report1 Send

response body

Customer.java

=-==========

package com.nt.model;

```
import java.util.List;
import java.util.Map;
import java.util.Set;
import lombok.AllArgsConstructor; import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor @AllArgsConstructor public class Customer {
}
private Integer cno;
private String cname;
private Float billAmt;
private String[] favColors;
private List<String> studies;
private Set<Long> phoeNumbers;
private Map<String, Object> idDetails; //HAS-A property
private Company company;
Customer object (cust)
package com.nt.model;
import lombok.AllArgsConstructor; import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor @AllArgsConstructor public class Company {
private String name;
private String addrs;
private String type;
private Integer size;
}
//Model class
package com.nt.model;
import java.util.List;
import java.util.Map;
import java.util.Set;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
"cno": 1001,
"cname": "raja",
```

"billAmt": 54566.66,

"favColors":[

"read",

"green", "blue"

1D Array

],

"studies": [

"10th",

"10+2",

1D Array

"B.Tech"

],

"phoeNumbers":

7576575, 544535345, AD Array

6465654

],

2D Array

"idDetails": {

"aadhar": 35345435,

"panNo": 354353534

},

"company":{

"name": "SAMSUNG",

"addrs": "hyd",

"type": "Eletronics", "size": 4000

2D array

```
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
}
@Data
```

Returning array of JSON Documents

=========

==========

```
//RestController class package com.nt.rest;
import java.util.List;
import java.util.Map;
@NoArgsConstructor
```

@AllArgsConstructor

@RequiredArgsConstructor

public class Customer {

@NonNull private Integer cno;

@NonNull

private String cname; @NonNull

private String cadd; @NonNull

private Float billAmt;

private String[] favColors;

private List<String> friends;

private Set<Long> phones;

private Map<String, Object> idDetails;

private Company company;

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 com.nt.model.Company;

import com.nt.model.Customer;

GET

@RestController

@RequestMapping("/customer-api")

public class CustomerOperationsController {

@GetMapping("/report2")

public ResponseEntity<List<Customer>> showReport2(){

Customer cust1=new Customer(1001,"raja","hyd", 90.0f); Customer cust2=new Customer(1002,"mahesh", "vizag", 90.0f); Customer cust3=new Customer(1003,"ramana","blore", 90.0f);

} }

cno:1001

cname:raja

billamt: 54566.66

favColors: [blue, red, geen]

studies: [10th,10+2, b Tech] phoneNumbers: [.... ....., ...] idDetails: [aahar: 454535, ... //HAS property

company:

return new ResponseEntity<List<Customer>>(List.of(cust1, cust2,cust3), HttpStatus.OK);

http://localhost:3131/Boot Rest Proj03-RestAPI-ProviderApp-JSONData/customer-api/report2

[

{

"cno": 1001,

"cname": "raja",

"cadd": "hyd",

"billAmt": 90.0, "favColors": null,

"friends": null,

"phones": null,

"idDetails": null,

built-in Jackson api jar files

Spring web starter gives >spring-boot-starter-json-3.4.1.jar - C:\Users\Natara > jackson-databind-2.18.2.jar - C:\Users\Nataraz\.m2 >jackson-annotations-2.18.2.jar - C:\Users\Nataraz\.

> jackson-core-2.18.2.jar - C:\Users\Nataraz\.m2\rep

jackson-datatype-jdk8-2.18.2.jar - C:\Users\Nataraz

"company": null

},

{

"cno": 1002,

"cname": "mahesh",

> >

jackson-datatype-jsr310-2.18.2.jar - C:\Users\Natar

jackson-module-parameter-names-2.18.2.jar - C:\U

Send

"cadd": "vizag",

"billAmt": 90.0,

"favColors": null,

"friends": null,

"phones": null, "idDetails": null, "company": null

},

{

"cno": 1003,

"cname": "ramana",

"cadd":"blore",

"billAmt": 90.0,

"favColors": null,

"friends": null,

"phones": null,

"idDetails":

null,

"company": null

}

]

What is the need of taking ResponseEntity<T> as the return of the end point, can we take directly <T> with out ResponseEntity?

if @Restcontroller class handler method/end point method return type is direct simple

type or wrapper type or String or some other class object with out using ResponseEntity object

then we will not get any control on Http Response status code and response headers.. So

it is recomanded to take ResponseEntity<T> as the return type endpoint method

=>The endpoints of @RestController class sends generated output to consumer app either as simple text or JSON content or Xml Content becoz the of @ResponseBody that is applied on the endpoint method through @RestController sub annotation =>@RequestBody and @ResponseBody are media type annotations becoz they are useful to convert input data or output content to various other formats as required like Object to JSON, Object to XML (@ReponseBody), JSON to Object, XML to Object (@RequestBody)

Company obj

name: samsung

addrs: hyd

type: Eletronice

size: 4000