

Web Services

(ReST API)

Jersey2.x and 3.x

Tomcat9

Jdk8 to jdk15

Integration of two applications:

It means connecting one application which is running in Server one with another application which is running in Server Two.

Example:

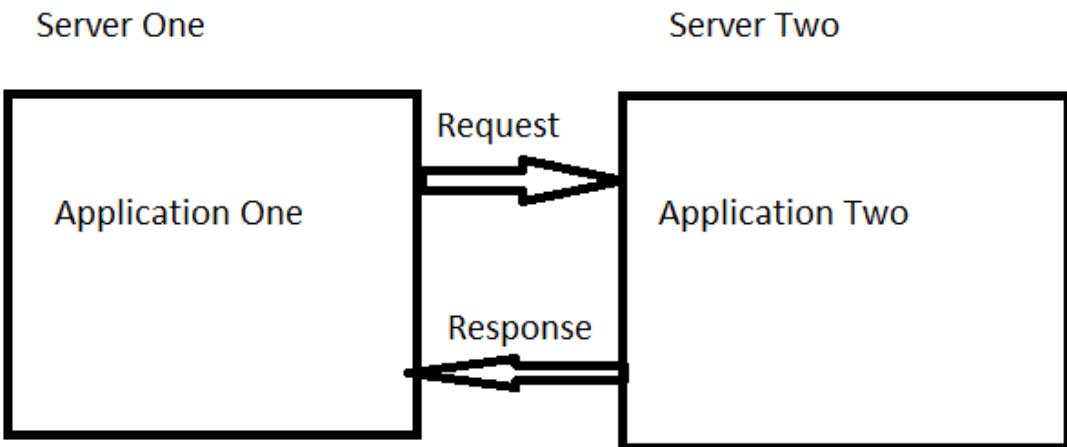
BookMyShow is connected to PayTM.

IRCTC is connected to SBI.

These two applications will communicate together to provide better response to end-user.

Here communication is done by sending request and receiving response, it looks like below.

Integration



Based on languages used to develop application, integration is divided into two types.

- 1.Homogeneous Integration
- 2.Heterogeneous Integration

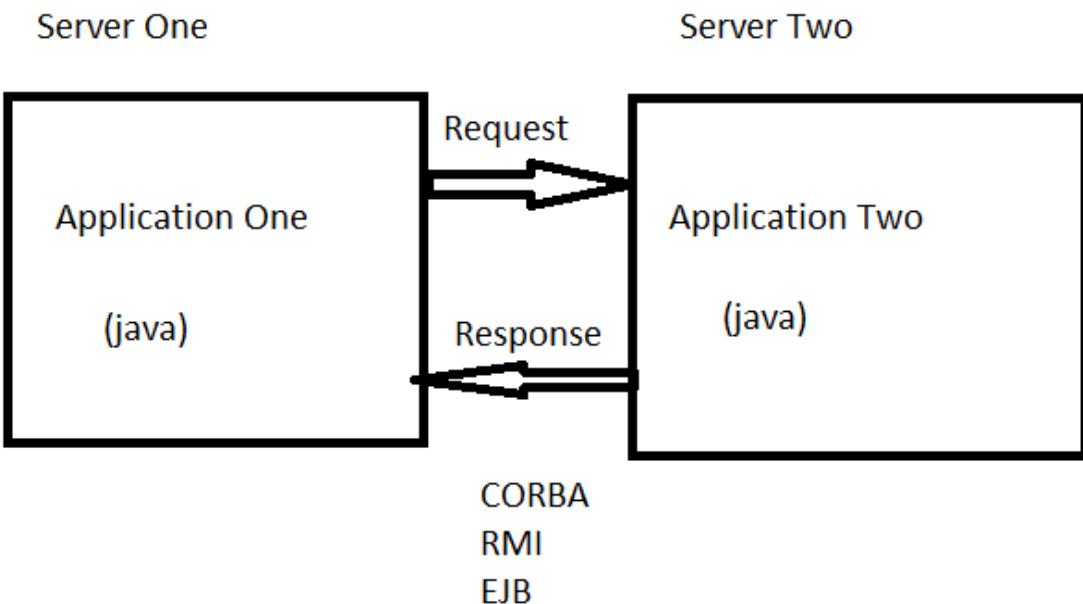
1.Homogeneous Integration:

If both applications are developed in same language and Integrated together is called Homogeneous Integration.

- In case of java, CORBA, RMI, EJB are web for Integration
- >CORBA – Common Object Request Browser Architecture.
 - >RMI - Remote Method Invocation.
 - >EJB - Enterprise Java Bean.

Example:

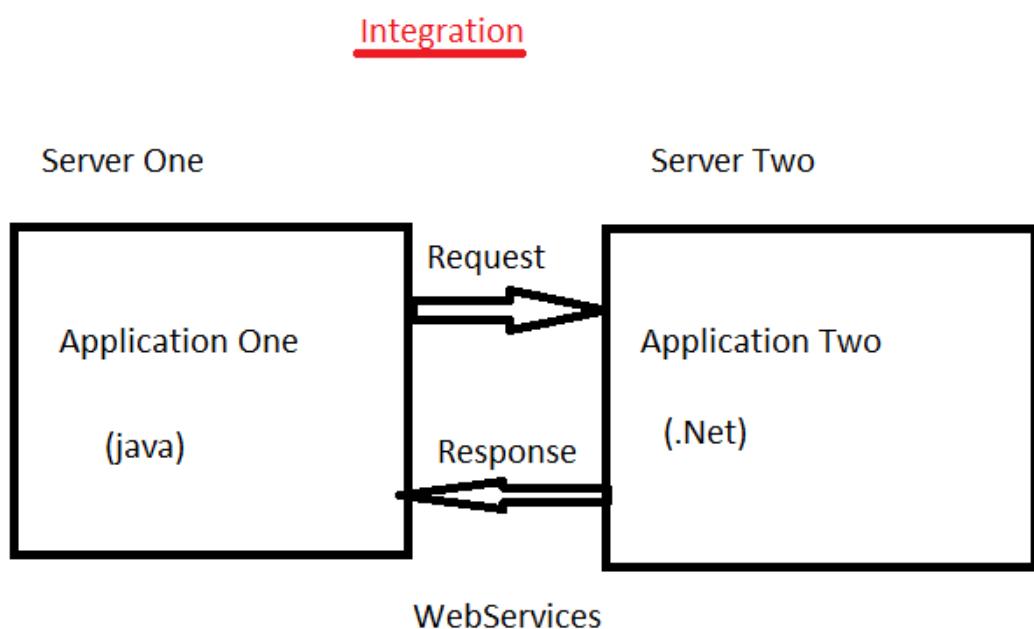
Integration



2. Heterogeneous Integration

If two applications are integrated together which are developed in different languages is called Heterogeneous Integration.

Example:

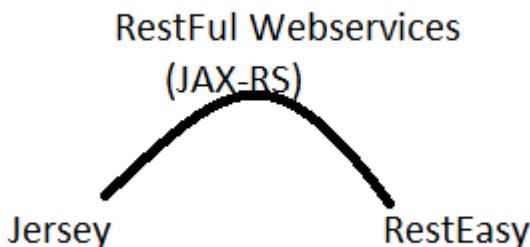


Here Webservices supports both Homogeneous and Heterogeneous Integration.

Define Web Services?

Integration of two different applications which might be running in same Server or different Server, also applications developed in same language/different languages.

To do Integration, minimum two applications are required. In that one application behaves like Producer and another application behaves like Consumer.



HTTP protocol Introduction

Web/Webservices applications uses HTTP protocol for communication between Consumer and Producer.

->HTTP- Hyper Text Transfer Protocol.

->This protocol exchanges data(Content) between two Apps (Consumer – Producer) which can be any
DataType(XML/JSON/HTML/TEXT/BINARY...etc)

->It supports two types of Messages

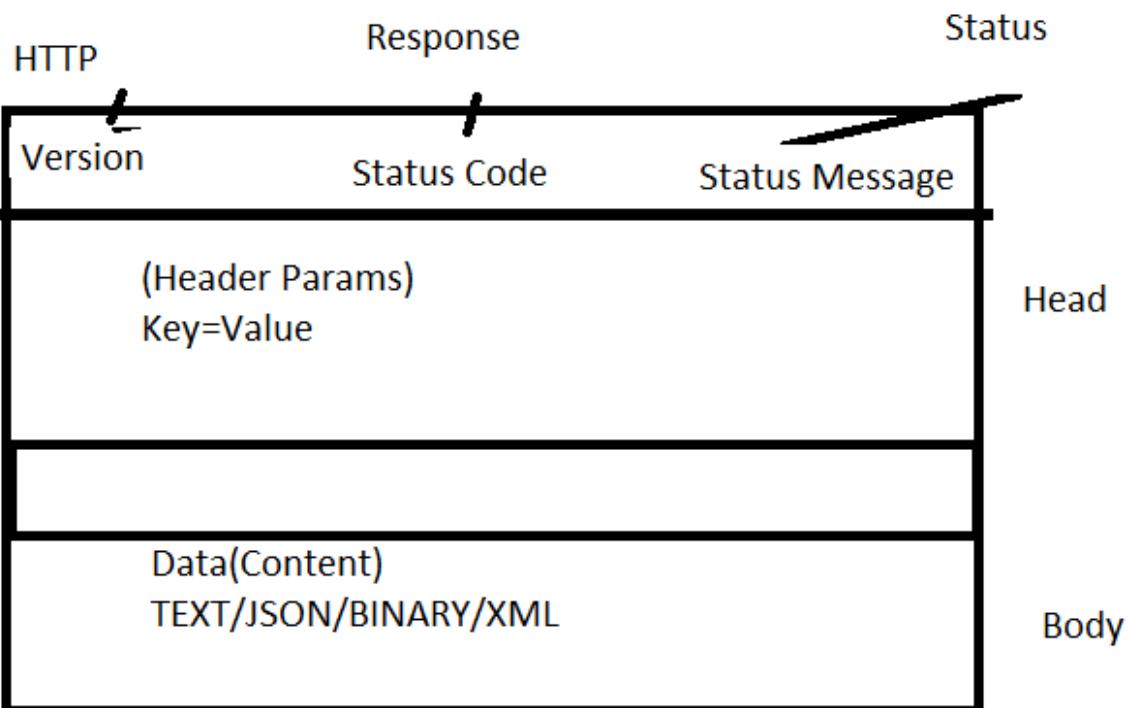
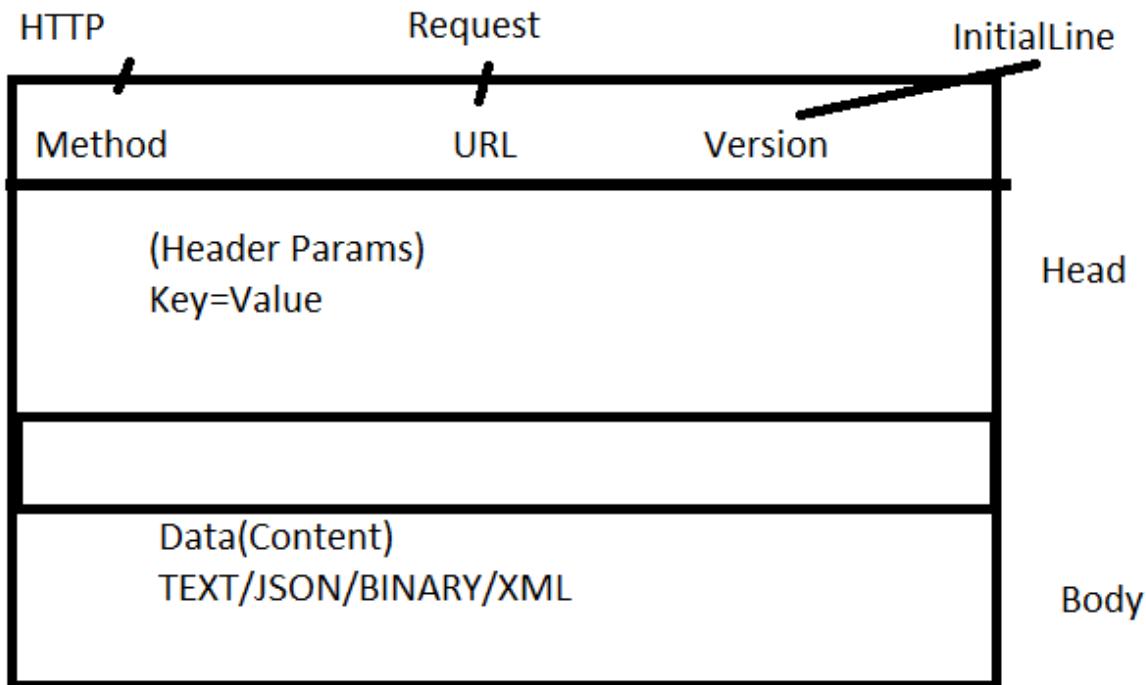
Those are:

HTTP Request or HTTP Response, contains two parts, those are Head and Body.

1.Body will Store data(Content) and Header will Stores Parameters(Key=Value).

2.Header also supports initial Line (Actual Details) which is different for Request and Response.

HTTP Formats



HTTP Request Methods:

Here, method means which indicates task that should be done by Server[Producer].

A task can be defined using below method

They are four types::

- 1.GET -Fetching Resource from Server.
- 2.POST -Creating new Resource at Server.
- 3.PUT -Modify existing Resource at Server.
- 4.DELETE -Remove existed Resource at Server.

->Here, Resource means Text data/File/Database/Documents...etc.

->Other HTTP methods types they are(5)::

- 5.PATCH
- 6.HEAD
- 7.OPTIONS
- 8.TRACE
- 9.CONNECT.

Xxx HTTP Response Status::

Status(Code-Message) indicates “what is final result given by Server/Producer”.

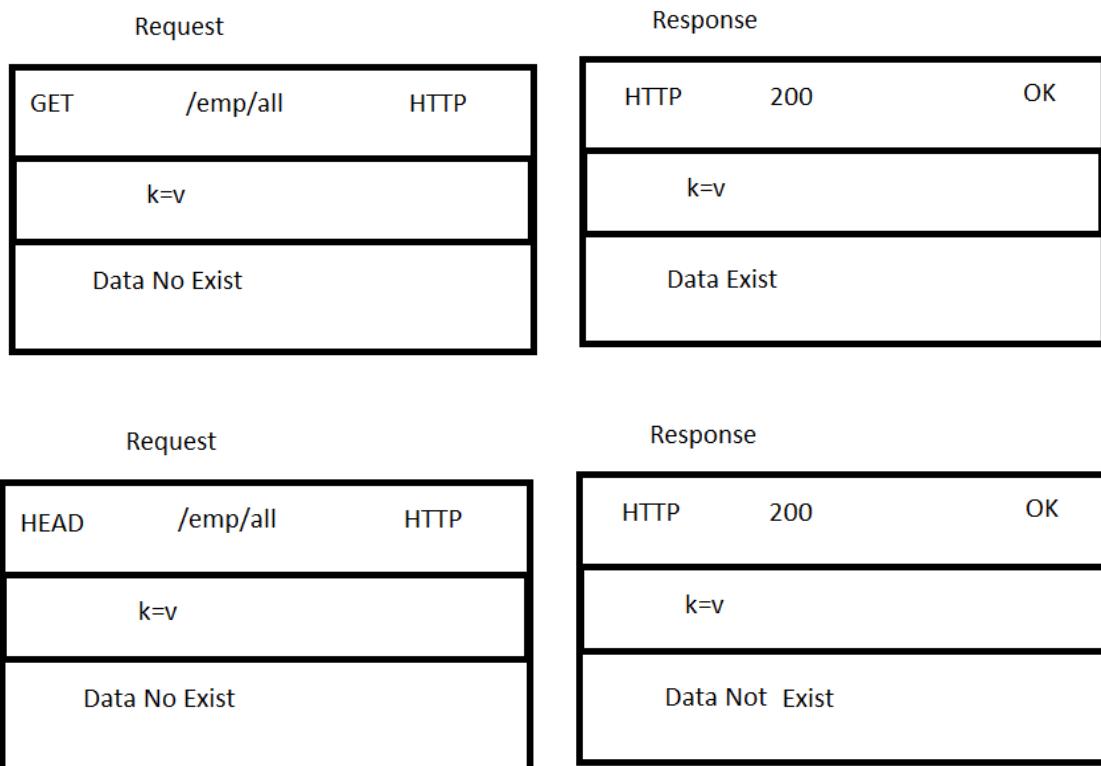
->HTTP supports 5 types of status Codes

| Code | Message |
|------|--------------------|
| 1xx | Information |
| 2xx | Success |
| 3xx | ReDirect |
| 4xx | Client Side Error |
| 5xx | Server Side Error. |

Q) What is difference between GET and HEAD?

A) GET: This method type is used to fetch data from server(Producer to Consumer).But it can never Support HTTP Request Body. It contains Response Body which gets data from Producer to Consumer.

HEAD: This method is used to “ Give a task to Producer” It never Supports Request Body and even no Response Body, Only HEAD section for both Request and Response.



Q) What is the difference between PUT and PATCH?

PUT: This HTTP method Indicates data modification is done at High level or almost data is modified or Complete Data is updated.

PATCH: This HTTP method is used to indicate Slight or Partial Data modifications.

->PUT even can be used for Partial data modification but, PATCH can never be used for Full or High-level data modification.

JSON

A Story Of JSON::

Minnie – Do you know about JSON

Mickey – Yes, of course, do you want to hear a story

Minnie (excited) – Yes



Mickey – Let's go to the riverside restaurant then

Minnie – Awesome, let's go



(At the restaurant)

Minnie – Let's Start!

Mickey – Okay! Let's say there are different countries and each has its own local language for communication

Minnie – Okay

Mickey – Now the people within the same country can communicate easily using their own language. But it will be difficult to interact with people from other countries

Minnie – That's True!

Mickey – So what can be a solution here

Minnie – Well! they can keep an interpreter who understands the language of both countries and can help people to communicate



Mickey -Yes, but do you think that will be practical and manageable

Minnie – hmmm...

Mickey – Imagine, every person or every group will need to go with an interpreter whenever they need to communicate. And then one interpreter may understand 2 or a few languages but not all. So again there will be overhead to keep changing interpreters for communicating with different groups



Minnie – Yes, that sounds scary. I did not think of that

Mickey – Can there be a better and simpler option

Minnie (thinking) – hmmm..

Mickey – What if they all can agree and create a simple common language (like English) that all can understand and use to communicate with each other



Minnie – That would be good, but all of them will have to learn the new language

Mickey – Yes, I will talk more about that later. For now, just imagine that it will be a simple structural language that anyone can learn easily

Minnie – Yes that will be great

Mickey – So now take the same example of programming languages We need a way so that different computer languages or applications built over different languages and platforms can communicate with each other

Minnie – So is **JSON** the common language here

Mickey – Yes,
JSON stands for JavaScript Object Notation

J Java

S Script

O Object

N Notation

Minnie – I get that

Mickey – JSON is an open standard lightweight data-interchange format that uses human-readable text to store and transmit data

Minnie – Okay

Mickey – It is basically a text format that makes it easy to share data between devices such as Clients and Servers

Minnie – Why the name “JavaScript Object Notation”. Has it to do anything with JavaScript language

Mickey – JSON uses the basic idea of javascript data structure. Its origin is based on how JavaScript objects work, so in that sense, it is related. Regardless of the fact that it has its origin from JavaScript, it is widely used across many Languages

Minnie – Alright!

Mickey – Now JSON is more like a structure or format

Minnie – What do you mean by that

Mickey – hmm... You can say it is not a complete language but the basic or grammar of a language

Minnie – Oh I see!

Mickey – So learning or understanding JSON is not like learning a completely new language, as we saw in our example of English

Minnie – So JSON is not a programming language

Mickey – Yes, it is not. It is a common and open standard format in which we can present our data or message and send over to other apps

Minnie – So is XML related to JSON

Mickey – XML is another way of communication. There can be more than one common language to communicate between people of different geographies and similarly, we have 2 formats XML and JSON to communicate between different programming languages or apps

Minnie – Okay so until now I have got this

1. **JSON** stands for **JavaScript Object Notation**
2. **JSON** is a syntax or format for storing and exchanging data
3. **JSON** is language independent
4. Platforms or applications built over different languages can exchange data or information using **JSON**

Mickey – Yes, well done!

Minnie – Okay Got it. So we now know that applications built over different languages can exchange data using JSON. But until now we have not discussed the actual format or structure of JSON

Mickey – Yes, now it's time to go deep

Minnie – Wow, this is getting interesting

Mickey – Before that do you want to get some coffee and cookies

Minnie – Yeah, I believe I will need them

(Mickey orders coffee and cookies for Minnie)



Mickey – Okay now you will need a pen and paper

Minnie – Here you are



Mickey – Okay, now give me a list of Dogs with their breed, color, age, etc. Can you give me the data here?

Minnie – Sure

So here you go

Buddy, German Shepherd, Black, 10

Glory, Beagle, Brown, 5

Tyson, Pug, White, 7

Lakmi Poodle, Grey, 2

Mickey – Okay that is good. Now suppose this will be a data of 100 Dogs. Do you think it will be easy to read, manage, update, and search in the way you have created?

Minnie – That will be a mess

Mickey – Okay so now let me structure this data in JSON

```
{  
  "Dogs": [  
    {  
      "name": "Buddy",  
      "breed": "German Shepherd",  
      "color": "Black",  
      "age": 10  
    },  
    {  
      "name": "Glory",  
      "breed": "Beagle",  
      "color": "Brown",  
      "age": 5  
    },  
    {  
      "name": "Tyson",  
      "breed": "Pug",  
      "color": "White",  
      "age": 7  
    }  
  ]  
}
```

```
        "age": 7
    },
    {
        "name": "Lakmi",
        "breed": "Poodle",
        "color": "Grey",
        "age": 2
    }
]
```

Minnie – Wow, this looks very formatted and easy to read
Is this JSON

Mickey – Yes! Exactly

Minnie – Wow

Mickey – Now, you see, it is easier to manage, update, and also to search anything both by humans and also by programming languages

Minnie – This is getting interesting. Tell me more...

Mickey – So here are some rules of JSON

- Data is in key/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays



Minnie – So in the above examples we have an array of Dogs

Mickey – Yes! Right. Let's understand more

A key/value pair consists of a field name or key (in double quotes), followed by a colon, followed by a value:

Example

`"name" : "Buddy"`

In JSON, keys must be strings, written with double quotes:

In JSON, values can be of type

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

Minnie – Yes, now I can see that in the example

Mickey – Also you must have noticed all data is separated by a comma,

Minnie – yeah! I see that

Mickey – Now another rule says, Curly braces hold objects or we can say JSON objects are surrounded by curly braces {}

Minnie – So this is one object

```
{  
  "name": "Buddy",  
  "breed": "German Shepherd",  
  "color": "Black",  
  "age": 10  
}
```

Mickey – Yes! Absolutely correct. There can be nested objects i.e.

objects within the object

And inside the objects

- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.

Minnie – Yes I can see in the data

Mickey – You can also access the object values using dot(.) in programming

Example

```
dogObject1 = { "name" : "Buddy", "breed" : "German Shepherd", "color" :  
  "Black", "age" : 10 };
```

```
x = dogObject1.name;
```

OR using square brackets []

```
x = dogObject1["name"];
```

Minnie – This is getting interesting and deep. I need more coffee

Mickey – Sure Minnie



Minnie – This is so refreshing. Let's continue now

Mickey – Sure, Now coming to Arrays, we know Square brackets hold arrays OR we can say Arrays are surrounded by Square Brackets

Minnie – Yes I see in the Dogs array

```
{ ← Object Starts
  "Dogs": [ ← Array Starts
    { ← Object Starts
      "name": "Buddy", ← Value
      "breed": "German Shepherd", ← Key
      "color": "Black",
      "age": 10
    } ← Object Ends
  ] ← Array Ends
} ← Object Ends
```

Mickey – Okay so tell me, how can I access the 2nd dog's detail from the Dogs array

Minnie – hmm don't know

Mickey – Suppose you store the complete Dogs array in a variable say `myDogs`

Now you can say

```
x = myDogs.Dogs[1]
```

Minnie – But we needed the 2nd dog's data, Why have you given 1 here

Mickey – That's because the index position starts from 0 and not 1

Minnie – Got it. So this will give us

```
{
  "name": "Glory",
  "breed": "Beagle",
  "color": "Brown",
  "age": 5
}
```

Mickey – Exactly

Minnie – I am getting this

Mickey – You can also parse or search any data inside a JSON using JSON Path

Minnie – What is [JSON Path](#)

Mickey – It is the location or address or a specific value or data in a JSON

You can create a JSON Path manually or use tools like JSON

PathFinder – <http://jsonpathfinder.com/>

Minnie – Great!

Mickey – There is also a Chrome extension by the same name – JsonPath Finder

Minnie – I see!

Mickey – Okay, let's do an activity



Minnie – Sure Mickey. I am excited!

Mickey – just copy the Dogs JSON and put it in here <http://jsonpathfinder.com/>

Then try to get JSON Path for the name of 3rd dog

Minnie – Okay

Mickey – On the right side expand the arrow to see your formatted data and then click on the value you want and JSON path will be displayed in the Path field above

JSON Path Finder

```
1 {  
2   "Dogs": [  
3     {  
4       "name": "Buddy",  
5       "breed": "German Shepherd",  
6       "color": "Black",  
7       "age": 10  
8     },  
9     {  
10       "name": "Glory",  
11       "breed": "Beagle",  
12       "color": "Brown",  
13       "age": 5  
14     },  
15     {  
16       "name": "Tyson",  
17       "breed": "Pug",  
18       "color": "White",  
19       "age": 7  
20     },  
21     {  
22       "name": "Lakmi",  
23       "breed": "Poodle",  
24       "color": "Grey",  
25       "age": 2  
26     }  
27   ]  
28 }
```

Path: x.Dogs[2].name

Dogs:

- 0:
- 1:
- 2:
 - name: Tyson
 - breed: Pug
 - color: White
 - age: 7
- 3:

Minnie – Okay

Mickey – What did you get

Minnie – This

[x.Dogs\[2\].name](#)

Mickey – yes, so this is the JSON path to the name of 3rd dog object

Minnie – Great! This is interesting

Mickey – So you will find a lot of tools online for beautifying JSON, validating JSON and generating JSON path

Minnie – Wow!

Mickey – Also remember

- The file type or file extension for JSON files is “.json”
- The Media type for JSON text is “application/json”

Minnie – What is Media type

Mickey – Media type or MIME-type is a label used to identify a type of data

It is used so the software can know how to handle the data.

So when you send data, you also send its media type so that the server will know what type of data you are sending and can parse it accordingly

Minnie – Okay got it, that is why in HTML requests and API request I see this header **content-type**

Mickey – Exactly, that is to provide the type of data like “application/json” if the data is in JSON

Minnie – Many things getting clear here

Mickey – Happy to know that

Minnie – So, that means every programming language or application will have some function to parse a JSON data

Mickey – Minnie you are asking all right questions today. Exactly, you will find a function `JSON.parse()` in all languages that can communicate with JSON format

Minnie – I am got so much information today

Mickey – Remember JSON is like One universal language for communication between programming languages and applications

Minnie – Yes I will

Mickey – Should I drop you home now

Minnie – No, let's order some more coffee and cake and enjoy the sunset

Mickey – How much coffee will you have today?

Minnie – Thanks for all this. I will never forget.

Mickey – I am always here for you

JSON: Java Script Object Notation

In Simple it is a Object in JavaScript.

-> It is a process of Object representation in JavaScript Language.

-> It takes less memory to show complete Object data compared with any other programming languages.

->So, it is called as Light Weight Object Format.

->JSON format is:

```
{  
    Key: value  
    Key: value  
    .....  
}
```

->Here one {} indicates one Object

->Key must be placed in double quotes(")

->value is placed in double quotes only if type String.

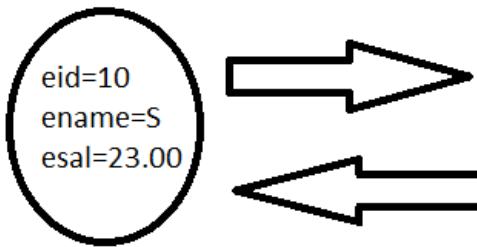
->JSON/SERIALIZATION: It is a process of Converting java Object to Json format.

->JSON/DESERIALIZATION: It is a process of Converting JSON to Java Object Format.

Example:

```
public class Emp{  
    int eid;  
    String ename;  
    double esal;  
}
```

Java Object



JSON
{
 "eid" : 10,
 "ename" : "S",
 "esal" : 23.00
}

JSON Converter API'S::

To Convert Object <-> JSON format, we should use one Converter API.

Few are given below::

- a.JACKSON Converter API**
- b.GSON(google) Converter API**

c.JSON-B(Binding)/JAXB Converter API.

->These Converters Supports two Operations

(Object to JSON and even JSON to Object).

->Any Converter we use, final Output is same. i.e Input and Output never gets changed.

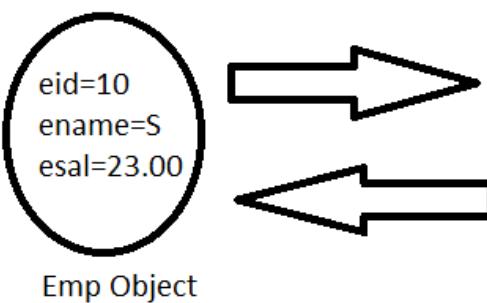
JSON Examples::

1. Working on Primitive variables (int, String, double, float, etc.....)

Example:

```
public class Emp{  
    int eid;  
    String ename;  
    double esal;  
}
```

Java Object



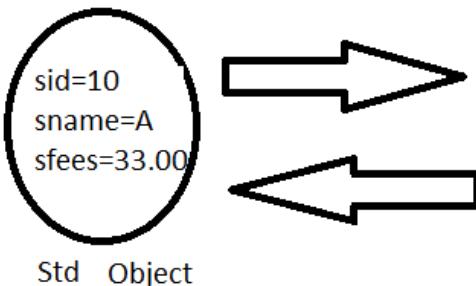
JSON

```
{  
    "eid" : 10,  
    "ename" : "S",  
    "esal" : 23.00  
}
```

Example:2

```
public class Std {  
    int sid;  
    String sname;  
    double sfees;  
}
```

Java Object



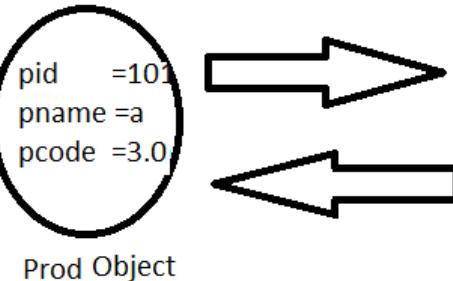
JSON

```
{  
    "sid" : 10,  
    "sname" : "A",  
    "sfees" : 33.00  
}
```

Example: 3

```
public class Prod{  
    int pid;  
    String pname;  
    double pcode;  
}
```

Java Object



JSON

```
{  
    "pid" : 101,  
    "pname" : "a",  
    "pcode" : 3.0  
}
```

2. Working on 1Dimensional Collections / Arrays.

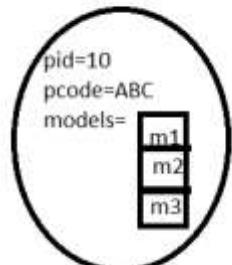
-> In case of Array/List/Set , JSON format is::

“key” : [value1,value2,value3,....]

Example:1

```
class Part{
    int pid;
    String pcode;
    List<String> models;
}
```

Java Object



JSON

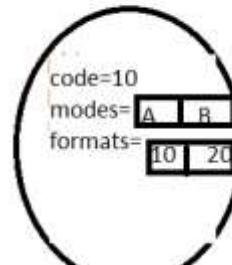
```
{
    "pid" : 10,
    "pcode" : "ABC",
    "models" : ["m1","m2","m3"]
}
```

Part Object

Example:1

```
class Grade {
    int code;
    List<String> modes;
    Set<Integer> formats;
}
```

Java Object



JSON

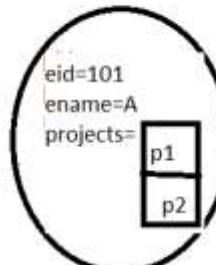
```
{
    "code" : 10,
    "modes" : ["A","B"],
    "formats" : [10,20]
}
```

Grade Object

Example: 3

```
class Emp {
    int eid;
    String ename;
    String projects[];
}
```

Java Object



JSON

```
{
    "eid" : 10,
    "ename" : "A",
    "projects" : ["p1","p2"]
}
```

Emp Object

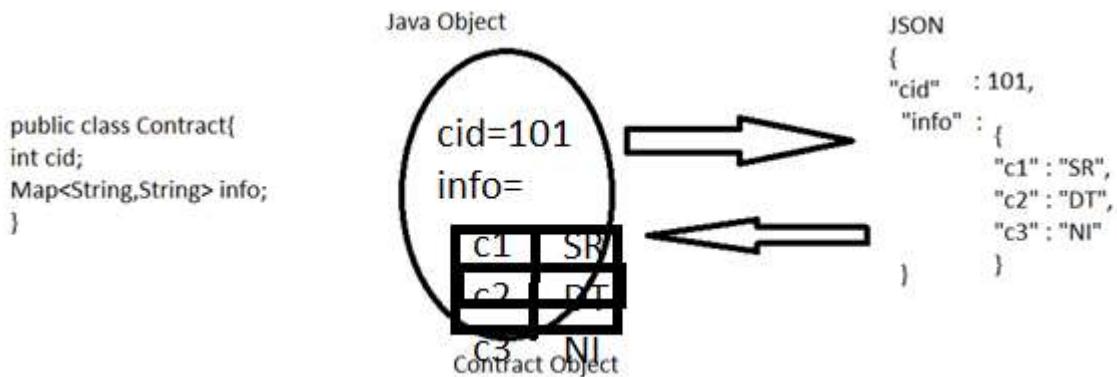
3. Working on 2Dimensional Collections::

For collection types like :: Map/Properties

JSON format is given as:

{“key”: value,.....}

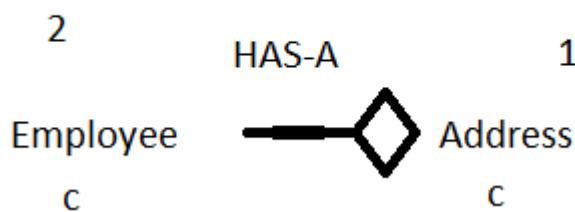
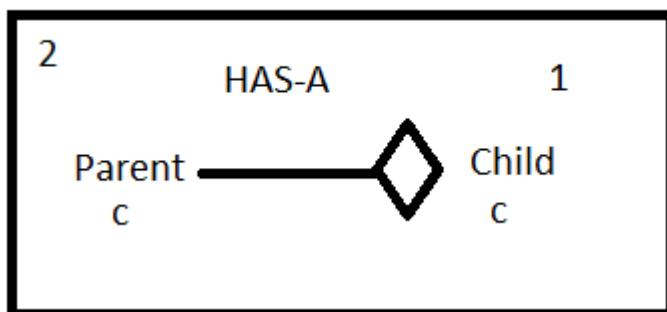
Example1::



4.Working on HAS-A Relation

HAS-A: Using one child(class /interface) as Datatype and creating one variable [reference variable]

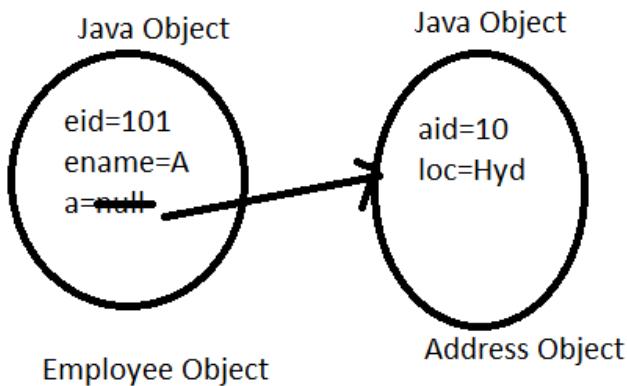
In parent (class) is called as HAS-A Relation



```
public class Employee{  
Address a;//has-a  
}
```

```
public class Address{  
}
```

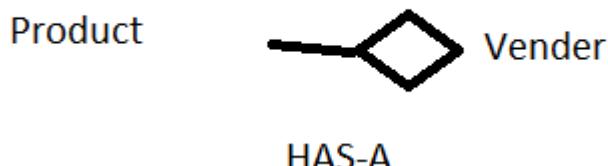
```
public class Employee{
int eid;
String ename;
Address a;//HAS-A
}
```



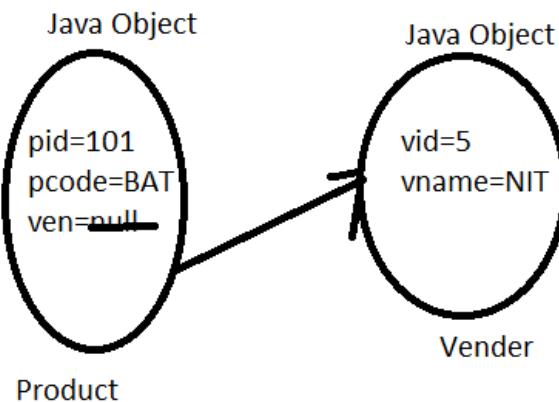
```
public class Address{
int aid;
String loc;
}
```

```
JSON
{
"eid" : 101,
"ename" : "A",
"Address": {
    "aid" : 10,
    "loc" : "Hyd"
}
}
```

Example:2



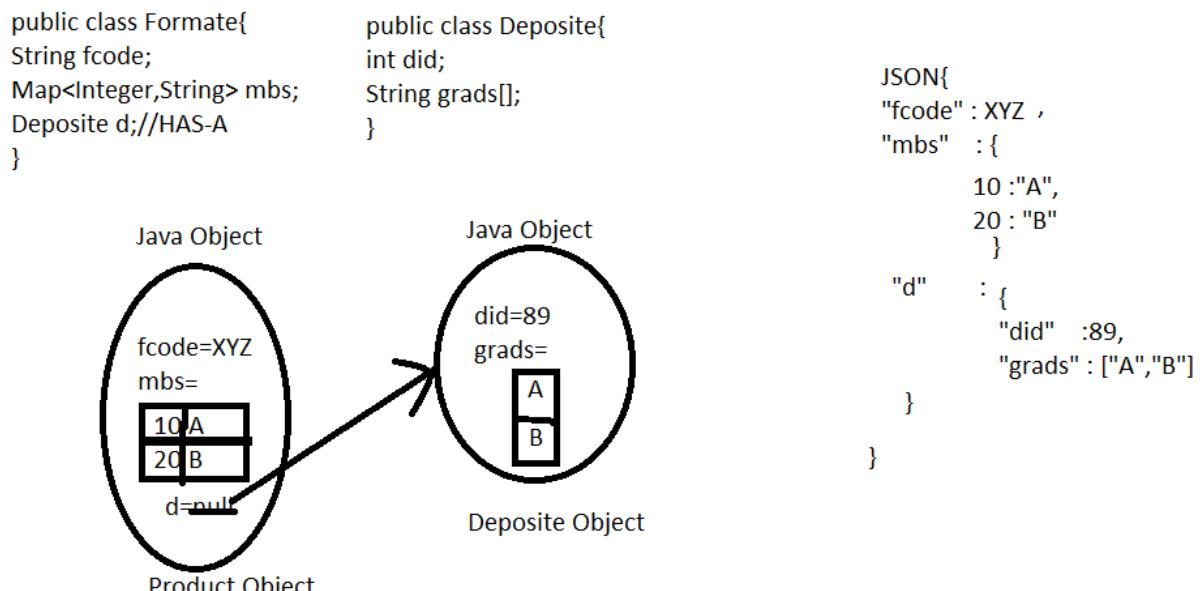
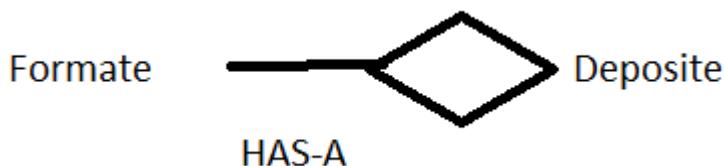
```
public class Product {
int pid;
String pcode;
Vender ven;//HAS-A
}
```



```
public class Vender{
int vid;
String vname;
}
```

```
JSON{
"pid" : 101,
"pcode" : "BAT",
"ven" : {
```

```
    "vid" : 5,
    "vname": "NIT"
}
```



JSON Converters

->Java Objects can be converted to JSON format even JSON Data can be converted to Java Objects format by using JSON converter API's.

Few JSON converter API's are::

- a. JACKSON API.
- b. GSON API.
- c. JSON-B/JAXB API.

Escape Character:

1. \n -----New Line
2. \a -----alert

3. \b -----backspace
4. \t-----tab space
5. \f-----form feed
6. \r-----carries return (means sending cursor, wherever it is, to starting position.)
7. \'-----prints`
8. \"-----prints"
9. \\-----prints\

JACKSON API

→ This is open source API given by Jersey(Company) which is used to convert Object<->JSON.

Class : ObjectMapper(c)
Methods : writeValueAsString(obj) : String
 readValue(T.class,JSON) :Object

Setup and Example::

Softwares :

JDK15, STS4.5, Eclipse, Jersey 1.19

jars.

Step1: Create Java Project

File>new>Java Project

->Enter name

Ex:JsonJacksonFirstApp

Download Link::

<https://repo1.maven.org/maven2/com/sun/jersey/jersey-archive/1.19.1/> here select zip file

->Finish

Step2: Add Jars to Project using Build Path

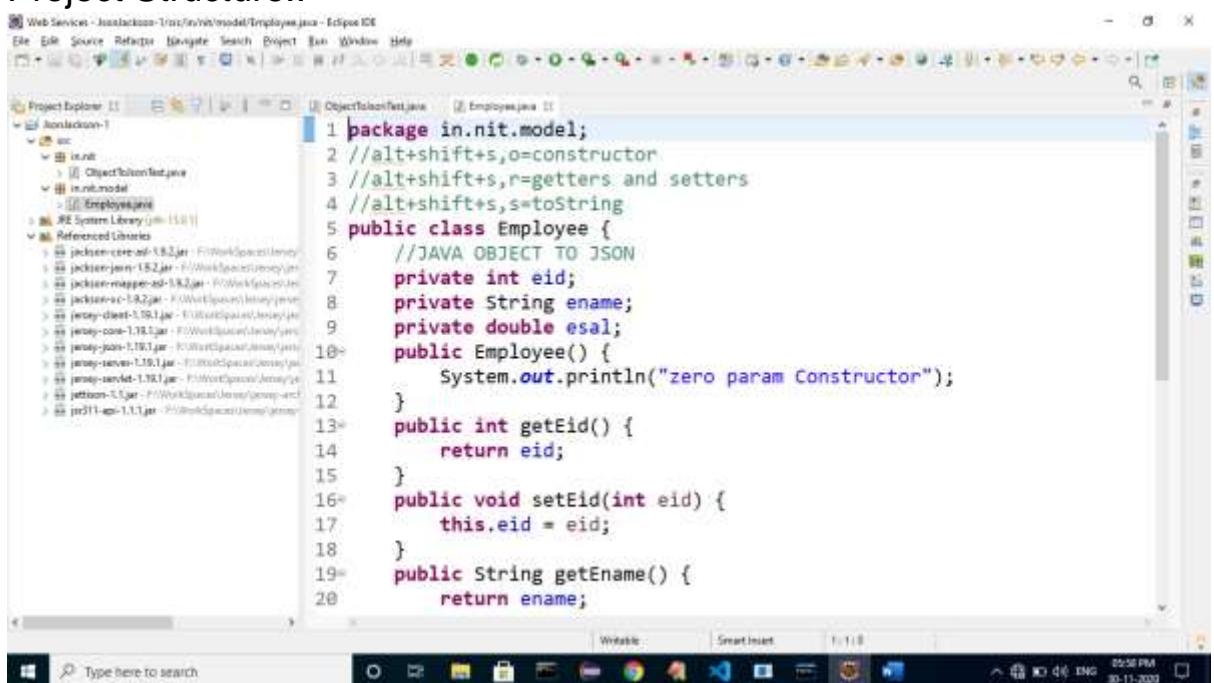
->Right click on Project>Build Path >Configure

BuildPath>click on library>Add External Jars>Choose 11 jars from Jersey jars lib Folder).

->Apply and Close.

| | | | |
|--|--------------------------|---------------------|----------|
| | jackson-core-asl-1.9.2 | 11-03-2016 02:43 PM | JAR File |
| | jackson-jaxrs-1.9.2 | 11-03-2016 02:43 PM | JAR File |
| | jackson-mapper-asl-1.9.2 | 11-03-2016 02:43 PM | JAR File |
| | jackson-xc-1.9.2 | 11-03-2016 02:43 PM | JAR File |
| | jersey-client-1.19.1 | 11-03-2016 02:43 PM | JAR File |
| | jersey-core-1.19.1 | 11-03-2016 02:43 PM | JAR File |
| | jersey-json-1.19.1 | 11-03-2016 02:43 PM | JAR File |
| | jersey-server-1.19.1 | 11-03-2016 02:43 PM | JAR File |
| | jersey-servlet-1.19.1 | 11-03-2016 02:43 PM | JAR File |
| | jettison-1.1 | 11-03-2016 02:43 PM | JAR File |
| | jsr311-api-1.1.1 | 11-03-2016 02:43 PM | JAR File |

Project Structure::



Step3: Define model class

->Right click on src->new class

->Enter Details

Package: in.nit.model

Name : Employee

--Code—

```
package in.nit.model;
//alt+shift+s,o=constructor
//alt+shift+s,r=getters and setters
```

```
//alt+shift+s, s=toString
public class Employee {
    //JAVA OBJECT TO JSON
    private int eid;
    private String ename;
    private double esal;
    public Employee() {
        System.out.println("zero param
Constructor");
    }
    public int getEid() {
        return eid;
    }
    public void setEid(int eid) {
        this.eid = eid;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public double getEsal() {
        return esal;
    }
    public void setEsal(double esal) {
        this.esal = esal;
    }
    @Override
    public String toString() {
        return "Employee [eid=" + eid + ", "
            + ename + ", esal=" + esal + "]";
    }
}
```

}

Step4:: Create One Test Class

```
package in.nit;

import org.codehaus.jackson.map.ObjectMapper;

import in.nit.model.Employee;

//ctrl+shift+o for imports
public class ObjectToJsonTest {
    //main >ctrl+space>enter
    public static void main(String[] args) {
        //try >ctrl space>enter
        try {
            //a.Create model class Object
            Employee emp=new Employee();
            emp.setEid(100);
            emp.setEname("Sankar");
            emp.setEsal(33.00);

            //b.create ObjectMapper Object
            ObjectMapper om=new
ObjectMapper();
            //c. call Write_() method
            String
json=om.writeValueAsString(emp);
            //d. Print JSON
            System.out.println(json);
            //Run Code :: ctrl+F11
        }
    }
}
```

```
} catch (Exception e) {  
}  
}  
->Right click on Test class>Run as >Java  
Application  
Output:::
```

```
{  
“eid” : 100,  
“ename”: “Sankar”,  
“esal” : 33.00  
}
```

->Maven Project creation for JsonJackson-1
Step1:

Create simple maven project
->File>new>maven project
->select checkbox
Create Simple Maven Project
Next>Enter Details
GroupId : nareshIt(company name)
ArtifactId: JsonJackson-1(Project Name)
Version : 1.0

->Finish

Step2: Add java version details
in pom.xml file

->Double click on pom.xml file>choose pom.xml
->Add below code after <version> tag
<properties>
<maven.compiler.source>15</maven.compiler.source>
<maven.compiler.target>15</maven.compiler.target>
</properties>

->press ctrl+A , ctrl+I and ctrl+S

Step3:

Also Add below dependencies in pom.xml

Here Search for dependencies:: <https://mvnrepository.com/>

<dependencies>

 <!--

<https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson> -->

 <dependency>

 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>

 <version>2.32</version>

 </dependency>

 </dependencies>

->press ctrl+A ,ctrl+I and ctrl+S

Step4:

Update Maven Project[Alt+F5]

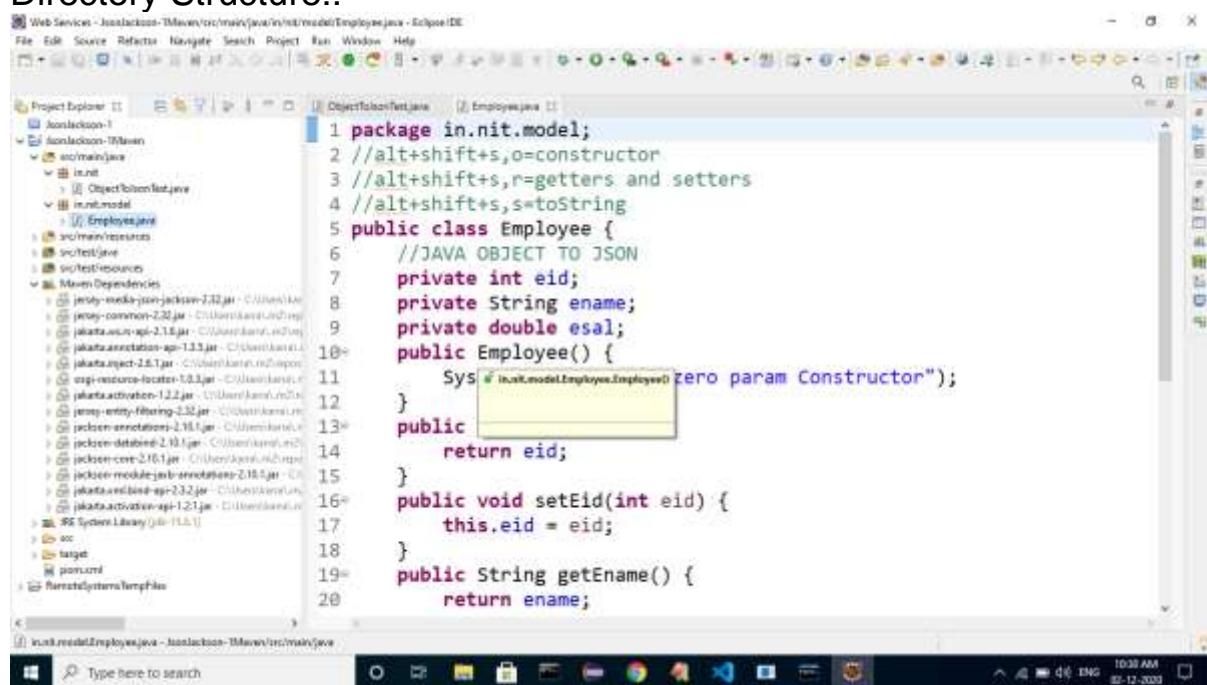
->right click on project>maven>Update Project.

->select checkbox

Force update of Snapshots and Releases

->Finish.

Directory Structure::



//Employee.java

package in.nit.model;

```
//alt+shift+s,o=constructor
//alt+shift+s,r=getters and setters
//alt+shift+s,s=toString
public class Employee {
    //JAVA OBJECT TO JSON
    private int eid;
    private String ename;
    private double esal;
    public Employee() {
        System.out.println("zero param Constructor");
    }
    public int getEid() {
        return eid;
    }
    public void setEid(int eid) {
        this.eid = eid;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public double getEsal() {
        return esal;
    }
    public void setEsal(double esal) {
        this.esal = esal;
    }
    @Override
    public String toString() {
        return "Employee [eid=" + eid + ", ename=" + ename + ",
esal=" + esal + "]";
    }
}
//ObjectToJsonTest.java

package in.nit;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;  
  
import in.nit.model.Employee;  
  
//ctrl+shift+o for imports  
public class ObjectToJsonTest {  
    //main >ctrl+space>enter  
    public static void main(String[] args) {  
        //try >ctrl space>enter  
        try {  
            //a.Create model class Object  
            Employee emp=new Employee();  
            emp.setEid(100);  
            emp.setEname("Sankar");  
            emp.setEsal(33.00);  
  
            //b.create ObjectMapper Object  
            ObjectMapper om=new ObjectMapper();  
            //c. call Write method()  
            String json=om.writeValueAsString(emp);  
            //d. Print JSON  
            System.out.println(json);  
            //Run Code :: ctrl+F11  
  
        } catch (Exception e) {  
            //  
        }  
    }  
}
```

Output::
zero param Constructor
{"eid":100,"ename":"Sankar","esal":33.0}

JSON JACKSON COLLECTIONS

->List, Set and Array are called as 1D Collections for this Data JSON format is given as::

[value1,value2,value3,...]

pom.xml::

```
<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-json-jackson</artifactId>
    <version>2.32</version>
</dependency>
</dependencies>
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.16</version>
    <scope>provided</scope>
</dependency>
```

1. Define model class (ID Collections)

```
package in.nit.model;
```

```
import java.util.List;
import java.util.Set;

import lombok.Data;
@Data
public class Student {
    private int stdId;
    private String stdName;
    private double stdFees;
    private List<Integer> marks;
    private Set<String> subjects;
    private String grades[];
```

```
}
```

2. Test class for Object -> JSON Conversions

```
//ObjectToJSONTest.class
```

```
package in.nit.test;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
```

```
import java.util.Set;

import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Student;

public class ObjectToJSONTest {

    public static void main(String[] args) {
        try {
            Student st=new Student();
            st.setStdId(101);
            st.setStdName("Sankar");
            st.setStdFees(23443.00F);

            /*
             * //jdk1.7 List<Integer> list=new ArrayList<>();
             */
            list.add(89);list.add(87);list.add(82);list.add(85);list.add(69);
            * st.setMarks(list);
            */
            /*
             * //jdk1.2
             * Arrays List<Integer>
            */
            list=Arrays.asList(34,56,78,90,66);
            * st.setMarks(list);
            */

            //jdk9 Factory Collections ,Immutable Collections
            List<Integer> list=List.of(34,56,78,90,66);
            st.setMarks(list);

            //jdk1.7
            Set<String> set=new HashSet<>();
            set.add("m1"); set.add("m2"); set.add("m3");
            set.add("m4"); set.add("m5");
            st.setSubjects(set);

            //jdk1.7
            String g[]=new String[5];
            g[0]="A"; g[1]="B"; g[2]="C"; g[3]="D"; g[4]="E";
            st.setGrades(g);
        }
    }
}
```

```
ObjectMapper ob=new ObjectMapper();
String json=ob.writeValueAsString(st);
System.out.println(json);
} catch (Exception e) {

}

}
Output::

{
  "stdId":101,
  "stdName":"Sankar",
  "stdFees":23443.0,
  "marks":[34,56,78,90,66],
  "subjects":["m1","m2","m3","m4","m5"],
  "grades":["A","B","C","D","E"]
}
```

Working with JSON-MAP and Properties Collections::

->MAP (I) or Properties(I) will store data in Key=Value format. It is almost similar to JSON Object only.

Syntax is::

```
{key:Value, key:Value, key:Value,...}
```

Example::

Step1::

Create maven Project and Add Jersey -media-json-jackson Dependency.

Step2:: Create model class as

```
package in.nit.model;
```

```
import java.util.Map;
```

```
import java.util.Properties;  
  
import lombok.Data;  
@Data  
public class Product {  
  
    private int pId;  
    private String pCode;  
    private Map<String, String> models;  
    private Properties info;  
}
```

Step3:: Test class

```
package in.nit.test;  
  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Properties;  
  
import com.fasterxml.jackson.databind.ObjectMapper;  
  
import in.nit.model.Product;  
  
public class Test {  
  
    public static void main(String[] args) {  
        try {  
  
            Product p1=new Product();  
  
            Map<String, String> map=new HashMap<>();  
            map.put("m1", "RED"); map.put("m2", "BLUE");  
            map.put("m3", "GREEN");  
  
            Properties p=new Properties();  
            p.put("v1", "3.2"); p.put("v2", "5.6");  
  
            p1.setPId(101);  
            p1.setPCode("abc");  
            p1.setModels(map);  
            p1.setInfo(p);  
        }  
    }  
}
```

```
ObjectMapper om=new ObjectMapper();
String json=om.writeValueAsString(p1);
System.out.println(json);

} catch (Exception e) {
}

}

}
```

JSON Output::

```
{
  "models": {
    "m1": "RED",
    "m2": "BLUE",
    "m3": "GREEN"
  },
  "info": {
    "v1": "3.2",
    "v2": "5.6"
  },
  "PCODE": "abc",
  "pid": 101
}
```

Jersey JSON HAS-A Relation

->Using Child class as a Datatype and creating a variable in Parent class is called as HAS-A Relation.

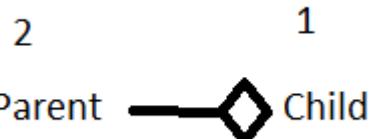
->Coding order Child first and Parent next.

->In this case for two classes, two objects are created, but JSON is only one.

->It means Parent JSON is created with Child JSON(Inner JSON).

Example::

HAS-A



Step1:create Java Project Name :: JsonJackson-4_HAS-A_Relation

Step2: in pom.xml

```
<properties>
    <maven.compiler.source>15</maven.compiler.source>
    <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
    <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
    <dependency>
        <groupId>org.glassfish.jersey.media</groupId>
        <artifactId>jersey-media-json-jackson</artifactId>
        <version>2.32</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.16</version>
        <scope>provided</scope>
    </dependency>
```

Step3::Child class

package in.nit.model;

```
import lombok.Data;
@Data
public class Model {

    private int mld;
    private String mCode;
```

}

Step4:: Parent class

```
package in.nit.model;

import lombok.Data;

@Data
public class Product {
    private int pId;
    private String pName;
    private Model mob;//HAS-A
}
```

Step5: Test class

```
package in.nit.test;

import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Model;
import in.nit.model.Product;

public class Test {

    public static void main(String[] args) {
        try {

            Model m=new Model();
            m.setMId(101);
            m.setMCode("ABC");
            Product p=new Product();
            p.setPId(102);
            p.setPName("Sankar");
            p.setMob(m);//link

            ObjectMapper om=new ObjectMapper();
            String json=om.writeValueAsString(p);
            System.out.println(json);

        } catch (Exception e) {
        }
    }
}
```

```
 }  
 }
```

JSON Output::

```
{  
 "mob":  
 {  
   "mcode": "ABC",  
   "mid": 101  
 },  
 "pid": 102,  
 "pnmae": "Sankar"  
}
```

Child class::

package in.nit.model;

```
import lombok.Data;  
@Data  
public class Address {  
  
    private String hNo;  
    private String hLoc;  
}
```

Parent class::

package in.nit.model;

```
import lombok.Data;

@Data
public class Employee {
    private int eId;
    private String eName;
    private Address eAdd;//HAS-A
}

Test class::

package in.nit.test;

import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Address;
import in.nit.model.Employee;

public class Test {

    public static void main(String[] args) {
        try {

            Address a=new Address();
            a.setHNo("2-62");
            a.setHLoc("Razole");

            Employee e=new Employee();
            e.setEId(101);
            e.setENmae("Sankar");
            e.setEAdd(a);//link
            ObjectMapper om=new ObjectMapper();
            String json=om.writeValueAsString(e);
            System.out.println(json);

        } catch (Exception e) {
        }
    }
}
```

JSON OUTPUT::

```
{  
    "enmae": "Sankar",  
    "eid": 101,  
    "eadd": {  
        "hloc": "Razole",  
        "hno": "2-62"  
    }  
}
```

Writing JSON to File

- >JACKSON Converter API supports writing and reading data from JSON File(--.json).
- >File name ends with .json (or any is fine like .txt) .
- >Use class: ObjectMapper and Use method: writeValue(file,object): void
- >Here File means java.io.File object that holds location of JSON file. Here, it creates a new JSON File.
- >Object: Any class Object that needs to be converted to JSON Format.

Step1: create a java project :: JsonJackson-6_File

Step2: in pom.xml file

```
<properties>  
    <maven.compiler.source>15</maven.compiler.source>  
    <maven.compiler.target>15</maven.compiler.target>  
</properties>  
<dependencies>  
    <!--  
  
        <dependency>  
            <groupId>org.glassfish.jersey.media</groupId>  
            <artifactId>jersey-media-json-jackson</artifactId>
```

```
<version>2.32</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.16</version>
    <scope>provided</scope>
</dependency>
</dependencies>
```

Step3::Student.class

```
package in.nit.model;

import lombok.Data;

@Data
public class Student {
    private int sId;
    private String sName;
    private String sCourse;
    private double sFees;
}
```

Step4::TestSaveFile.class

```
package in.nit.test;

import java.io.File;

import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Student;

public class TestSaveFile {
```

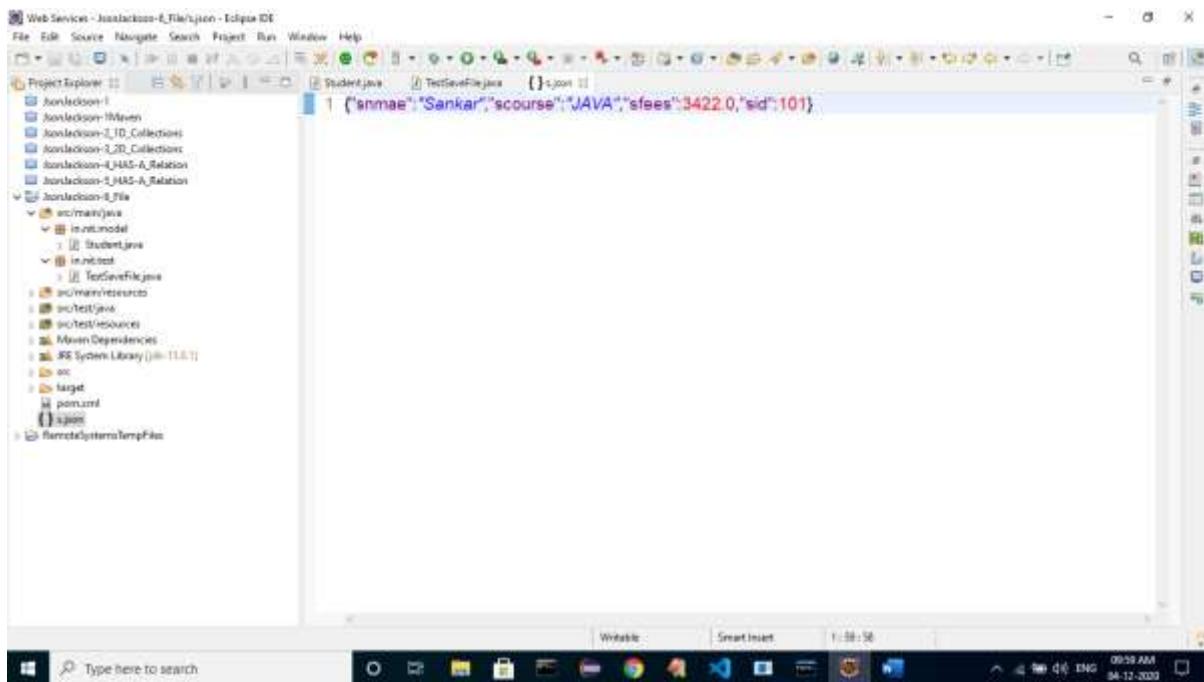
```
public static void main(String[] args) {  
    try {  
  
        Student s=new Student();  
        s.setSId(101);  
        s.setSNmae("Sankar");  
        s.setSCourse("JAVA");  
        s.setSFees(3422.00f);  
  
        ObjectMapper om=new ObjectMapper();  
        File f=new File("s.json");  
        om.writeValue(f,s);  
        System.out.println("Done");  
  
    } catch (Exception e) {  
    }  
  
}
```

Step4::Run App

Step5:Refresh App

See s.json file

For Any Queries: karrasankar@gmail.com
For Projects Github: <https://github.com/karrasankar158/Webservice>



Output::

s.json

{

```
"snmae": "Sankar",  
"scourse": "JAVA",  
"sfees": 3422.0,  
"sid": 101  
}
```

JSON-JACKSON read JSON as Object

->ObjectMapper(c) has provided one method `readValue()` (`String JSON, T.class`): `T Object`.

->Here T =Model class Object.

->readValue() method converts JSON String Format to java object Format .

->If key is not Present then returns Default Value.

->* Here, { } is a valid JSON that creates java Object using default Constructor.

Step1: create java Project :: JsonJackson-7_JSON-Object

Step2: in pom.xml file

```
<properties>
    <maven.compiler.source>15</maven.compiler.source>
    <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
    <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson -->
    <dependency>
        <groupId>org.glassfish.jersey.media</groupId>
        <artifactId>jersey-media-json-jackson</artifactId>
        <version>2.32</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.16</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Step3:: model class(Product)

package in.nit.model;

import lombok.Data;

```
@Data
public class Product {
    private int pid;
    private String pcode;
    private double pcost;
    //note Default Constructor,setter,getter toString must override
}
```

Step4:: `JsonObject.class`

```
package in.nit.test;

import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Product;
public class JsonObject {

    public static void main(String[] args) {
        try {
            String
json={"\"pid\":101,\"PCODE\":\"ABC\",\"PCOST\":343.00"};
            //note double values ,float values .00f,.00d not allowed
if you mention you not get output
            Product p=new Product();
            ObjectMapper om=new ObjectMapper();
            p=om.readValue(json,Product.class);
            System.out.println(p);
        } catch (Exception e) {
            }
    }
}

//or here get default values.
package in.nit.test;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Product;
public class JsonObject {

    public static void main(String[] args) {
        try {
            //String
            json={"\"pid\":101,\"PCODE\":\"ABC\",\"PCOST\":343.00"};
            String json="{}";
            //note double values ,float values .00f,.00d not allowed
            if you mention you not get output
                Product p=new Product();
                ObjectMapper om=new ObjectMapper();
                p=om.readValue(json,Product.class);
                System.out.println(p);
        } catch (Exception e) {

        }
    }
}
```

Output::

Product(pid=101, pcode=ABC, pcost=343.0)

->If input JSON String is a Invalid Format like Symbol “ is missing or any other then JACKSON throws: JSONParseException.

```
public static void main(String[] args) { here : missing , so getting JSONParseException
    try {
        String json={"pid":101,"pcode": "ABC","pcost":343.0};
        //String json="0";
        //note double values ,float values .00f,.00d not allowed if you
mention you not get output
        Product p=new Product();
        ObjectMapper om=new ObjectMapper();
        p=om.readValue(json,Product.class);
        System.out.println(p);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

->String JSON="{}";is a valid JSON i.e creating Object using default constructor.

String JSON="{}";

->String JSON=" "; is Invalid JSON MismatchedInputException:: No content to map due to end-of-Input.

```
try { here { missing .MismatchedInputException
    String json="\"pid\":101,\"pcode\": \"ABC\", \"pcost\":343.0";
    //String json=" ";
    //note double values ,float values .00f,.00d not allowed if you mention you not get output
    Product p=new Product();
    ObjectMapper om=new ObjectMapper();
    p=om.readValue(json,Product.class);
    System.out.println(p);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

->String JSON="{ }"; is Invalid JSON JSON.EOFException :Unexcepted end-of-input expected close marker.

->String JSON="}"; is Invalid JSON JSONParseException is thrown by JACKSON .

->String JSON={"pid":"10"}; is Valid.

Object created with pId=10(After Parsing).

->String JSON={"pid":"11.30"}; is Valid.

Object created with pId=11 (After Parsing).

JAVA DE-COMPLIER

->JAD download Link: <https://waraneckas.com/jad/>

->Click on "jad 1.5.8g for Windows 9.x/NT/2000"

->Extract folder

->Create one Java File(ex: A.java)

A{

}

->Compile Java File(cmd>javac A.java)

->create jad file(cmd>jad A.class)

->Open A.jad using Notepad or Editplus

JSON-JACKSON :Read JSON from File

->JACKSON API has provided class ObjectMapper(c) which as method is : readValue(file,class<T> clr);//To Object.

Here, T=class name for Object creation.

Step1:Create JSON with Data.

Example: prods.json

```
JSON
{
    "pld"      : 101,
    "pCode"    : "ABC";
    "pCost"    : 333.4;
}
```

Step2: Define Model class

```
package in.nit.model;

import lombok.Data;

@Data
public class Product {
    private int pid;
    private String pcode;
    private double pcost;
    //note Default Constructor,setter,getter toString must override
}
```

Step3:: Define Test class to read Data

```
package in.nit.test;

import java.io.File;

import com.fasterxml.jackson.databind.ObjectMapper;

import in.nit.model.Product;
public class JsonObject {
```

```
public static void main(String[] args) {  
    try {  
        File f=new File("prods.json");//prods.json  
        Product p=new Product();  
        ObjectMapper om=new ObjectMapper();  
        p=om.readValue(f,Product.class);  
        System.out.println(p);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

}

Prods.json::

```
{  
    "pid":101,  
    "PCODE": "ABC",  
    "pcost":33.44  
}
```

Output::

Product(pid=101, pcode=ABC, pcost=33.44)

//JDK 10

//Local variables type references(var)

->In JDK10, var reserve word is added into java Runtime.

->We need not to provide data for variables which is decided by java Compiler at the time of Compiling File.

->Compiler choose best/suitable DataType for variable based on data it holds.

Example:

| Source Code | Compiled Code |
|-------------------|---------------------|
| var id1=10; | byte b=10; |
| var id2=1000; | char c='142710'; |
| var id3=10000000; | int i=0x3b99ca00; |
| var id4=false; | boolean flag=false; |
| var id5=3.3; | double d=3.3D |
| var id6="Hello"; | String s="Hello"; |

->Invalid usages of var reserve word ...

- a) Using var as class name, interface name, enum name, annotation name is not allowed.
- b) Using var to create instance or static variables(at class level) is not allowed.
- c) Using var without any initial value not allowed.

Example: var a;//InValid

- d) Variable with null is InValid.

Example: var b=null;//InValid

- e) Changing variable value to another type is not allowed.

Example: var k=10;

K=5.5;//is InValid

Using var as valid types

- a) Var to create local variable with any one value(not a null).

Example: var a=-7;//Valid

- b) Var as method name valid.

Example: public void var() {};//valid

- c) var as reference variable type

Example: var a=new Employee();
var a1=new Thread();
var n=new String();

- d) var as Collection/Array variable

Example: var m=new ArrayList<String>();
Var k=new int[]{12,34,45,32,89};

- e) var var=10;//Also Valid.

Few More Valid Examples::

```
var a=10;  
var m=new Product();  
var k=new String("Hello");  
var p=new int[10,20,30,40];  
var f=new File("f.json");  
var n=new ArrayList<String>();  
var v=Boolean.valueOf("true");
```

Jackson Annotations

- Read + Write Annotations
 - `@JsonIgnore`
 - `@JsonIgnoreProperties`
 - `@JsonIgnoreType`
 - `@JsonAutoDetect`
- Read Annotations
 - `@JsonSetter`
 - `@JsonAnySetter`
 - `@JsonCreator`
 - `@JacksonInject`
 - `@JsonDeserialize`
- Write Annotations
 - `@JsonInclude`
 - `@JsonGetter`
 - `@JsonAnyGetter`
 - `@JsonPropertyOrder`
 - `@JsonRawValue`
 - `@JsonValue`
 - `@JsonSerialize`

GSon-Google Converter

->In pom.xml file

```
<properties>
    <maven.compiler.source>15</maven.compiler.source>
    <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
    <!--
        https://mvnrepository.com/artifact/com.google.code.gson/gson -->
        <dependency>
            <groupId>com.google.code.gson</groupId>
```

```
<artifactId>gson</artifactId>
<version>2.8.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.16</version>
    <scope>provided</scope>
</dependency>
        </dependencies>
```

- > Gson is a JSON Converter
- > Object -> JSON format.
- > Json to java Object format
- > it is faster compared to JACKSON older version
- > api

Class : Gson,GsonBuilder
Package: com.google.gson

Methods::

- a) toJson(obj):String
This method is used to convert java Object to JSON String format.
- b) toJson(obj,fileWriter):void
This method is used to convert Java Object to JSON File Format(_.json).
- c) fromJson(json,class<T>):T
This method is used to convert JSON String to Java Object of a class Type.
- d) fromJson(fileReader,class<T>):T
This method is used to convert JSON file to Java Object format of a class type.
- e) setPrettyPrinting():
Works only for toJson(), it will display data in Well arranged format.

-----Normal Output-----

```
{"eid":100,"ename":"sankar","esal":23.44}
```

-----Pretty(nice)print-----

```
{  
"eid":100,  
"ename":"sankar",  
"esal":23.44  
}
```

Note::

Above converter shows keys with null values,

Else normal code like Gson g=new Gson();

Will not display null keys in JSON String

Gson g=new GsonBuilder()

```
.serializeNulls()  
.setPrettyPrinting()  
.create();
```

-----Example1-----

Java Object to JSON Format

Step1: Define One Model class (Employee)

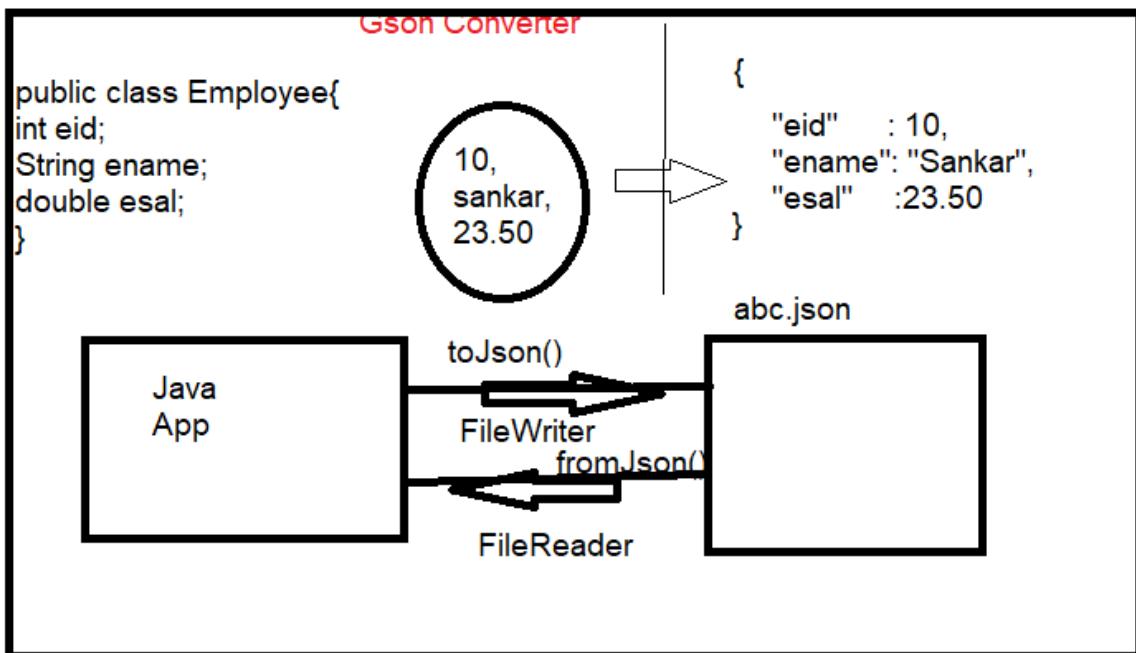
Step2: Test class

- a) Create Object to model class
- b) * Create Object to Gson class.
- c) * Call method toJson(Obj):String
- d) Print JSON.

-----Example2-----

JSON to Java Object Format

- a) Define One Model class (Employee)
- b) Test class
 1. Create JSON String.
 2. * Create Object to Gson class.
 3. * Call method fromJson(json,class<T>):T
 4. Print Object.



Step1: Model class://Employee.java

package in.nit.model;

import lombok.Data;

```

@Data
public class Employee {
    private int eid;
    private String eName;
    private double eSal;
}
    
```

Step2: Test class //ObjectToJsontest1.java

package in.nit.test;

import com.google.gson.Gson;

import in.nit.model.Employee;

public class ObjectToJsontest1 {

```

public static void main(String[] args) {
    try {
        
```

```
//create model class object
    Employee e=new Employee();
    e.setEId(101);
    e.setEName("Sankar");
    e.setESal(33.00D);
    //Create Gson class Object
    Gson g=new Gson();
    //Call toJson() method
    String json=g.toJson(e);
    //print Gson String
    System.out.println(json);
}
catch(Exception e) {
    e.printStackTrace();
}
}

Output::
{"eId":101,"eName":"Sankar","eSal":33.0}
```

Step2: ObjectToJsontest2

```
package in.nit.test;

import com.google.gson.Gson;

import in.nit.model.Employee;

public class ObjectToJsontest2 {

    public static void main(String[] args) {
        try {
//1.create model class object
        Employee e=new Employee();
```

```
e.setEId(101);
e.setEName("Sankar");
e.setESal(33.00D);
//2.Create Gson class Object
//3.Call toJson() method
String json=new Gson().toJson(e);
//4.print Gson String
System.out.println(json);
}
catch(Exception e) {
    e.printStackTrace();
}
}

}
```

Output::

```
{"eId":101,"eName":"Sankar","eSal":33.0}
```

Step2:: ObjectToJsontest3

```
package in.nit.test;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import in.nit.model.Employee;

public class ObjectToJsontest2 {

    public static void main(String[] args) {
        try {
//1.create model class object
```

```
Employee e=new Employee();
e.setEId(101);
e.setEName("Sankar");
e.setESal(33.00D);
//2.Create Gson class Object
//3.Call toJson() method
Gson g=new GsonBuilder()
    .serializeNulls()
    .setPrettyPrinting()
    .create();
String json=g.toJson(e);
//4.print Gson String
System.out.println(json);
}
catch(Exception e) {
    e.printStackTrace();
}
}

}
```

Output::

```
{
    "eId": 101,
    "eName": "Sankar",
    "eSal": 33.0
}
```

Example2::

Step1: Create Model class

```
package in.nit.model;

import lombok.Data;

@Data
public class Employee {
    private int eId;
    private String eName;
    private double eSal;
}
```

Step2:: g.json

```
{  
    "eId": 101,  
    "eName": "Sankar",  
    "eSal": 33.0  
}
```

Step3: JsonToObjectTest.class

```
package in.nit.test;  
  
import java.io.FileReader;  
import java.io.Reader;  
  
import com.google.gson.Gson;  
  
import in.nit.model.Employee;  
  
public class JsonToObjectTest {  
  
    public static void main(String[] args) {  
        try {  
            //1.create File Reader Object  
            Reader r=new FileReader("g.json");  
            //2.Create Gson class Object  
            Gson g=new Gson();  
            //3.Call toJson() method  
            Employee e=g.fromJson(r,Employee.class);  
  
            //4.print Gson String  
            System.out.println(e);  
        }  
        catch(Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Output::

Employee(eId=101, eName=Sankar, eSal=33.0)

-----Example3-----

Step1:: Create Model class

```
package in.nit.model;

import lombok.Data;

@Data
public class Employee {
    private int eId;
    private String eName;
    private double eSal;
}
```

Step2:: JsonToObjectTest.java

```
package in.nit.test;

import com.google.gson.Gson;

import in.nit.model.Employee;

public class JsonToObjectTest {

    public static void main(String[] args) {
        try {
//1.Define JSON String
        String
json={"eId":101,"eName":"Sankar","eSal":22.33};
//2.Create Gson class Object
        Gson g=new Gson();
//3.Convert JSON to Employee class
        Employee e=g.fromJson(json,Employee.class);

//4.print Gson String
        System.out.println(e);
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
}
```

Output::

Employee(eId=101, eName=Sankar, eSal=22.33)

-----Example4-----

Step1: Create one model class

package in.nit.model;

import lombok.Data;

```
@Data  
public class Employee {  
    private int eId;  
    private String eName;  
    private double eSal;  
}
```

Step2: **TestObjectToJsonFileTest.java**

package in.nit.test;

import java.io.FileWriter;

import java.io.Writer;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import in.nit.model.Employee;

public class TestObjectToJsonFileTest {

public static void main(String[] args) {

//JDK 7 auto closeable+ try with resources.

try(Writer writer=new FileWriter("g.json")) {

//1.Model class Object

```
Employee e=new Employee();
e.setEId(103);
e.setEName("Sankar");
e.setESal(34.88);
```

//2.Create Gson class Object

```
Gson g=new GsonBuilder()
.setPrettyPrinting()
.create();
```

//3.Convert JSON to Store into File

```
g.toJson(e,writer);
```

//4.print Gson String

```
System.out.println("Done"); writer.close();
```

```
}
```

```
catch(Exception e) {
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

Output::

Done

g.json:

```
{
  "eId": 103,
  "eName": "Sankar",
  "eSal": 34.88
}
```

@Expose(Gson API)

- >@Expose: (Gson API) This annotation is used to consider a variable(Field) to be taken into JSON conversion.
 - >In simple it is reverse to @Transient in JACKSON API.
 - >@transient will not take variables(Fields) into JSON.
 - >By Default @Expose is not going to work. It must be activated, like
- ```
Gson g=new GsonBuilder()
 .excludeFieldsWithoutExposeAnnotation()
 .create();
```

-----Code-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/com.google.code.gson/gson -->
 <dependency>
 <groupId>com.google.code.gson</groupId>
 <artifactId>gson</artifactId>
 <version>2.8.6</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.projectlombok/lombok -->
```

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

### Step1: Model class

```
package in.nit.model;

import com.google.gson.annotations.Expose;

import lombok.Data;

@Data
public class Student {
 //Note primitive types are not recommended take fields as wrapper
 //types.

 @Expose //meaning please use this property in JSON.
 private Integer stdId;

 @Expose//meaning please use this property in JSON.
 private String stdName;
 private Double stdFee;
 private String stdCourse;
 private String secureCode;
}
```

---

->@Expose: It consider a variable that is taken from object and shown in JSON even taken from JSON and placed in Object.

@Expose having two attributes

- serialize (default- true) [Object->JSON]
- deserialize(default=true)[JSON-object]

\*) if serialize=false then data exist in Object but not given to JSON String

\*) if deserialize=false then data exist in JSON but not given to Object.

\*) if deserialize=true/serialize=true then data exist in Object and JSON.

---

-----Example2-----

Model class

```
package in.nit.model;
```

```
import com.google.gson.annotations.Expose;
```

```
import lombok.Data;
```

```
@Data
```

```
public class Student1 {
```

```
 //Note primitive types are not recommended take fields as wrapper
 types.
```

```
 @Expose //meaning please use this property in JSON.
```

```
 private Integer stdId;
```

```
 @Expose(serialize = true, //take stdName from Obj->show in
 JSON
```

```
 deserialize = false) //Do not take stdName from JSON
 into Obj
```

```
 private String stdName;
```

```
 @Expose(serialize = false, //Data in Object (not taken into JSON)
```

```
 deserialize = true) //Data in JSON->Taken in object
```

```
 private Double stdFee;
```

```
//@Expose(deserialize = true) //default added as serialize=true
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

@Expose(serialize = false)//default added as deserialize=true

private String stdSource;

}

Step2: Test class

Object->JSON(serialize)

package in.nit.test;

import java.io.FileWriter;

import java.io.Writer;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import in.nit.model.Student;

public class ObjectToJsonTest {

    public static void main(String[] args) {

        try {

            //1.Model class Object

            Student s=new Student();

            s.setStdId(101);

            s.setStdName("Sankar");

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
s.setStdFee(34.89);
s.setStdSource("Web Services");
s.setSecureCode("gJGe");

//2.Create Gson class Object
Gson g=new GsonBuilder()
 .excludeFieldsWithoutExposeAnnotation()
 .create();
//3.call toJson() method
String json=g.toJson(s);

//4.print Gson String
System.out.println(json);

}

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

}

Output:: For Student s=new Student();
{"stdId":101,"stdName":"Sankar"}
```

Test class:

```
package in.nit.test;
```

```
import java.io.FileWriter;
import java.io.Writer;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import in.nit.model.Student1;

public class ObjectToJsonTest {

 public static void main(String[] args) {

 try {

 //1.Model class Object
 Student1 s=new Student1();
 s.setStdId(101);
 s.setStdName("Sankar");
 s.setStdFee(34.89);
 s.setStdSource("Web Services");

 //2.Create Gson class Object
 Gson g=new GsonBuilder()
 .excludeFieldsWithoutExposeAnnotation()
 .create();
 }
 }
}
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
//3.call toJson() method
String json=g.toJson(s);

//4.print Gson String
System.out.println(json);

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

}
```

Output : for Student1 s=new Student1();

{"stdId":101,"stdName":"Sankar"}

Model class::

package in.nit.model;

```
import com.google.gson.annotations.Expose;
```

```
import lombok.Data;
```

```
@Data
```

```
public class Student1 {
```

```
 //Note primitive types are not recommended take fields as wrapper
 types.
```

**@Expose** //meaning please use this property in JSON.

```
private Integer stdId;

 @Expose(serialize = true,//take stdName from Obj->show in
JSON
 deserialize = false)//Do not take stdName from JSON
into Obj
private String stdName;

 @Expose(serialize =false,//Data in Object (not taken into JSON)
 deserialize = true)//Data in JSON->Taken in object
private Double stdFee;

 @Expose(deserialize = true)//default added as serialize=true
//@Expose(serialize = false)//default added as deserialize=true
private String stdSource;
}
Test class::
package in.nit.test;

import java.io.FileWriter;
import java.io.Writer;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import in.nit.model.Student1;
```

```
public class ObjectToJsonTest {

 public static void main(String[] args) {

 try {

 //1.Model class Object
 Student1 s=new Student1();
 s.setStdId(101);
 s.setStdName("Sankar");
 s.setStdFee(34.89);
 s.setStdSource("Web Services");

 //2.Create Gson class Object
 Gson g=new GsonBuilder()
 .excludeFieldsWithoutExposeAnnotation()
 .create();
 //3.call toJson() method
 String json=g.toJson(s);

 //4.print Gson String
 System.out.println(json);

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

```
 }
}

}
```

Output::

```
{"stdId":101,"stdName":"Sankar","stdSource":"Web Services"}
```

---

Test class ::

```
//JSON->Object(deserialize)
```

```
package in.nit.test;
```

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import in.nit.model.Student1;
```

```
public class ObjectToJsonTest1 {
//deserialize
 public static void main(String[] args) {

 try {

 //1.String JSON
 String
 json={"\"stdId\":202,\"stdName\":\"Sankar\",\"stdFee\":23.98,\"stdSource\""
 :"Java"};

 //2.Create Gson class Object
 Gson g=new GsonBuilder()
.excludeFieldsWithoutExposeAnnotation() .create();
 //3.call toJson() method
 Student1 s=g.fromJson(json, Student1.class);
```

```
//4.print Gson String
System.out.println(s);

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

}
```

Output::

Student1(stdId=202, stdName=null, stdFee=23.98, stdCourse=Java)

---

## Working with HAS-A variable and Collection variables.

Step1: Model class:

```
package in.nit.model;

import lombok.Data;

@Data
public class Address {
 private Integer hno;
 private String loc;
}
```

Step2: Model class :

```
package in.nit.model;

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

import lombok.Data;

@Data
public class Employee {
```

```
private Integer empld;
private String empName;
private Address add;//HAS_A
private List<String> projects;
private Map<String, String> varsions;
}
```

Test clas::

```
package in.nit.test;
import java.io.FileWriter;
import java.io.Writer;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import in.nit.model.Address;
import in.nit.model.Employee;

public class ObjectToJsonTestAndFile {
 //Serialize
 public static void main(String[] args) {

 try {

 //1.Model class Object
 Address a=new Address();
 a.setHno(230);
 a.setLoc("hyd");
 Employee e=new Employee();
 e.setEmpld(294);
 e.setEmpName("Sankar");
 e.setAdd(a);//HAS-A
 //JDK9
 e.setProjects(List.of("p1","p2","p3","p4"));
 e.setVarsions(Map.of("p1","v1","p2","v2","p3","v3"));

 //2.Create Gson class Object
 }
 }
}
```

```
Gson g=new GsonBuilder()
 .setPrettyPrinting()
 .serializeNulls()//show null values
 .create();

 //3.call toJson() method
 String json=g.toJson(e);

 //4.print Gson String
 System.out.println(json);
 //Write Same Date To file
 Writer w=new FileWriter("s.json");
 g.toJson(e,w);
 System.out.println("Done");
 w.close();

 }

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

}
```

Output::

```
{
 "emplId": 294,
 "empName": "Sankar",
 "add": {
 "hno": 230,
 "loc": "hyd"
 },
 "projects": [
 "p1",
 "p2",
 "p3",
 "p4"
],
 "varsions": {
 "p1": "v",
 "p2": "v1",
 "p3": "v2",
 "p4": "v3"
 }
}
```

```
 "p4": "v3"
}
}
Done
```

Output s.json::

```
{
 "empld": 294,
 "empName": "Sankar",
 "add": {
 "hno": 230,
 "loc": "hyd"
 },
 "projects": [
 "p1",
 "p2",
 "p3",
 "p4"
],
 "varsions": {
 "p1": "v",
 "p2": "v1",
 "p3": "v2",
 "p4": "v3"
 }
}
```

---

@)Where can we add @Expose?

- A) Only at variable level not even at method or class.

# JAXB / JSONB

->Java Architecture for Xml Binding(JAXB)

## What is JSON-B?

JSON-B is a standard binding layer and API for converting Java objects to and from JSON documents. It's similar to Java Architecture for XML Binding (JAXB), which is used to convert Java objects to and from XML.

A) Marshalling: Converting Object to XML Format.

B) UnMarshalling: Converting XML to Object .

-> \* Every class Object can not be converted to XML Format.

Only JAXB class(class+JAXB Annotations) objects can be converted to XML Format.

---

XML:

HTML= Hyper text Markup Language used to design Web pages.

XML=eXtensible Markup Language used to Store data(Global Data Format) in tags format.

->XML is tag based language.

->XML Stores data.

->1TAG =1 Element+0-n attributes.

-><employee eid='101'>SAM</employee>

Here employee = element name

eid = attribute name

101 = attribute value

<employee eid = '101'>=Open tag/TAG

</employee> = Close tag

SAM = DATA/Tag Data

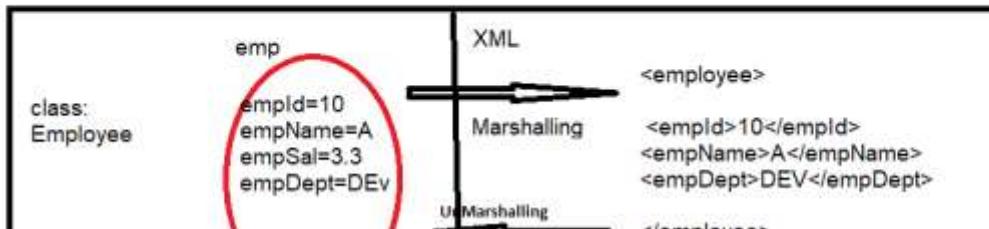
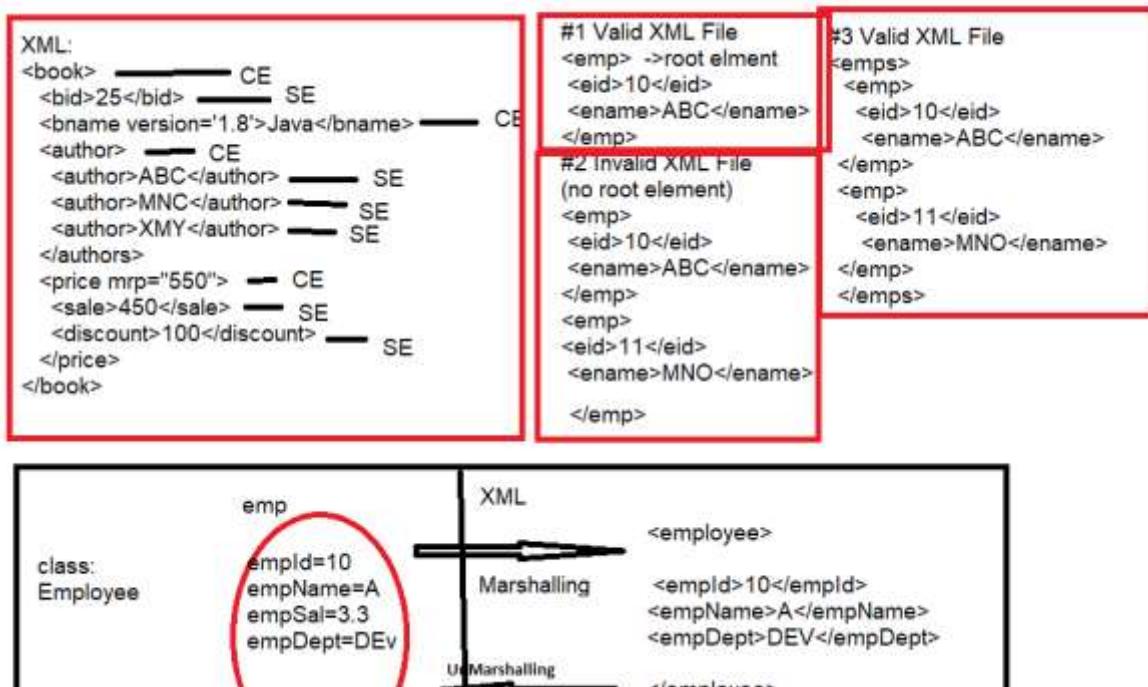
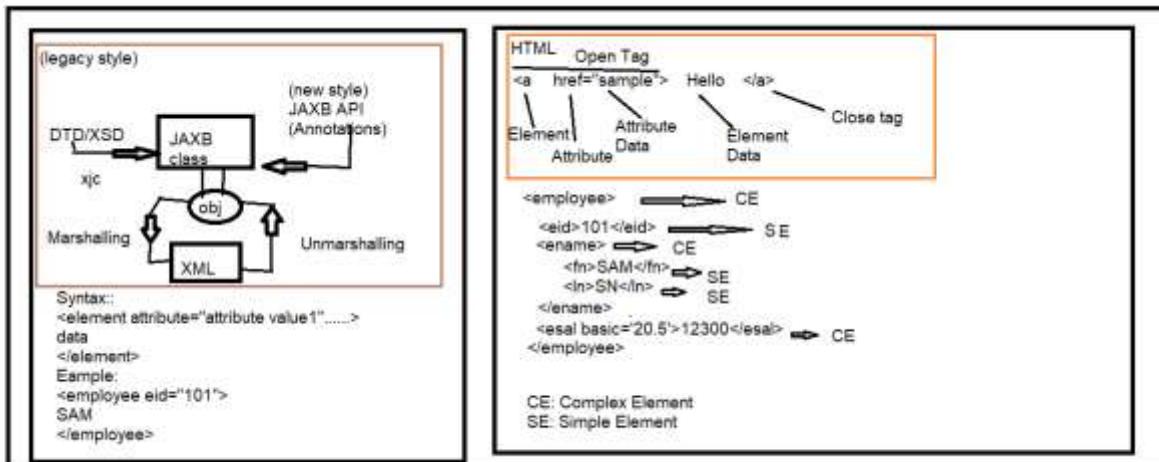
->Attribute must have value either single quoted(') or double quoted(")

<employee eid="101" ... (valid)

<employee eid='101'..(valid)

<employee eid='101'..(InValid)

---



## Types of elements:

- Complex Elements (CE) : An element that contains either child element or attribute (even both)
- Simple Elements(SE): An element that contains only Data(no child/no attribute).

- >Root Element: Every XML File /Data must contains root element.
- > An element that contains all other elements as child.
- >It should be unique (no duplicate).
- >One XML file contains only one root element.

Example:

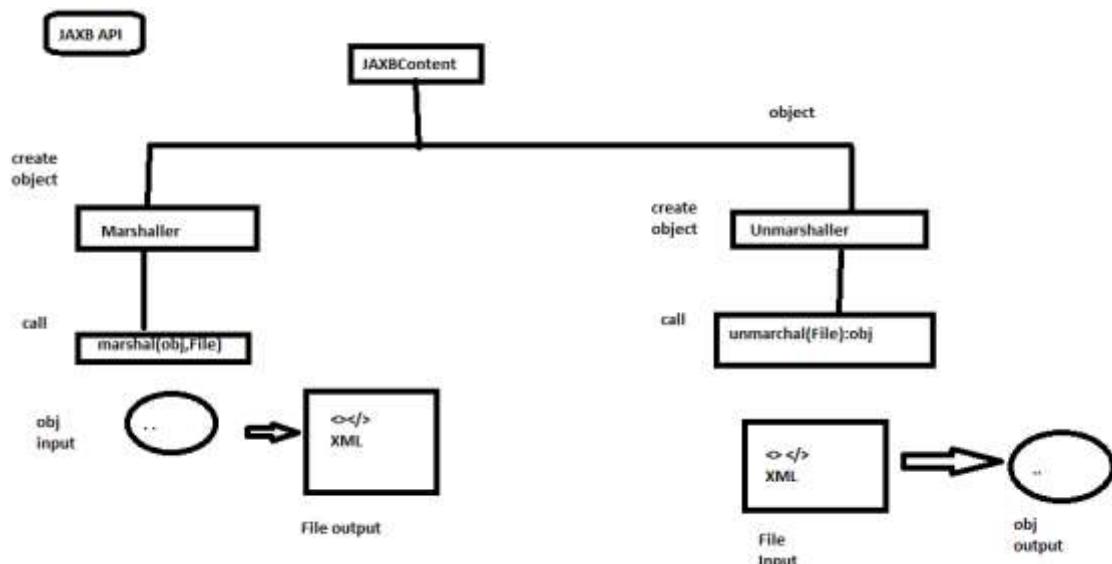
```
web.xml =><web-app> root element
springConfig.xml =><beans>root element.
hibernate.cfgs.xml =><hibernate-configuration>root-element
```

## Valid XML File must have root element

->XML tags are case-sensitive, HTML tags are case-Insensitive

i.e:

```
<html></HTML>(valid in HTML)
<employee></Employee>(invalid in XML)
<Employee><Employee>(valid)
<EMPLOYEE></EMPLOYEE>(valid)
<EmPLoyee></EmPLoyee>(valid)
```



## JAXB Operations

Java class

Object $\leftarrow\rightarrow$  XML

Marshalling: Object => XML

Unmarshalling: XML=>Object

-----Steps-----

Step1: Create Object to JAXBContent

Step2: Create Marshaller/Unmarshaller Object

Step3: call

a.marshal(Object,File):void → this method is used to convert Java Object to XML Format into one file.

b.unmarshal(File):Object → This method is used to convert File Data[XML File] to Java Object Format.

---

\*\*Note:

Define one class+JAXB Annotations → JAXBContent class

-> \* ClassName behaves as RootElement

-> \* Request/Must apply annotation is: @XmlRootElement.

(all other annotations are optional).

-----Code-----

Step1: in pom.xml file

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api
-->
 <dependency>
```

```
<groupId>javax.xml.bind</groupId>
<artifactId>jaxb-api</artifactId>
<version>2.3.1</version>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.3</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>
</dependencies>
```

Step2: Define One model class

```
package com.nt.test;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.Data;

//class+JAXB Annotations ==>JAXB Class
@XmlRootElement//<employee>.....</employee>
@Data
public class Employee {

 private Integer empld;
 private String empName;
 private Double empSal;
 private String empDept;
```

}

Step2: MarshallingObjectToXmlTest.class

```
package in.nit.model;

import java.io.File;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import com.nt.test.Employee;

public class MarshallingObjectToXmlTest {
//Object->XML
 public static void main(String[] args) {
 try {
 //Output XML File Located with Name
 File f=new File("g.xml");
 //1.Create Model class Object
 Employee e =new Employee();
 e.setEmpId(101);
 e.setEmpName("Sankar");
 e.setEmpSal(34.88D);
 e.setEmpDept("Dev1");
 //2.Create JAXBContent Object
 //It supports converting only Employee class:Object<-
>XML
 JAXBContext
context=JAXBContext.newInstance(Employee.class);
 //3.create Marshaller Object using context Object
 Marshaller marshaller=context.createMarshaller();
 //4.Call mthod marshal
 marshaller.marshal(e, f);
 //print any message
 System.out.println("Done");
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

}

Output::

Done

Note: Refresh Project

g.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<employee>
 <empDept>Dev1</empDept>
 <empld>101</empld>
 <empName>Sankar</empName>
 <empSal>34.88</empSal>
</employee>
```

Step5:

Open XML file either using any browser or using text editor (notepad/edit plus) .in Eclipse double click on file

->ClassName behaves as root element(first letter lower case).

---

## Working on Annotations:

- a) XmlRootElement: This annotation must be applied at model class level so, that JAXB Marshalling/Unmarshalling is going work, else not.

Example1: @XmlRootElement  
Class Employee{

In XML File

<employee>....</employee>

}

Example2: @XmlRootElement(name="emp-data") In Xml File  
Class Employee{

<emp-data>..</emp-data>

}

---

- b) @XmlAttribute: By default every variable behaves like element/tag.  
To convert any variable as attribute use this annotation.

(Optional)

Example1:

```
@XmlRootElement
Class Employee{
 @XmlAttribute
 private Integer empld; <employee empld='10'>...</employee>
}
```

- We can even provide element name externally, given as:

Example2::

```
@XmlRootElement XML File Output
Class Employee{ <employee eid='10'>..</employee>
 @XmlAttribute(name="eid")
 private int empld;
}
```

- 
- c) @XmlTransient: To avoid any variable taking into XML Operations (Marshalling/Unmarshalling) then use this annotation.

(Optional)

->Variable exist in model, may be stored in DB and also shown at UI but not shown at XML File.

Example1:

```
@XmlRootElement In XML File <empDept> is not displayed.
Class Employee{
 @XmlTransient
 private String empDept;
}
```

- 
- d) @XmlElement: To provide element details like name., we can use this by default variable name is taken as element name. To modify That use this annotation.

Example1:

```
@XmlRootElement
Class Employee{
 @XmlElement(name="ename") In XML File
 String empName; <employee>
 <ename>AA</ename>
}
```

→ To enable above annotations(b),(c) and (d) write @XmlAccessorType over model class

Example:

```
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
Public class Employee{
```

.....

.....

}

-----Code-----

Step1: In Pom.xml File

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api
-->
 <dependency>
 <groupId>javax.xml.bind</groupId>
 <artifactId>jaxb-api</artifactId>
 <version>2.3.1</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
```

```
<version>2.3.3</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

## Step2:Model class

```
package com.nt.test;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

import lombok.Data;

//class+JAXB Annotations ==>JAXB Class
@XmlRootElement//<employee>.....</employee>
//Enable JAXB Annotations which are defined on FIELDS(variables)
@XmlAccessorType(XmlAccessType.FIELD)
@Data
public class Employee {
 @XmlAttribute(name = "employee-id")
 private Integer empld;
```

```
@XmlElement(name="employee-name")
 private String empName;
@XmlElement(name="employee-salary")
 private Double empSal;
@XmlElement(name="employee-project")
private String empProjects;
@XmlElement(name="employee-experience")
private Double empExp;
//Ignore any variable
@XmlTransient
 private String empDept;

}
```

### Step3: Test Class

```
package in.nit.model;

import java.io.File;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import com.nt.test.Employee;

public class MarshallingObjectToXmlNewTest {
//Object->XML
 public static void main(String[] args) {
 try {
 //Output XML File Located with Name
 File f=new File("g.xml");
 //1.Create Model class Object
 Employee e =new Employee();
 e.setEmpId(101);
 e.setEmpName("Sankar");
 e.setEmpSal(34.88D);
 e.setEmpProjects("medical ");
 e.setEmpExp(2.5D);
 e.setEmpDept("Dev1");
 //2.Create JAXBContent Object
```

```
//It supports converting only Employee
class:Object<->XML
 //method newInstance says work for Employee
Object<->XML
 JAXBContext
context=JAXBContext.newInstance(Employee.class);
 //3.create Marshaller Object using context Object
Marshaller
marshaller=context.createMarshaller();
 //4.Call method marshal
 //emp->XML->File(g.xml)
marshaller.marshal(e, f);
 //print any message
System.out.println("Done");
} catch (Exception e) {
 e.printStackTrace();
}

}

}
```

Output::

Done

Note: Refresh project,

g.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<employee employee-id="101">
 <employee-name>Sankar</employee-name>
 <employee-salary>34.88</employee-salary>
 <employee-project>medical </employee-project>
 <employee-experience>2.5</employee-experience>
</employee>
->see empDept value is not coming because of @XmlTransient
used.
```

->If XML element value is NULL then tag is not displayed.  
Default values are:int-0,double-0.0,String-null,Boolean=false.

# Unmarshalling

->Steps1:

->Define Existed XML File location using java.io.File Object

->File f=new File("g.xml");

Step2: Create JAXBContext Object for Model class

->JAXBContext

context=JAXBContext.newInstance(Employee.class);

Step3: Create Unmarshaller Object using method

createUnmarshaller() which is defined in JAXBContext.

->Unmarshaller um=context.createUnmarshaller();

Step4: Call method unmarshal() which takes java.io.File as input and returns Object(java.lang) which must be downcasted.

->Object ob=um.unmarshal(f);

->Employee emp=(Employee)ob;

Step5: Print data.

->System.out.println(emp);

->In Pom.xml File

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api
-->
 <dependency>
 <groupId>javax.xml.bind</groupId>
 <artifactId>jaxb-api</artifactId>
 <version>2.3.1</version>
 </dependency>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.3</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>
</dependencies>
```

->In Model class

```
package com.nt.test;
```

```
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
```

```
import lombok.Data;
```

```
//class+JAXB Annotations ==>JAXB Class
@XmlRootElement//<employee>.....</employee>
```

```
//Enable JAXB Annotations which are defined on
FIELDS(variables)
@XmlAccessorType(XmlAccessType.FIELD)
@Data
public class Employee {
 @XmlAttribute(name = "employee-id")
 private Integer empId;
 @XmlElement(name="employee-name")
 private String empName;
 @XmlElement(name="employee-salary")
 private Double empSal;
 @XmlElement(name="employee-project")
 private String empProjects;
 @XmlElement(name="employee-experience")
 private Double empExp;
 //Ignore any variable
 @XmlTransient
 private String empDept;
}
```

g.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<employee employee-id="101">
 <employee-name>Sankar</employee-name>
 <employee-salary>34.88</employee-salary>
 <employee-project>medical </employee-project>
 <employee-experience>2.5</employee-experience>
</employee>
```

->Test class

```
package in.nit.model;

import java.io.File;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Unmarshaller;
```

```
import com.nt.test.Employee;

public class MarshallingObjectToXmlNewTest {
//Object->XML
 public static void main(String[] args) {
 try {
 //Output XML File Located with Name
 File f=new File("g.xml");

 //Create JAXBContext object for Model class
 JAXBContext
context=JAXBContext.newInstance(Employee.class);
 //3.createUnmarshal Object using context Object
 Unmarshaller un=context.createUnmarshaller();
 //4.Call mthod marshal
 //unmarshal
 Object ob=un.unmarshal(f);
 //print any message
 System.out.println(ob);
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

Output::

---

Employee(emplId=101, empName=Sankar, empSal=34.88,  
empProjects=medical , empExp=2.5, empDept=null)

```
package in.nit.model;

import java.io.File;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Unmarshaller;

import com.nt.test.Employee;
```

```
public class UnMarshallingObjectToXmlNewTest1 {
//Object->XML
 public static void main(String[] args) {
 try {
 //Output XML File Located with Name
 File f=new File("g.xml");

 //Create JAXBContext object for Model class
 JAXBContext context=JAXBContext.newInstance(Employee.class);
 //3.createUnmarshaller Object using context Object
 Unmarshaller un=context.createUnmarshaller();
 //4.Call method marshal
 //unmarshal
 Object ob=un.unmarshal(f);
 //print any message
 //toString ->Dynamic polymorphism ,call to an
 overridden method.
 //Downcasting Resulting Data

 Employee emp1=new Employee();
 System.out.println(emp1.getEmpProjects());//if you call
like you get null value.

 if(ob instanceof Employee) {//It Avoids
ClassCastException
 Employee emp=(Employee) ob;
 System.out.println(emp);
 System.out.println("-----");
 System.out.println(emp.getEmpProjects());
 System.out.println("-----");
 System.out.println(emp.getEmpExp());
 }

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

Instaceof:::

```
package com.nt.test;
class Student{
}
public class Employee extends Student {
 public static void main(String[] args) {

 //Type comparison operator is instanceof
 Employee e=new Employee();
 System.out.println(e instanceof Student);//true

 //If we apply the instanceof operator with any variable that
 has null value,it return false.
 Student s=null;
 System.out.println(s instanceof Employee);//false

 //When subclass type refers to the Object of parent class , it
 it also called as downcasting.
 //If we perform it direclty , compiler gives compilation error.
 //Employee e1=new Student();

 //If You perform it by TypeCasting, then ClassCastException
 thrown at runtime
 //Employee e2=(Employee)new Student();
 //System.out.println(e2);

 //but if we use instanceof operator, downcasting is possible
 Student s1=(Student)e;//copy employee object to student..
 if(s1 instanceof Employee) {
 Employee e2=(Employee)s1;
 System.out.println("DownCasting Done");
 }
 }
}
```

```
package com.nt.test;
class Student{
}
public class Employee extends Student {
 public static void main(String[] args) {

 //Type comparison operator is instanceof
 Employee e=new Employee();
 System.out.println(e instanceof Student);//true

 //If we apply the instanceof operator with any variable that has null value,it return false.
 Student s=null;
 System.out.println(s instanceof Employee);//false

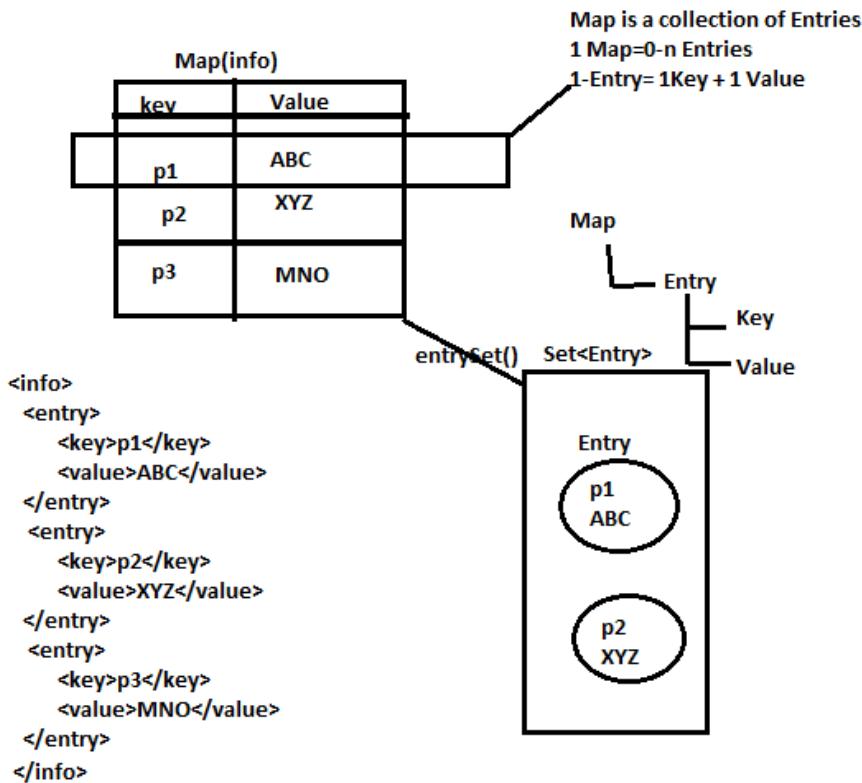
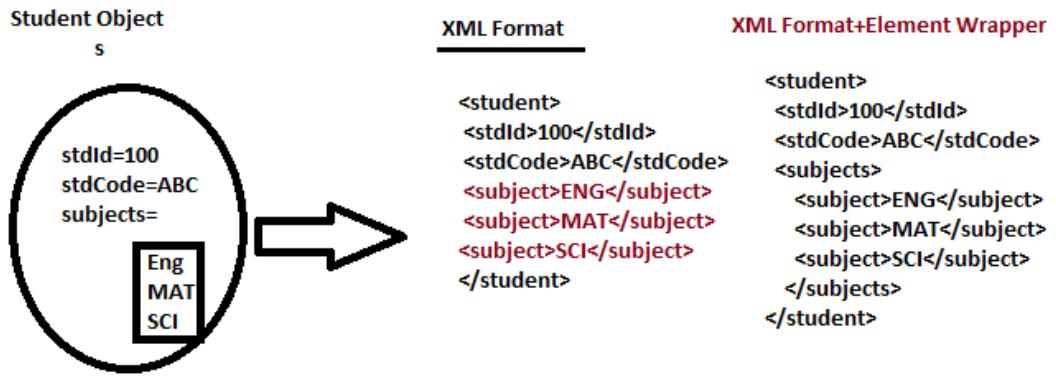
 //When subclass type refers to the Object of parent class , it is also called as downcasting.
 //If we perform it directly , compiler gives compilation error.
 //Employee e1=new Student();

 //If You perform it by TypeCasting, then ClassCastException thrown at runtime
 //Employee e2=(Employee)new Student();
 //System.out.println(e2);

 //but if we use instanceof operator, downcasting is possible
 Student s1=(Student)e;//copy employee object to student..
 if(s1 instanceof Employee) {
 Employee e2=(Employee)s1;
 System.out.println("DownCasting Done");
 }
 }
}
```

---

## Collection Type:List,Set,Map,Properties



->List: in this case, it holds multiple values.

->For every value variable is taken as element name(Tag name).  
 ->i.e if List contains 20 values then variable names comes for 20 times with one by one value in XML File.

->\*\* Consider a variable

subj: List<String>

Which is defined in Model class. Then simple Output for XML looks like:

<student>

.....

```
<subj>ENG</subj>
<subj>MAT</subj>
<subj>SCI</subj>
.....
</student>
```

---

- ➔ ElementWrapper: It is a tag/element added for wrapping all list/set collection tags. In simple parent tag for all list type tags.
- ➔ Writing Element Wrapper is optional.
- ➔ It make XML more readable.

Example:::

In pom.xml file::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api
-->
 <dependency>
 <groupId>javax.xml.bind</groupId>
 <artifactId>jaxb-api</artifactId>
 <version>2.3.1</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <dependency>
 <groupId>com.sun.xml.bind</groupId>
```

```
<artifactId>jaxb-impl</artifactId>
<version>2.3.3</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>
</dependencies>
```

Create model class:

```
package com.nt.test;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

import lombok.Data;

//class+JAXB Annotations ==>JAXB Class
@XmlRootElement//<employee>.....</employee>
//Enable JAXB Annotations which are defined on FIELDS(variables)
@XmlAccessorType(XmlAccessType.FIELD)
@Data
public class Student {
```

```
private Integer stdId;
private String stdCode;
@XmlElementWrapper(name = "all-subjects")
@XmlElement(name="subj")
private List<String> subject;

}
```

Note :: List.of() method is in jdk9 ,so use jdk 9 or above and must Jdk9 or above in project facets of eclipse

Test class::

```
package in.nit.model;

import java.io.File;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import com.nt.test.Student;

public class UnMarshallingObjectToXmlNewTest {
//Object->XML
 public static void main(String[] args) {
 try {
 //Create Model class Object
 Student s=new Student();
 s.setStdId(100);
 s.setStdCode("ABC");
 //jdk9
 s.setSubject(List.of("ENG","MAT","SCI"));

 //or
 //Generic type safe<>
 //List<String> list=new ArrayList<>();
 //list.add("ENG"); list.add("MAT"); list.add("SCI");
 //s.setSubject(list);

 //or
 // List<String> list=List.of("ENG","MAT","SCI");
 //s.setSubject(list);
```

```
//Create JAXBContext object for Model class
JAXBContext
context=JAXBContext.newInstance(Student.class);
//3.create marshaller Object using context Object
Marshaller m=context.createMarshaller();
//4.Call mthod marshal
m.marshal(s, new File("g.xml"));
//print any message
System.out.println("Done");
} catch (Exception e) {
 e.printStackTrace();
}
}

}
```

Output::

Done

g.xml::

note: refresh Project

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<student>
 <stdId>100</stdId>
 <stdCode>ABC</stdCode>
 <all- subjects>
 <subj>ENG</subj>
 <subj>MAT</subj>
 <subj>SCI</subj>
 </all-subjects>
</student>
```

---

->**Map:** It holds data in key-value format.  
Map holds data in Set of Entries format.  
i.e 1 Map =0-n Entries.

1 Entry =1 key + 1 Value

XML Format for Map is::

```
<mapElement>
 <entry>
 <key>--</key>
 <value>---</value>
 </entry>
 <entry>.....
.....
</mapElement>
```

#### -----Example-----

Model class::

```
package com.nt.test;
import java.util.Map;
import javax.xml.bind.annotation.XmlRootElement;

import lombok.Data;

@XmlRootElement
@Data
public class Company {

 private Integer cld;
 private Map<String, String> projects;
}
```

Test class:

```
package in.nit.model;
import java.io.File;
import java.util.Map;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import com.nt.test.Company;
public class UnMarshallingObjectToXmlNewTest {
//Object->XML
 public static void main(String[] args) {
 try {
 //Create Model class Object
 Company c=new Company();
 c.setCld(101);
```

```
//jdk9
c.setProjects(Map.of("p1","v1","p2","v2","p3","v3"));

//Create JAXBContext object for Model class
JAXBContext
context=JAXBContext.newInstance(Company.class);
//3.create marshaller Object using context Object
Marshaller m=context.createMarshaller();
//4.Call mthod marshal
m.marshal(c, new File("g.xml"));
//print any message
System.out.println("Done");
} catch (Exception e) {
 e.printStackTrace();
}

}

}
```

Output::

Done

Note:refresh Project

g.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<company>
<CId>101</CId>
<projects>
<entry>
<key>p1</key>
<value>v1</value>
</entry><entry>
<key>p2</key>
<value>v2</value>
</entry>
<entry>
<key>p3</key>
<value>v3</value>
</entry>
</projects>
</company>
```

## -----Core Java Concept-----

```
package com.nt.test;

import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

/*
 * @) How to Iterate Map?
 * A) Map should be converted to Entry Set Format(Set<Entry>)
 * Then it can be iterated.
 *
 */

public class MapTest {
 public static void main(String[] args) {
 //Adding values in HashMap
 Map<String, String> m=new HashMap<>();
 m.put("p1", "ABC");
 m.put("p2", "XYZ");
 m.put("p3", "MNO");

 //Set Entry Format
 Set<Entry<String, String>> set=m.entrySet();

 //iterate
 Iterator<Entry<String, String>> itr=set.iterator();

 //loop...multiple values so use loop
 while(itr.hasNext()) {
 Entry<String, String> entry=itr.next();
 //read keys and values from Map
 System.out.println(entry.getKey()+"-"+entry.getValue());
 }
 System.out.println("-----");
 //read only values from Map
 Collection<String> values=m.values();
 System.out.println(values);
 System.out.println("-----");
 //read only keys from Map
 }
}
```

```
Set<String> keys=m.keySet();
System.out.println(keys);
```

```
}
```

Output::

```
p1-ABC
p2-XYZ
p3-MNO
```

---

```
[ABC, XYZ, MNO]
```

---

```
[p1, p2, p3]
```

| JSON APIs For Java      | GivenBy                                 | Use                                                                                                                       | Performance             | Dependencies                                                                       |
|-------------------------|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------|-------------------------|------------------------------------------------------------------------------------|
| JACKSON                 | Open Source given by Jersey company     | Convert java(Object) to JSON or Convert JSON to java(Object)                                                              | slow                    | <a href="#">jersey-media-json-jackson</a>                                          |
| GSON (Google)           | Open Source given by Google company     | Convert java(Object) to JSON or Convert JSON to java(Object)<br>pretty printing possible here.<br>Versioning support      | faster                  | <a href="#">GSON</a>                                                               |
| JSON-B(Binding)<br>JAXB | Open Source given by SUN/ORACLE company | Convert java(Object) to JSON or Convert JSON to java(Object)<br>Convert java(Object)to XML<br>Convert XML to java(Object) | compare to Gson<br>slow | <a href="#">jaxb-api</a><br><a href="#">jaxb-impl</a><br><a href="#">jaxb-core</a> |

## Web Services

->Integration of two different applications which might be running in same Server or different Server, also applications developed in same /Different language.

->To do integration, minimum two applications are required, in that one application behaves like **Producer** and another application behaves like **Consumer**.

->Every application will be developed using three Layers. Those are::

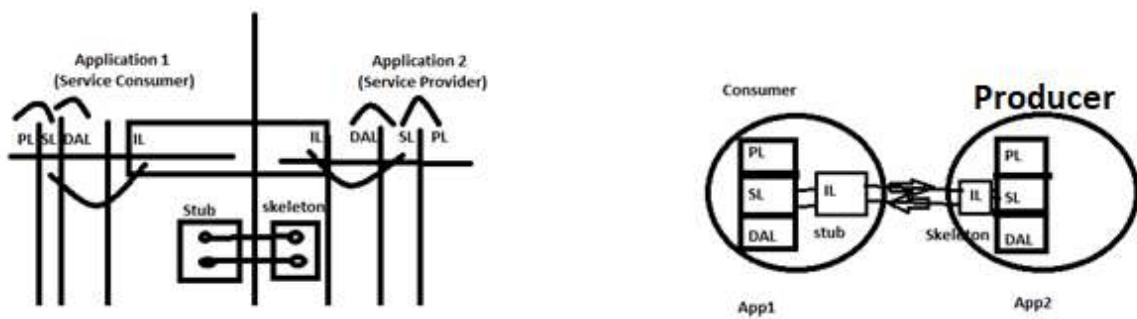
- 1) Presentation Layer (PL)

- 2) Service Layer (SL)
- 3) Data Access Layer (DAL)

->To integrate application these three layers cannot be used. We need fourth layer, i.e **Integration Layer (IL)** at both applications side.

->Every integration layer will be connected to its respected Service Layer (i.e same application) and another application's Integration Layer.

It looks like below:

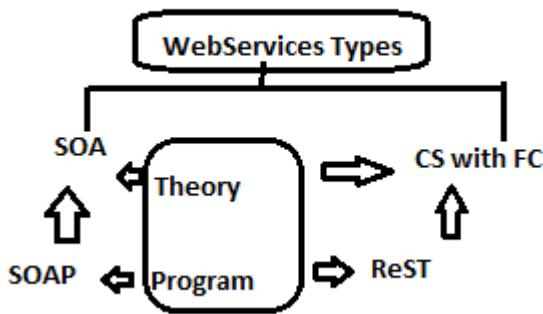


->Once request is made to Producer Application, Producer returns response after processing with HTTP Status.

->Every HTTP Status will be having one code and its equal message. Those are given as:

| Code      | Message           |
|-----------|-------------------|
| 1xx ----- | Information       |
| 2xx ----- | Success           |
| 3xx ----- | Redirect          |
| 4xx ----- | Client Side Error |
| 5xx ----- | Server Side Error |

WebServices Types:



->Based on Design Patterns and Theory concepts (Architecture and Flow )Webservices is divided into two types::

- 1) SOAP (Old Coding/Old Model)
- 2) ReST (New Coding/New Model)

->SOAP follows SOA Design Pattern for implementation by using XML as mediator language.

## SOA(Service Oriented Architecture):

->It is a Design pattern used to link two different applications which are developed in either same language or different language.

It has three components:

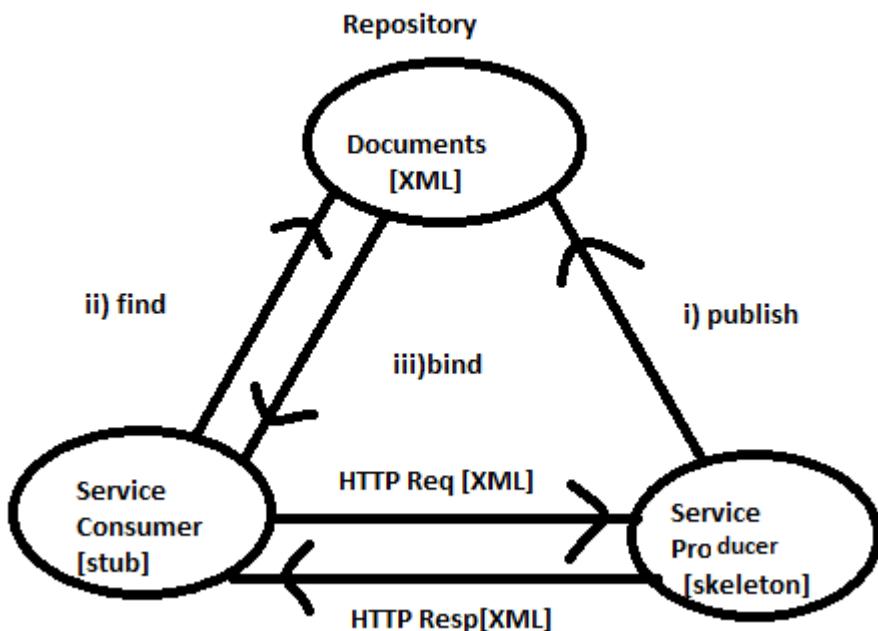
- 1) Service Producer.
- 2) Service Consumer.
- 3) Repository(Registry And Discovery).

It follows three operations in a chain:

- 1) Public Operation
- 2) Find Operation
- 3) Bind Operation

->Public and Find will be called one time as a set up and Bind will be called for every request-response execution.

SOA Design is::



→ SOA Execution steps:

Step#1: Define one Application which provides services, which is also called as Service Producer Application.

Here we write some login like classes, methods, ... to provide service. These are called Skeleton logic.

Step#2: Execute “publish” operation which converts skeleton logic to XML format, which is also called as Document.

All modifications of skeleton logic we must republish for new changes effected to Document.

Step#3: Repository is a memory location which runs in Cloud environment (internet).

It holds every Service with unique id. (Example: EMP-PRODUCER-8856).

Every Service contains details like: Service Name, Location of Service, Inputs and Outputs.

Example:

Service Name: EMP-SER-55  
Location of Service: <http://85.26.0.1:8956>  
Inputs: int, double  
Outputs: double

Step#4 Service Consumer executes “find” operation, that means:

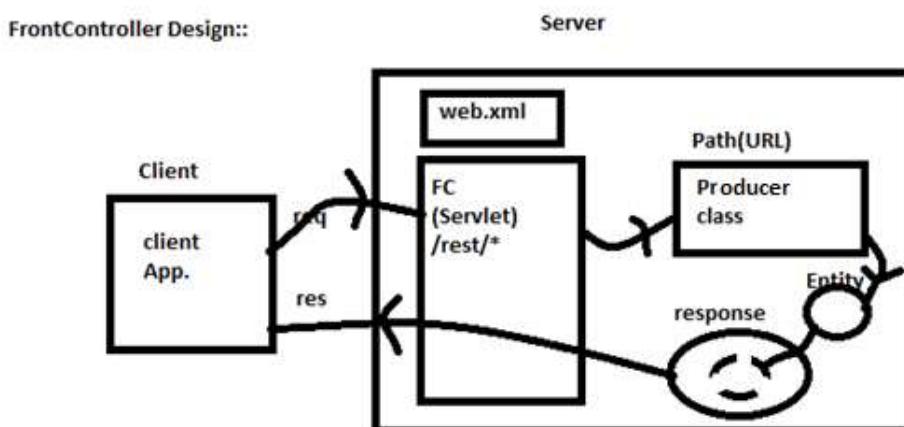
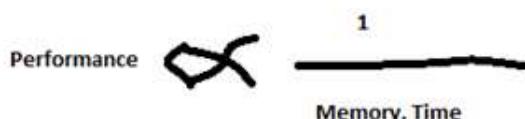
- Go to Repository
- Search for the Service, based on Service Id (name).
- Read Service.
- Generate “stubs”.

Here stubs are known as supporting code (classes in case of java) to make request to Producer Application.

Step#5: By using Stubs, define logic(class) which makes HTTP Request (XML) and gets HTTP Response (XML) back. It is called as “bind” operation.

## FrontController Design Pattern

->Design Patterns are used to reduce application memory and time (execution time), so that performance of application will be improved.



→ In the above design execution flow is given as:

- 1) Client machine makes request (browser, mobile, ATM, Swing Machine, etc...) which will be sent to FrontController.
- 2) In Server, FrontController behaves like Entry and Exist point. Every request made to application will be taken by FrontController.
- 3) FrontController is a Servlet (mostly predefined) which must be configured in web.xml file using Directory Match URL Pattern(Ex /rest/\*).

Note:

→ As per Advanced java, URL Patterns are three types:

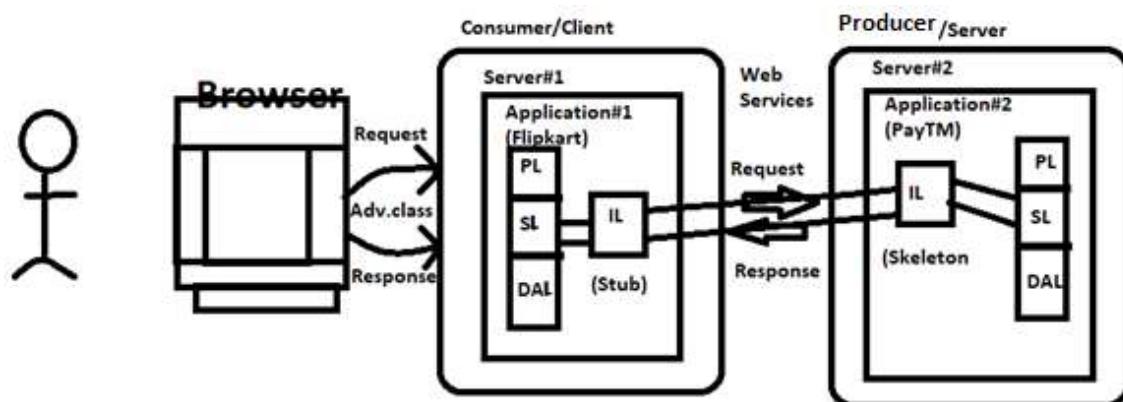
- 1) Exact match (ex. /ab)
- 2) Extension Match (ex. \*.htm, \*.ab, \*.do)
- 3) Directory match (ex. /ab/\*, /mvc/\*).
- 4) Based on URL (Path) FrontController will identify one Service Producer class. It will execute request and returns a final message, called as “Entity”.
- 5) Entity cannot be transferred over Network, so it will be wrapped (placed) into Response object.
- 6) Finally, FrontController will transfer response to Client Machine.

Notes:

- 1) If FrontController is not used then No. of operations =No. of Servlets.
- 2) Every Servlet will occupy more memory for it's super classes, HTTP Request/Response, Config, Context, Listener, Filters, etc...
- 3) FrontController makes  
No. of operations =No. of simple classes with methods.
- 4) FrontController is used by Structs Framework, Spring Framework, ReST Web Services, etc...

## Client-Server(CS) (or) Consumer-Producer (CP) Design Pattern:

- In case of Web Services, two applications will be linked together (also called as Integration)
- In this case one application running in Server #1 will be connected to another application running in Server #2. These two applications can be integrated using Extra Layered or Fourth Layer or Integration Layer (IL).
- This Integration layer will be connected to Service Layer in same application and Integrated layer other application. At a time one application behaves as Consumer or Client Application (which makes request) and another application behaves as producer or Server (which gives response).
- Design looks like:



### **Presentation Layer(PL)::**

->It contains view logic which is visible to end user. It can be implemented using Technologies like HTML, JSP, JSF, Angular, CSS, JavaScript, etc.. (UI Technologies).

### **Service Layer(SL):**

->It contains Business Logic and basic calculations. It is implemented using a loosely coupled Design Pattern “POJO-POJO”.

### **Data Access Layer(DAL):**

->It is used to perform Database operations like insert, update, delete and select.

It is implemented using JDBC, Hibernate, Spring jdbc, Spring ORM, Spring DATA JPA, etc.

### Integration Layer(IL):

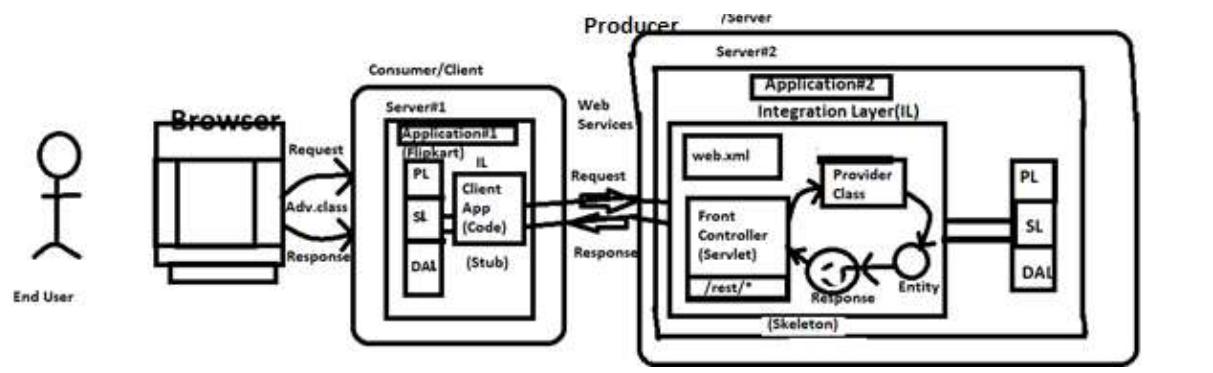
- > This layer is used to link our application with another application which is called as Integration Layer.
- > In this case two applications which want to integrate must have two Integration Layers.
- > One Integration Layer(IL) contains Producer code (known as Skeleton) and another Integration Layer(IL) contains Consumer code(known as Stub).
- > Stub will make HttpRequest to Skeleton.
- > Skeleton return HttpResponse to Stub.
- > These Integration Layers are implemented using Web Services.

## CP with FC Design

[CP=Consumer-Producer, FC=Front Controller]

- > It is a combination of two designs.
- > One is CP and another one is FC.
- > To integrate two different applications using Web Services with http Protocol (Request / Response), this design is used.
- > ReST Web Services implemented this Design.
- > Consumer Integration Layer(IL) should have Client Application code (class) which makes HttpRequest.
- > Producer Integration Layer(IL) is implemented using FrontController (FC) design which contains two types of files.
  - 1) web.xml
  - 2) Producer class.
- > In web.xml, we should configure one predefined Servlet with Directory Match URL Pattern (Ex. /rest/\*).
- > This Servlet behaves like entry and exit gate for Integration Layer(IL) of Producer.

- >It identifies one Producer class based on URL(Path).
- >Producer class will be linked with Service Layer(SL) of Application #2.
- >Finally Producer class return Entity which is placed into HttpResponse and given back to Client Application of Application #1.



## Assignment

# Text file to json format::

Step1: In pom.xml file

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
```

<!--

<https://mvnrepository.com/artifact/com.google.code.gson/gson -->>

```
<dependency>
 <groupId>com.google.code.gson</groupId>
 <artifactId>gson</artifactId>
```

```
<version>2.8.6</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

Step2: Create Model class::

```
package in.nit.model;
import lombok.Data;
@Data
public class Employee {
 private int eId;
 private String eName;
 private String eCountry;
 private String eCode;
 private String eDate;
}
```

Step3: Test class

g.txt::

```
1 Sankar IND 3gvh44fw2 342020-09-12
2 Raja us f34g4brtr 342020-09-12
```

```
package in.nit.test;
import java.io.File;
import java.util.Scanner;
import com.google.gson.Gson;
import in.nit.model.Employee;
public class ObjectToJsontest {
```

```
public static void main(String[] args) {
 try {
 //holding file
 File f=new File("g.txt");

 //reading file
 Scanner sc=new Scanner(f);

 //1. Model class Object
 Employee e=new Employee();

 //loop
 for(int i=0;i<2;i++) {
 int id=sc.nextInt();
 e.setEId(id);
 String name=sc.next();
 e.setEName(name);
 String eCountry=sc.next();
 e.setECountry(eCountry);
 String eCode=sc.next();
 e.setECode(eCode);
 String date=sc.next();
 e.setEDate(date);

 //2.Create Gson class Object
 Gson g=new Gson();
```

```
//3.Call toJson() method
String json=g.toJson(e);

//4.print Gson String
System.out.println(json);
//next line
sc.hasNextLine();

}

sc.close();
}

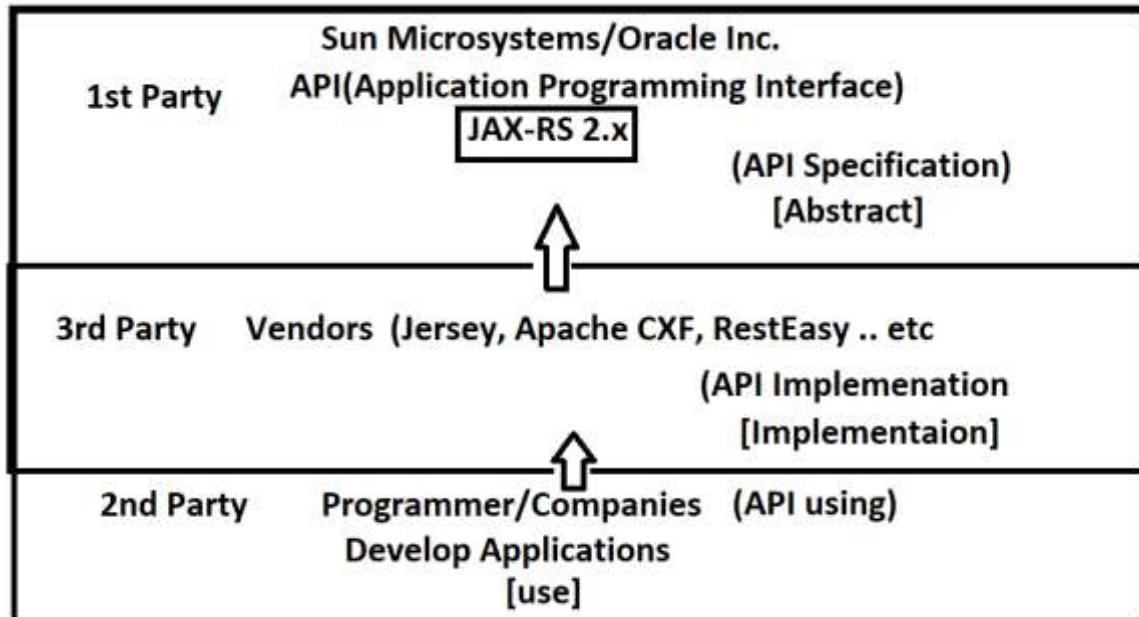
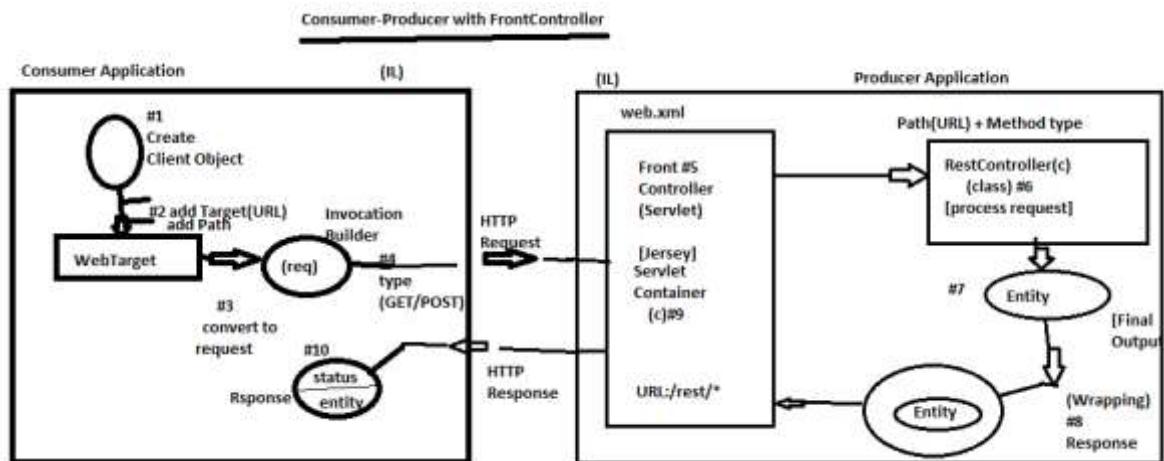
catch(Exception e) {
 e.printStackTrace();
}

}

Output ::

{"eld":1,"eName":"Sankar","eCountry":"IND","eCode":"3gvh44fw2","eDate":"342020-09-12"}
{"eld":2,"eName":"Raja","eCountry":"us","eCode":"f34g4brtr","eDate":"342020-09-12"}
```

---



| API contains |            |
|--------------|------------|
| packages     |            |
| class        | interface  |
| enum         | annotation |
| error        | exception  |

## Restful API: JAX-RS 2.x

### packages:

```
javax.ws.rs (main)
javax.ws.rs.client
javax.ws.rs.container
javax.ws.rs.core
javax.ws.rs.ext
```

- > (main) =To write producer app code
- >client =To write consumer app code
- >container =It creates object to our code and calls methods(logic)
- >core =common code for client and producer
- >ext =supports extension services like HK2(DI),  
MVC-JSP,Exports/Imports..etc.

# ReSTful-Webservices

Q) Webservices?

A) Linking two or more different applications which are running in different servers. (Also called as Integration).

Q) Types of Webservices?

A) Two Types:

- a. Legacy (SOAP=Simple Object Access Protocol)
- b. New (ReST)

Q) ReST Full form?

A) Re =Representation

S =State (Data in Global Format)

T =Transfer (send/receive using HTTP).

->It means send/receive data in Global format using HTTP protocol

Re = Representation ( it is conversion process object<->XML/JSON)

S = State (Data in Global Format)

T = Transfer (send/receive using HTTP)

→ \* ReST Architecture follow a Design CP with FC

-> XML OR JSON cannot be send over network directly. So, ReST uses HTTP (Hyper Text Transfer Protocol) to transfer data.

[ Consumer-Producer with FrontController\*\*]

---

→ API = Application Programming Interface

*(It is not a complete Implementation)*

→ It must be implemented by Programmer actually. But we are taking help of Vendors(3<sup>rd</sup> parties).

→ 3<sup>rd</sup> Parties are proving implementation for API may be in different code.

But programmer can use any implementation, Code will be same.

@) What is difference between API Specification and API implementation?

A) API Specification means rules and naming details like class name, method name, annotation name, input and output details (It is like abstract code) . API Implementation means rules with logic(implementation) [code is added]

---

## CP with FC Execution Flow:

->Consumer Application has to make request by following below steps:

Step1: Define one Client object

Step2: Add URI (server details of Producer) and Path (class and method details) which creates WebTarget Object.

Step3: Convert to request object given as Invocation.Builder

Step4: Provide method Type(GET/POST) which is equal to making Request.

⇒ Now Producer application is gets executes for Request

Step5: FC(Front Controller) is a pre-defined servlet given by vendor even programmer must configure in web.xml ( or even Java based config)

Example: Jersey Vendor has given ‘ServletContainer’ (it is a servlet).

Step6: FC will call one RestController class by using Path(URL) and Method Type.

Step7: After processing request it return Entity(Final Output).

Step8: It placed into Response(wrappering).

Step9: FC reads response and sends back to Consumer.

⇒ Now Consumer takes Response given by Producer application.

Step10: HTTP Response given by Producer is stored in one Type Object i.e Response(javax.ws.rs.core), which is mainly contains Status(HTTP Status) and Entity (Output).

---

## ReST Webservices (Java) – Producer Application Implementation

Two types of files required.

They are:

1. ReST Controller class( c ).
2. Front Controller Config File(web.xml/Java Config Style)

-----ReST Controller Syntax-----

```
package in.nit.____;
@Path("/<URL>")
public class _____{
 //Method Type Annotation
 @GET //POST, @PUT @DELETE
 public <ReturnType> <methodName>(<Params>){
 //logic;
```

}

}

→ One RestController is created as one class for one module.

Example: Employee Module =EmployeeController  
Payment Module =PaymentRestController  
Admin Module =AdminRestController

->\*\*\* RestController is also called as Resource class

---

-----Example-----

```
//ctrl+shift+o

@Path("/emp")

public class EmployeeRestController{

 @GET

 public String showMsg(){
 return "Welcome to Employee";
 }
}
```

Q1) Define one JAX-RS (RestFul) application by using details, like:

Module name: Admin having path /admin.

That contains a method: processMsg() which returns String type

Like 'Welcome to Admin'.

```
A) package in.nit.controller;
//ctrl+Shift+o

@Path("/admin")
public class AdminRestController{
 @GET
 public String processMsg(){
 return "Welcome to Admin";
 }
}
```

Q2) Define one JAX-RS (RestFul) application by using details, like:

Module name: Payment having path/pay.  
It contains method doProcess which return String type  
Like ‘Payment Done’.

A) Package in.nit.controller;

```
//ctrl+shift+o
@Path("/pay")
public class PaymentRestController{
 @GET
 Public String doProcess(){
 return "Payment Done";
 }
}
```

---

Q3) Define one JAX-RS (RestFul) application by using details, like:

Module name: Contract having path /cnt  
It contains method checkStatus which return String type  
Like ‘Under Process’.

Q4) Define one JAX-RS (ReSTFul) application by using details, like:

Module name: Product having path /prod.  
It contains method addProduct which return String type  
Like ‘Product Added’.

Q5) Define one JAX-RS (ReSTFul) application by using details, like:

Module name: User having path /user.  
It contains method checkData which return String type  
Like ‘User Validated’.

---

1. @Path: (javax.ws.rs package)

This annotation is used to provide unique Path(URL).

So, that client can access one class and method.

→ Path behaves like URL.

→ In Adv Java, Servlet <url-pattern> similar to Path

- Path must be provided at class Level.
- \*\* Path can also be provided at Method Level(optional).
- Path is case-sensitive.  
i.e : /pay, /PAY, /Pay are different.  
-> Two classes should never contain same path which is InValid if contains.

Example: @Path("/one")

```
class Simple{
}

@Path("/one")

class Process{//InValid
}
```

---

## 2. HTTP Method Type Annotations (javax.ws.rs. page).

@GET, @POST, @PUT, @DELETE

- These annotation can be provided only at method Level.
- Every Java Method which is inside RestController must have any one HTTP Method Type Annotation.
- One Java Method should never contain zero or more then one HTTP Method Type Annotation.

Example:

```
@Path("/emp")
public class EmpRestController{
 @POST
 String saveEmp(){.....}
 @DELETE
 String removeEmp(){....}
 @GET
 String fetchEmp(){.....}
 @PUT
 String updateEmp(){.....}
}
```

---

- FC can be configured using web.xml, because it is a Servlet defined by Vendors. One vendor to another Vendor FC gets changed. RestController code remains same.
- Jersey Vendor has provided one FC (pre-defined Servlet)  
Named as: ServletContainer.
- This FC(Servlet)
  - a) Detect the classes (RestController) based on one package Name
  - b) Once classes are found, then objects are created.
  - c) Methods are also called when client request is Came.
  - d) Return Response back to Client.

→ Create Maven WebApp::

In pom.xml File

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>

 <dependency>
 <groupId>junit</groupId>
```

```
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
</dependencies>
```

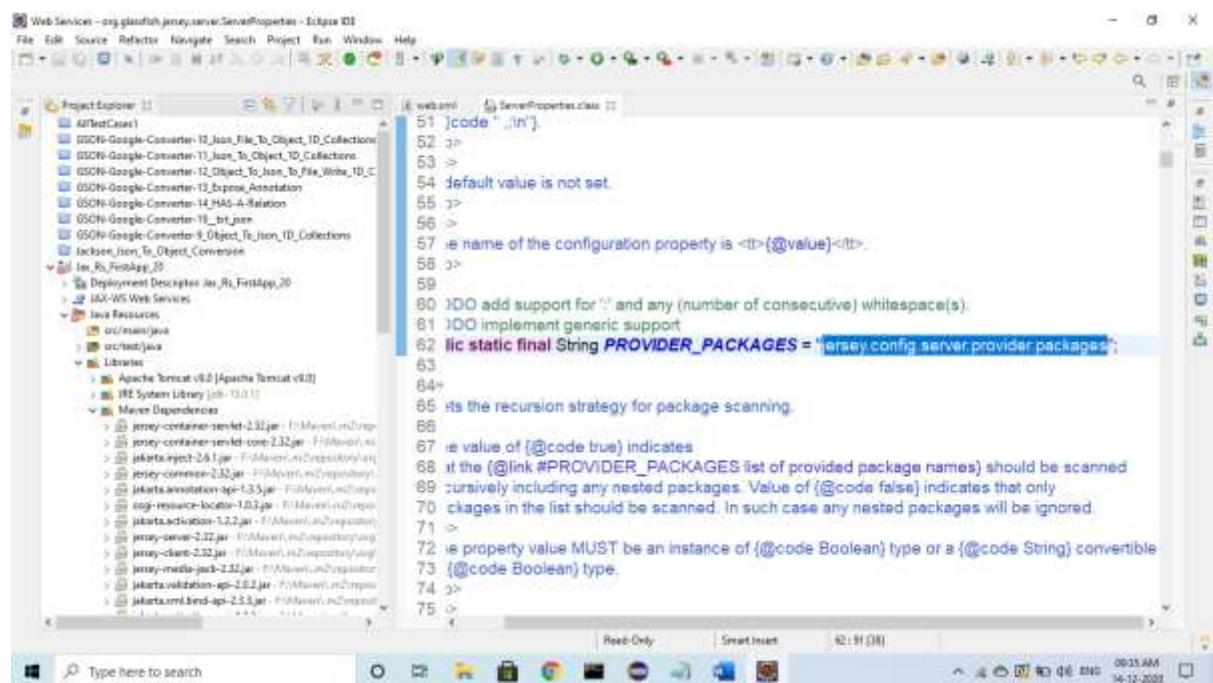
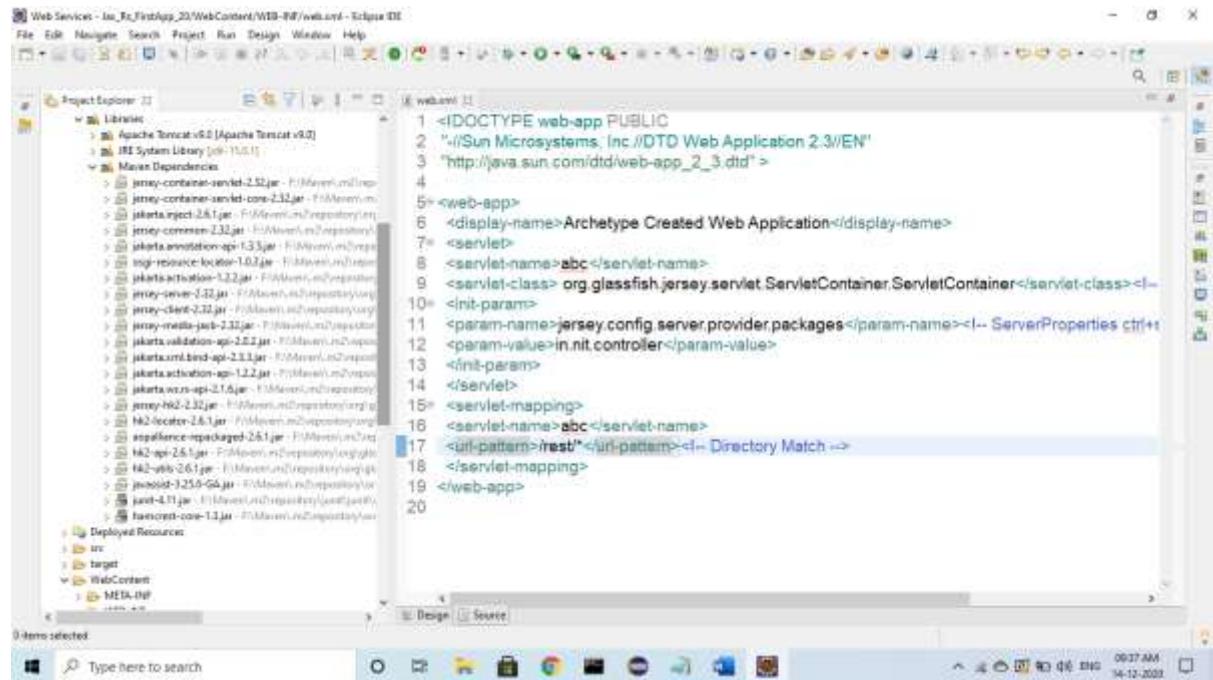
-----web.xml-----

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>
org.glassfish.jersey.servlet.ServletContainer.HttpServletContainer</servlet-
class><!-- ServletContainer ctrl+shift+T -->
<init-param>
<param-name>jersey.config.server.provider.packages</param-
name><!-- ServerProperties ctrl+shift+t -->
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/rest/*</url-pattern><!-- Directory Match -->
</servlet-mapping>
</web-app>
```

---

Dependencies(Jars):



→ Key : **ctrl+shift+t** Open Type(To open any predefined class/interface).

- a.**ctrl+shift+t** => Enter ‘ServletContainer’ => choose first option.
- b.**ctrl+shift+t** => Enter ‘ServerProperties’ => choose first option.

Scroll down to copy package name key  
 (jersey.config.server.provider.packages)

Note::

Full name for FC, given by Jersey Vendor is:  
 org.glassfish.jersey.servlet.ServletContainer.

- ⇒ Work done by FC-detect the RestController-create object-all its methods  
On request-send response back to client.

Q1) How FC will detect the RestControllers?

- A) By using one package name as <init-param> value.  
But <param-name> must be:  
Jersey.config.server.provider.packages
- 

## Jersey-Webservices/ReSTful/Producer App

### -----First Application-----

Software Setup::

- JDK15 : <https://jdk.java.net/15/>
  - Tomcat 9.0 : <https://tomcat.apache.org/download-90.cgi>
  - Eclipse IDE 2020-09 :  
[https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2020-09/R/eclipse-jee-2020-09-R-win32-x86\\_64.zip&mirror\\_id=105](https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2020-09/R/eclipse-jee-2020-09-R-win32-x86_64.zip&mirror_id=105)
- 

### -----PROJECT SETUP-----

Step#1 JDK15 setup in Eclipse 2020-09 (only one time, for one workspace)

>Windows>Preferences>Search with 'installed JRE'>choose same

>Click on Add>Standard VM>Browser(Directory) for location  
Ex: F:\Program Files\Java\JDK15.

>Finish >Apply and Close.

Step2: Tomcat9 setup in Eclipsev2020-09(only one time for one workspace)

>Windows>Preferences>Search withn'Server'

>Choose Runtime Environment >click on Add button

>Select server home(Ex: Apache Tomcat9)>next  
>Browser for location  
Ex: F:\Program Files\Apache Software Foundation\Tomcat9  
>Next>Finish

Step#3 Maven Web Project Setup(one time for one application)

a) Create Maven Web Project

>File>new>maven project>next  
> Search with 'maven-archetype-webapp' > choose same  
> next> enter details (example below)

GroupId : in.nit

ArtifactId: JerseyProducerFirstApp

Version : 1.0

b) Update JRE and Server using Build path

➤ Right click>build path>Configure Build path>click on Library tab  
➤ Choose JRE System Library Operation>Click on Edit  
➤ Select workspace Default JRE option>Apply  
⇒ Click on Add Library >Server Runtime>Choose Apache Tomcat>Finish  
⇒ Apply and Close.

Step#4 Provide Dependencies and plugins

➤ Open pom.xml  
➤ Add below content in pom.xml

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
```

```
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->
 <!-- Jersey Container Servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!-- Jersey HK2 -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
 </dependencies>
```

→ Then press **ctrl+A,ctrl+I=>ctrl+S** in pom.xml

## Step#5

Update Maven Project(Alt+F5)

- Right click on Project > Maven > Update Project
- Choose Checkbox [v]Force Update
- Finish

-----Coding-----

1.web.xml (Front Controller config code)

2. RestController class

#1 Remove index.jsp under location:: /src/main/webapp/WEB-INF

## #2 Open web.xml and Configure FC

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/* </url-pattern>
 </servlet-mapping>
</web-app>
```

Step3 Define one RestController class

>right click on src/main/java>new >class

>Enter package and class name.

Example: package :in.nit.controller

Name: EmployeeRestController

➤ Finish

-----Code-----

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

//ctrl+shift+o for imports
@Path("/emp")
public class EmployeeRestController {
 @GET
 public String showMsg() {
 return "Welcome to RestWebServices";
 }
}
```

}

## Step#4 Run Application

>Right click on Project >Run on server>Next>Finish

>Enter URL in browser

Protocol: //IP:PORT/ProjectName/WebXmlFcUrl/classPath

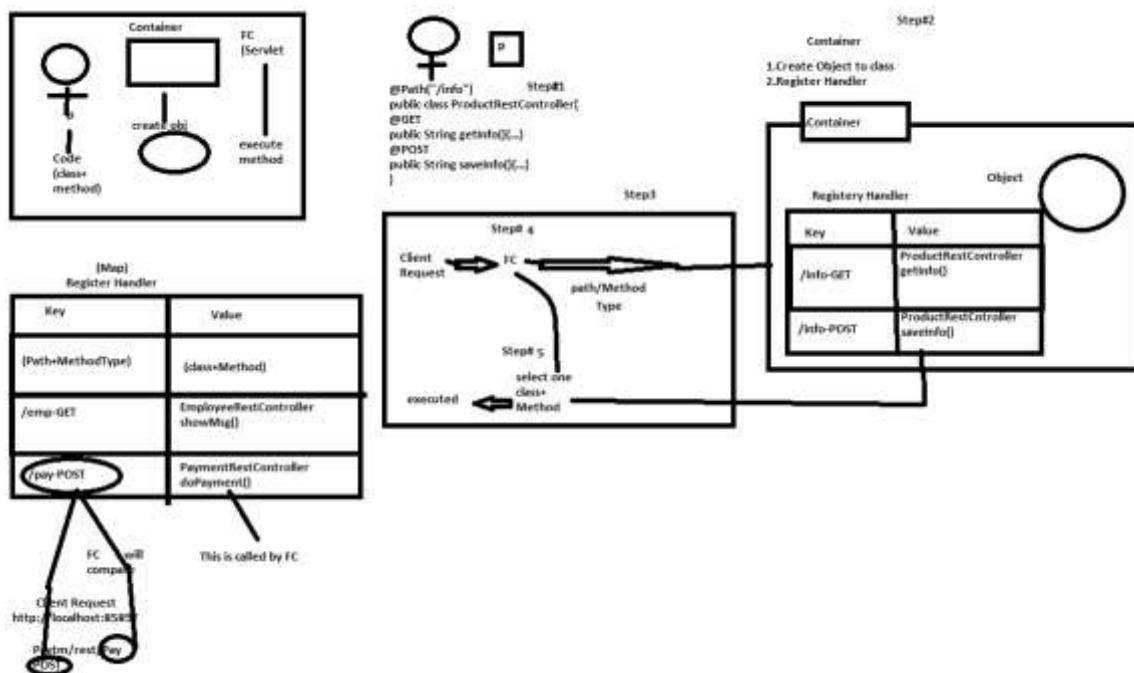
<http://localhost:3031/JerseyProducerFirstApp-21/rest/emp>

url-patterns:

---

<!--<url-pattern>/\*</url-pattern> All -->  
 <url-pattern>/rest/\*</url-pattern><!-- Directory Match -->

---



## Work flow-Stage-1: RestController

→ For one Module, we define one RestController

Example: Admin Module=AdminRestController

Payment Module =PaymentRestController

User Module=UserRestController

- It is a simple(normal class) but annotated with JAVAX-RS API(javax.ws.rs).
  - Code (RestController class and Methods) written by programmer only.
  - Object creation done by Container(WebServices container) named as
  - FC will call method when request came to it.  
(10-request -10 time method is called).
  - Registry Handler is created by container and handled by FC.  
Register Handler looks like a Map  
(Key-Path+HttpMethodType,value-RestController+Method)
- 

### \*\*\*\*\*Work Flow -Stage 1 Execution\*\*\*\*\*

- a) Client is making Request
  - b) Request given to FC
  - c) FC will read PATH and HTTP method type from request
  - d) FC will go to Registry handler
  - e) Compare Path and HTTP Method Type
  - f) Read Matching class and Method
  - g) Select those class and method
  - h) Execute method
  - i) Get Result
- 

@) Can we define multiple RestControllers in RestApp?  
A) Yes.

Step1: in pom.xml file::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
```

```
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->
<!-- Jersey Container Servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
</dependency>
<!-- Jersey HK2 -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

</dependencies>
```

## Step2:web.xml file

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
 <display-name>JerseyProducerSecondApp</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value><!-- in.nit for detect all subpackages
in.nit.controller for detect only controller package -->
 </init-param>
 </servlet>
 <servlet-mapping>
```

```
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### Step3: Rest Controllers

```
package in.nit.controller;
```

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;

//ctrl+shift+o for imports
@Path("/emp")
public class EmployeeRestController {
 @GET
 public String showMsg() {
 return "Welcome to RestWebServices";
 }
}

package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/pay")
public class PaymentRestController {
 @GET
 public String showMsg() {
 return "payment Done";
 }
}
```

- Right click on Project>Run As>Run on Server>Enter URL in browser

➤ URL:

Protocol://IP:PORT/ProjectName/FCWebXmlURL/classPath  
<http://localhost:3031/JerseyProducerFirstApp-21/rest/emp>  
<http://localhost:3031/JerseyProducerFirstApp-21/rest/pay>

Q) What is the work done by Programmer, Container and FC?

A) Programmer -Writing code

Container -Creating Object (to RestController)

FC -Execute Method when client request came.

---

Every Webservice must provide 3 concepts for one Application.

- 1) Implementation (Coding).
- 2) Security (username/password).
- 3) Restriction(Role).

Jersey is a API, only for JAX-RS, provided in two versions.

- 1) Jersey 1.x (web.xml)
  - 2) Jersey 2.x(Annotation and Java Config based).
- 

## Work flow-Stage-2 : (RestController)

->HTTP Methods:

GET : Fetch Resource from Producer Application(server)

POST : Create new Resource at Producer Application(server)

PUT : Modify Resource at Producer Application(server)

DELETE: Remove Resource at Producer Application (server)

➔ Resource means any type of Data.

Example: Text, Image, Audio, Pdf, Database, XML, JSON..etc.

➔ To handle above Operations, ReSTFul(JAX-RS) API

@GET, @POST, @PUT, @DELETE

Define in package: javax.ws.rs

➔ Every (Java) Method in RestController class

Must be connected to any one Http Method Type.

Example Code::

<properties>

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <!-- Jersey Container Servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!-- Jersey HK2 -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
 </dependencies>
```

In web.xml file::

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
 <display-name>JerseyProducerSecondApp</display-name>
 <servlet>
```

```
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit</param-value><!-- in.nit for detect all subpackages
in.nit.controller for detect only controller package -->
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

RestController::

```
package in.nit.controller;

import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;

//ctrl+shift+o for imports
@Path("/emp")
public class EmployeeRestController {
 @GET
 public String showMsg() {
 return "Fetched Data...";
 }
 @POST
 public String showMsg1() {
 return "Saved Data...";
 }
 @PUT
 public String showMsg2() {
 return "Modified Data...";
 }
 @DELETE
```

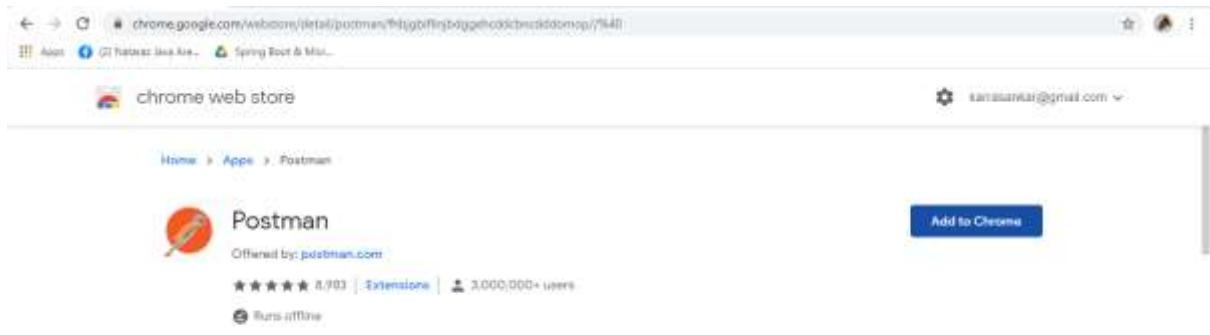
```
public String showMsg3() {
 return "Removed Saved Data...";
}
}
```

---

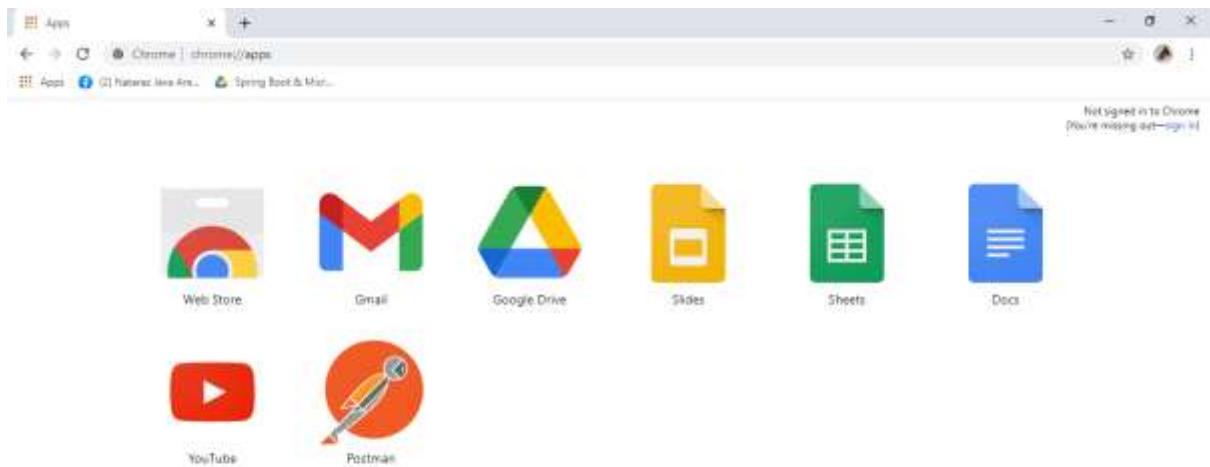
# POSTMAN TOOL

- To test/access above producer application, use
  - a) HTTP Test Tool(POSTMAN TOOL)
  - b) Program(Client App/Consumer App, Junit).

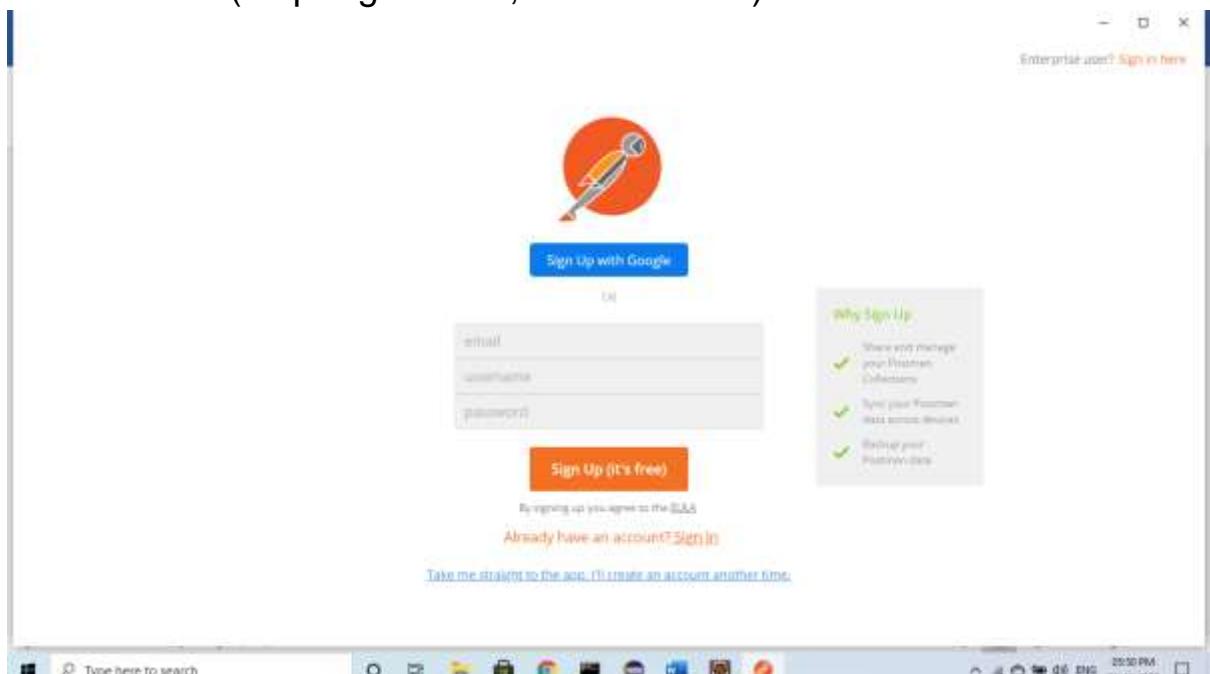
- Adding POSTMAN Tool to Chrome browser
- Open Google Chrome Browser( or any other browser)
- Enter 'POSTMAN TOOL CHROME EXTENSION'.
- Choose First Link
- Click on 'Add to Chrome'(Blue color button)
- Select 'Add App'(wait for few minutes)
- Close browser and open again



For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>



- ⇒ Now, to do below steps to run postman
- Client on Apps Symbol or enter chrome ://apps/in browser
- Client on POSTMAN Symbol
- \*\* For First time (Skip registration, at bottom link)



- Click Take me Straight to App  
→ Enter details and click on SEND.

|             |                                                                                                                                                           |        |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| [DROP DOWN] | [.....URL.....]                                                                                                                                           | [SEND] |
| GET         | <a href="http://localhost:3030/JerseyProducerThirdApp_HTTP_Methods-23/rest/emp">http://localhost:3030/JerseyProducerThirdApp_HTTP_Methods-23/rest/emp</a> | SEND   |

Note:

- URL is case-sensitive  
/emp, /eMp, /Emp, /EMP are different.
- If Request URL/Path not exist at Producer (or spell/case is wrong)  
Then FC will throw Error:HTTP STATUS 404- Not Found

Producer app URL/Path: /emp

Request URL/Path" /EMP, /EmP, /emps (Invalid).

- If request HTTP Method Type (Example: POST) is not exist at Producer  
Then FC will throw Error: HTTP STATUS: 405-Method Not allowed.  
Example: Producer is having - @GET and @DELETE  
Request is : @POST (which is not exist at Producer)
- If request is Successfully processed then FC will return :HTTP STATUS: 200 OK

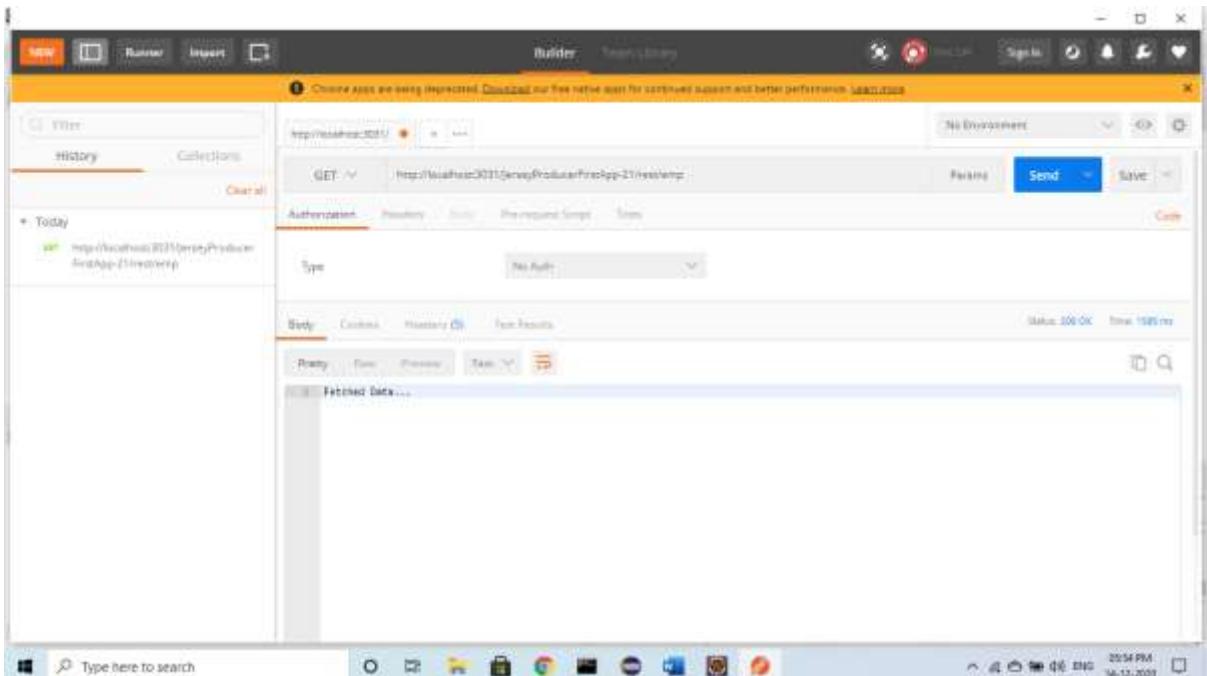
Run App Process::

Step1: run as Server

Step2: copy url in chrome browser (or)default eclipse browser url

Step3: open PostMan Tool

Step4: Select mode like GET, POST and Paste url then send see results::



## ReSTful -Jersey Vendor-Consumer Application FirstApp

- Consumer Application makes HTTP Request to Producer Application.
- Producer App will process request and return response back to Consumer.

Example: BookMyShow -----linked with -----PayTm

Here : PayTM is Producer and  
BookMyShow is Consumer.

Consumer Coding Steps:

- Define one Client Object
- Add Target(URI), Path to Client Object

URI=Protocol://IP:PORT/ProjectName

Example: URI=http://localhost:3030/EmpProducerApp

That return WebTarget

(WebTarget=location of RestWebservices Provider)

- Convert WebTarget to Request format which return  
Innocation.Builder
- Provider HTTP by Method Type like get(),post()..etc  
That returns Response
- Response Given by FC contains mainly Status and Entity (Final  
Output).

-----Coding Steps-----

- Consider one Producer application is already defined.

-----Producer Code-----

(Note Webapp Project)

1.pom.xml

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <!-- Jersey Container Servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!-- Jersey HK2 -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
</dependencies>
```

2: In web.xml file::

```
<web-app>
 <display-name>JerseyProducerSecondApp</display-name>
 <servlet>
```

```
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit</param-value><!-- in.nit for detect all subpackages
in.nit.controller for detect only controller package -->
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### 3. RestController

```
package in.nit.controller;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

//ctrl+shift+o for imports
@Path("/producer")
public class ProducerRestController {
 @GET
 public String showMsg() {
 return "From Producer Service..";
 }
}
```

-----Consumer Coding Steps-----  
(Note:: Simple Project Not WebApp Project)

Step#1 Create Simple Maven Project

>File>new>Maven Project>click checkbox [v] create Simple Project

>Next>Enter details>Click Finish

Example: GroupId: in.nit

ArtifactId: Jersey2FirstConsumer

Version : 1.0

## Step#2 pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <!-- Jersey Container Servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!-- Jersey HK2 -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
</dependencies>
```

## Step#3 Consumer Code:

```
package in.nit.test;
```

```
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTest {
 public static void main(String[] args) {
 String uri="http://localhost:3031/JerseyProducerFirstApp-21/";
 String path="/rest/producer";
 try {
 //1. Create Client Object
 Client c=ClientBuilder.newClient();

 //2.Define WebTarget by adding Target(URI) and Path
 WebTarget wt=c.target(uri).path(path);
 //3.Convert into Request Object
 //(Equals to HTTP request under Construction)
 Invocation.Builder builder=wt.request();

 //4.Provide HTTP Method Type
 Response resp=builder.get();

 //5. Read and Print Response Details
 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 }
 }
}
```

```
 System.out.println(resp.readEntity(String.class));
 } catch (Exception e) {
 e.printStackTrace();
 }
}
}
```

---

-----Execution Order-----

1. Run Producer Application
    - Right click on Producer> Run AS >Run on Server
  2. Run Consumer Application
    - Right click on Consumer Test class> Run As > Java Application
- Example Output::

200  
OK  
From Producer Service..

---

Q) What is Target (URI)?

A) It is a location of Producer Application

Q) What is Path?

A) It is URL pattern given to FC and Class, method

Q) What are different HTTP Methods Types?

A) get(), post(), put() and delete()..etc.

Q) What is Response (javax.ws.rs.core)?

A) Response is class that holds data of HTTP Reponse given by Producer application.

Q) What is status?

A) It is a number/code given by Server/Producer to indicate current status of Request.

200 Success, 404 URL is wrong, 500 Exceptions

Q) What is Entity

A) It is a Final Response given by Producer RestController.

Q) If Producer is not started! Then if we run Consumer App, what will happen?

A) It throw Exception: java.net.ConnectException: Connection refused:connect.

---

## Rules To Write RestController : Path and Method Type

---

1. Every RestController must have Path at Class Level  
It must be unique.  
Path (URL) is case-sensitive.

```
@Path("/emp")
class EmpRest{} //----Valid
```

---

```
@Path("/adm")
class AdmRest{} //----Valid
```

---

```
class PrtRest{} // (Invalid, it has no @Path)
```

---

```
@Path("/emp")
class TestRest{}//---(InValid, it is already exist at EmpRest class)
```

---

```
@Path("/Emp")
class OneRest{}//----Valid
```

2. Every Java Method must have one HTTP Method only.  
Path at Method level is optional for only one Method Type one Time.

```
@Path("/emp")
class EmpRest{//---Valid
@GET
void show(){
}
}
```

---

```
@Path("/emp")
class EmpRest{ //---Invalid. Two methods found with same
//Type(GET) without Path.
@GET
void show(){}
@GET
void find()// ** At Max we can define one Java Method without
//Path with one HTTP Method Type.
}
}
```

---

```
@Path("/emp")
class EmpRest{//----Valid
@GET
void show(){}
@GET
@Path("/find")
void find(){}
}
```

---

```
@Path("/emp")
class EmpRest{//--Valid
@GET
@Path("/show")
void show{}
@GET
@Path("/find")
void find(){}
}
```

```
@Path("/emp")
class EmpRest{ //-- Valid
@GET
void show(){}
@POST
void find(){}
}
```

---

```
@Path("/emp")
class EmpRest{--Valid
@GET
void show(){}
@POST
void save(){}
@put
void update(){}
@DELETE
void remove(){}
}
```

---

```
@Path("/emp")
class EmpRest{--Valid. One Method can exist without path for @GET
@GET
void showA(){}
@GET
@Path("/b")
void showB(){}
@GET
@Path("/c")
void showC(){}
@GET
@Path("/d")
void showD(){}
}
```

---

```
@Path("/emp")
class EmpRest{//--Invalid. One Java Method only one MethodType
@GET
@POST
void show(){}
}
```

---

### -----Producer Code-----

(It is a WEBAPP)

Note: Maven WebApp packages not come by default ,means when you are creating webapp in that time enter GroupId, ArtifactId, package name not come to project.

→ In pom.xml file::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
<!--container Servlet-->
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
<! --h2-- >
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>

</dependencies>
```

Step2:EmployeeRestController::

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

@Path("/emp")
public class EmployeeRestController {
 @GET
 public String showA() {
 return "GET Employee ShowA";
 }
 @GET
 @Path("/b")
 public String showB() {
 return "GET Employee Showb";
 }
 @POST
 public String showC() {
 return "POST Employee showC";
 }
 @POST
 @Path("/d")
 public String showD() {
 return "POST Employee showD";
 }
}
```

### Step3:ProductRestController

```
package in.nit.controller;

import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;

@Path("/prod")
public class ProductRestController {
```

```
 @GET
 public String showA() {
 return "GET Employee ShowA";
 }
 @POST
 public String showB() {
 return "POST Employee Showb";
 }
 @PUT
 public String showC() {
 return "POST Employee showC";
 }
 @DELETE
 public String showD() {
 return "DELETE Employee showD";
 }
}
```

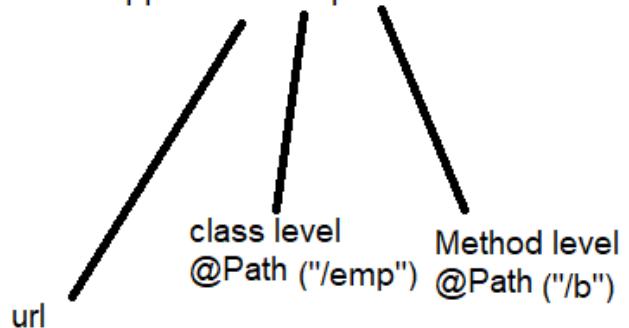
→ In web.xml file::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
```

```
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

Note:: in Chrome right click on postman icon create desktop icon

<http://localhost:3031/EmployeeProducerApp-25/rest/emp/b>



-----Consumer App-----

Note:: it is Simple Maven Project

Step1:: in Pom.xml file::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

</dependencies>
```

## Step2:: consumerTestApp::

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTest {

 public static void main(String[] args) {
 String uri="http://localhost:3031/EmployeeProducerApp-25";
 String path="/rest/prod";
 //String path="/rest/emp";
 //String path="/rest/emp/b";
 //String path="/rest/emp/d";
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(uri).path(path);

 Invocation.Builder builder=wt.request();
```

```
Response resp=builder.get();

System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));
} catch (Exception e) {
 e.printStackTrace();
}

}
```

---

Execution: step1: Producer App it is webapp ..note without stop server you can run consumer App.

Step2: Consumer App

Output::

```
String path="/rest/prod";
Response resp=builder.get();
```

200  
OK  
GET Employee ShowA

Output1:

```
String path="/rest/emp/d";
Response resp=builder.post(Entity.text("NONE"));
```

200  
OK  
POST Employee showD

Output::

```
String path="/rest/prod";
Response resp=builder.put(Entity.text("NONE"));
```

200

OK

POST Employee showC

Output::

```
String path="/rest/prod";
Response resp=builder.delete();
```

200

OK

DELETE Employee showD

---

Full Consumer Test Code::

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTest {

 public static void main(String[] args) {
 String uri="http://localhost:3031/EmployeeProducerApp-25";
 String path="/rest/prod";
 //String path="/rest/emp";
 //String path="/rest/emp/b";
 //String path="/rest/emp/d";
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(uri).path(path);

 Invocation.Builder builder=wt.request();

 //Response resp=builder.get();

 //Response resp=builder.post(Entity.text("NONE"));

 //Response resp=builder.put(Entity.text("NONE"));
 }
 }
}
```

```
Response resp=builder.delete();

System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));
} catch (Exception e) {
 e.printStackTrace();
}

}

}
```

---

## Parameters to Producer Application

- Parameter means input given to task/code/logic.
- In Rest Producer Application it supports reading inputs (Parameters) given by Consumer Application.

### Types Of Parameters::

1. Query Parameters (URL?key&key=value)----- ? &
  2. Matrix Parameters (URL;key=val;key=val)----- ;
  3. Header Parameters (HTTP Header Section)-----key=value
  4. Form Parameters (HTML Form/Form Object)-----action="\_\_"
  5. Path Parameters/Path Variable/Template  
(URL/val/val/val.....)----http://.....5/A
  6. Cookie Parameters (HTTP Cookies)
  7. Bean Parameters (Parameters converted to Object).
  8. @DefaultValue----- used for default values :---10,Sankar,A,23.33
  9. @FormDataParam----- used for file uploading
- 

1. Query Parameters: (URL?key=val&key=val)

This concept is supported by web and web related (web services) applications it is used to send data from Consumer (client) app to Producer(Server).

- It is used to send data in key=value format.

- Both key and values are String type.
  - We can send multiple keys in any order.  
By using Symbols: ?, &
  - To read this data in RestWebservices Syntax is:  
    @QueryParam("key")Datatype LocalVariable
  - These are used to Send data[ from Consumer to Producer application only] in key=value Format using Request URL.
  - Note: Data will be send in key value format, both are String type by default.
- \*\* Above Syntax Supports reading and parsing data.

Example#1 URL?sname=ABC (HTTP Request)

Reading Data:

```
@QueryParam("sname")String sn
```

-----Equal Meaning in Advanced Java-----  

```
String sn=request.getParameter("sname");
```

---

Example2:

URL?sid=95 (HTTP Request)  
    @QueryParam("sid")int id

-----Equal Meaning in Advanced Java-----  

```
String sid=request.getParameter("sid");
Int id=Integer.parseInt(sid);
```

---

Example#3: URL?sfee=9.88 (HTTP Request)

Reading data:

```
 @QueryParam("sfee")double fee
```

-----Equal Advanced Java Meaning-----  

```
String sfee=request.getParameter("sfee");
double fee=Double.parseDouble(sfee);
```

---

- If Key is not present in Request URL then FC provides default values.  
Like int-0, String=null, double=0.0 ... etc.
- In case of multiple keys, order is not required to be followed to send data using URL.

Example:

```
..../emp?esal=3.3&eid=12&ename=A
..../emp?eid=10&ename=A&esal=3.3
..../emp?eid=19&esal=2.4&ename=A
```

(all are same)

- \*\* If same key is provided multiple times in request with different values, then first combination in order is taken into App by FC.

Example: .../emp?eid=5&eid=8&eid=7

Then selected: eid=5

- \*\* If DataType is mismatched with Request URL and Producer code then FC throws HTTP STATUS: 404 NOT FOUND

Example: ../emp/?eid=9A

Then Status is : 404 NOT FOUND

-> It Supports Data Type conversion from String to int, double, Boolean, char.....

-> Producer can read multiple key, value pairs at a time without following order in requestURL. Data binding is done based on key, not on order.

-> If request has(key,value) pair which is not present in Producer Application is called as Extra pair. These are ignored by FC( No Error, No Exception).

---

- \*\* To send data from Consumer Application use method

wt.queryParam(key,value);

-----Code-----

-----Producer Code-----

- It is a web app

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
```

[https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->](https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet-->)

```
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

</dependencies>
```

In web.xml file::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

EmployeeRestControllerParameter::

```
package in.nit.controller;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
```

```
@Path("/emp")
public class ProductRestControllerParameters {
 //.../emp?eid=9&ename=A&esal=3.4

 @GET
 public String readInputs(@QueryParam("eid")int eid,
 @QueryParam("ename")String
ename,
 @QueryParam("esal")double
esal
) {

 return " Input Data Is: ID "+eid+", NAME= "+ename+", ,
ESAL= "+esal;
 }

}
```

---

Output: Producer

Default Values::

<http://localhost:3031/EmployeeProducerApp-25/rest/emp>

Input Data Is: ID +0, NAME= null , ESAL= 0.0

Give Values Like:

emp?eid=101&ename=Sankar&esal=45.5

<http://localhost:3031/EmployeeProducerApp-25/rest/emp?eid=101&ename=SANKAR&esal=34.5>

Input Data Is: ID +101, NAME= SANKAR , ESAL= 34.5

---

-----Consumer Code-----

In pom.xml file::

<properties>

```
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>

</dependencies>
```

### ConsumerTest::

```
package in.nit.test;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTest {
```

```
public static void main(String[] args) {
 String uri="http://localhost:3031/EmployeeProducerApp-25";
 String path="/rest/emp";
 try {

 //1. Create Client Object
 Client c=ClientBuilder.newClient();

 //2. Add target and path
 WebTarget wt=c.target(uri).path(path);

 //3. Add parameters Here to Request
 //queyParam(key,value)
 wt=wt.queryParam("eid",101);
 wt=wt.queryParam("ename","Sankar");
 wt=wt.queryParam("esal",45.67);

 //4. Convert to Request
 Invocation.Builder builder=wt.request();

 //5. Provide method type
 Response resp=builder.get();

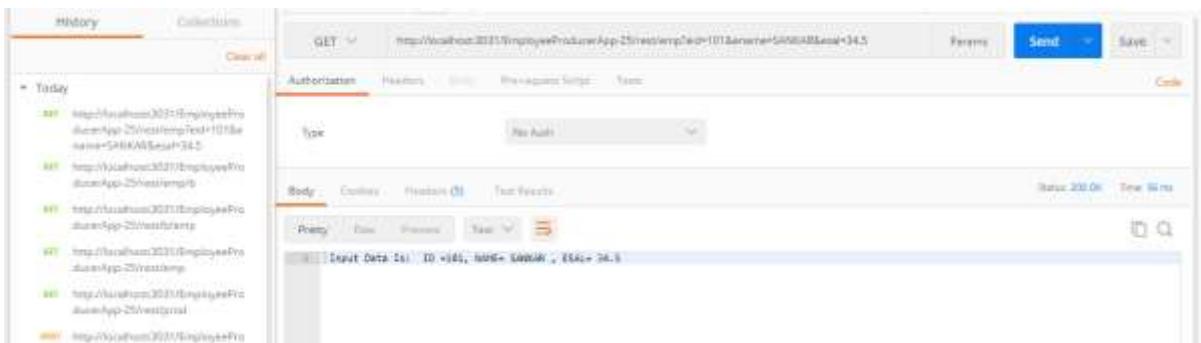
 //6. Print Entity and Status
 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 System.out.println(resp.readEntity(String.class));
 } catch (Exception e) {
 e.printStackTrace();
 }
}

}
```

---

-----POSTMAN SCREEN-----

GET <http://localhost:3031/EmployeeProducerApp-25/rest/emp?eid=101&ename=SANKAR&esal=34.5> SEND



Consumerclass Output:: it is Simple Maven Application

200

OK

Input Data Is: ID +101, NAME= Sankar , ESAL= 45.67

---

### Assignment(Token):

Define one Application using JAX-RS Producer code.

Module: GstService that is having path: /gst

Operation name: calcuateAmt()

Parameters: baseCost and gstPercentage

Return String Final Cost is: (baseCost+gstAmt)

Gst=Amt=baseCost\*gstPercentage/100.0

Hint: Use Query Param.

-----Assignment Producer Code-----

Note::it is WebApp

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

</dependencies>
```

In web.xml file::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

### GSTProducerController:

```
package in.nit.controller;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
@Path("/gst")
public class ProducerGSTParameters {
 //.../gst?bcost=5.6&gper=2

 String message=null;
 @GET
 public String readInputs(@QueryParam("bcost")double amt,
 @QueryParam("gper")int per
) {

 if(amt>0.0&&per>0) {
 double gstAmount=amt*per/100.0;
 double finalAmt=amt+gstAmount;

 message="Final Cost is: "+finalAmt;

 }
 else {
 message=" Please Enter Base Cost(bcost) and
GSTPercentage(gper)";
 }

 return message;
 }

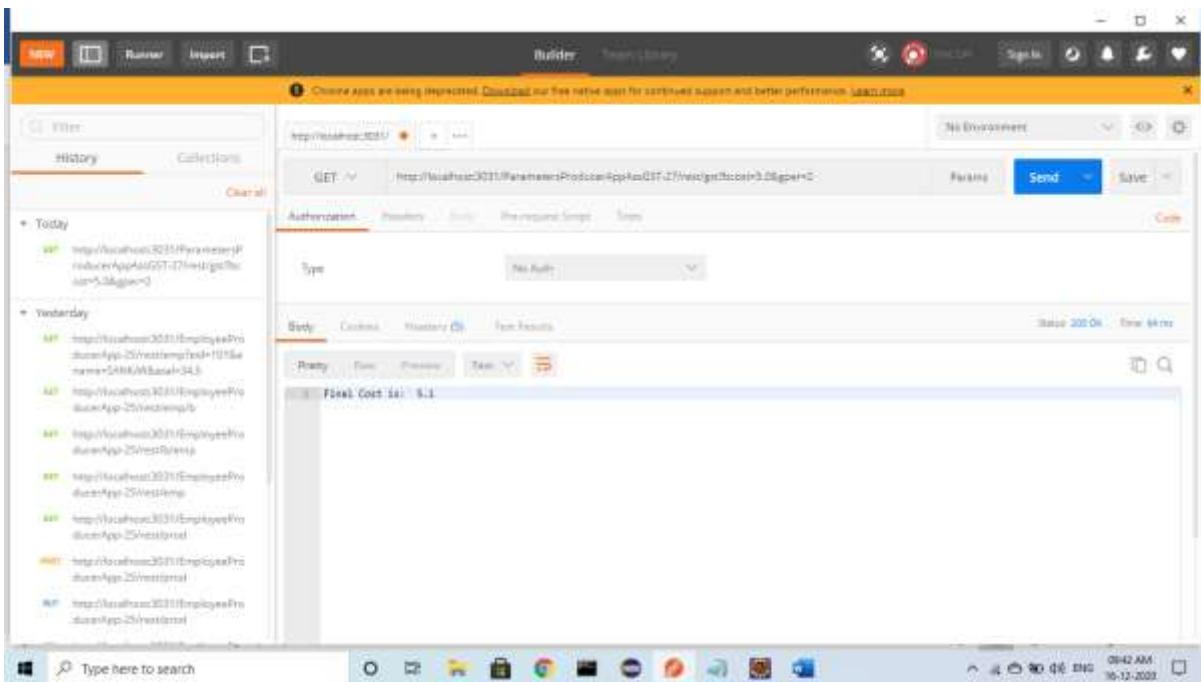
}
```

Note: if webapp is copy paste app then change app name in Web Project Settings  
<http://localhost:3031/ParametersProducerAppAssGST-27/rest/gst?bcost=5.0&gper=2>

Final Cost is: 5.1

Output in chrome :::

-----POSTMAN SCREEN-----




---

### Consumer Code

---

It is Simple Maven Application

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
```

<https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet> -->

```
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

</dependencies>
```

### GSTConsumerApp:

```
package in.nit.test;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTestAssignmentGST {

 public static void main(String[] args) {
 String uri="http://localhost:3031/ParametersProducerAppAssGST-
27";
 String path="/rest/gst";
 try {

 //1. Create Client Object
 Client c=ClientBuilder.newClient();

 //2. Add target and path
 WebTarget wt=c.target(uri).path(path);

 //3. Add parameters Here to Request
 //queyParam(key,value)
 wt=wt.queryParam("bcost",5.2);
 wt=wt.queryParam("gper",2);

 //4. Convert to Request
 Invocation.Builder builder=wt.request();
```

```
//5.Provide method type
Response resp=builder.get();

//6.Print Entity and Status
System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));
} catch (Exception e) {
 e.printStackTrace();
}
}
}
}
```

Execution Steps:

1. First run Producer(WebApp)
2. Consumer(Simple Maven App)

OutPut::

```
200
OK
Final Cost is: 5.304
```

If you are given wrong QueryParam(bcode) in consumer App or zero ::

```
200
OK
Please Enter Base Cost(bcost) and GSTPercentage(gper)
```

## 2. Matrix Parametes(URL;key=value;key=value)

- Matrix parameters are also similar to Query Parameters Only.
- But Symbols are modified from ?, & to ; Symbols.
- Query Parameters are related to WebApplications also supported in webservices as our FC is a Servlet. It is not recommended to use.
- QueryParam works slow compared to MatrixParam.

Q) Why should we not use? And & symbols?

A) Overloaded Operator : +, ?, &  
An operator having multiple meanings at multiple coding types.

It is slow in execution always.

Int +int=> sum

String+String=>append

a>b?true:false

? Positional parameter in JDBC

Generic JDK1.5 ? DataType Decided at runtime

---

- ➔ It supports data format key=value.
- ➔ Both key, values are String type.
- ➔ Symbol used here is ; (semi-colon)
- ➔ Syntax to read data:

`@MatrixParam("key")DataType LocalVariable`

- ➔ It supports reading data and parsing data.
  - ➔ Order is not required to send multiple parameters
  - ➔ If Data Type is not matched, then FrontController(FC) return HTTP Status 404.
  - ➔ If no key is found then default value is given by FC
- 

### Assignment(Token2):

Define one Application using producer JAX-RS API

Module name: Student

Having path: /std

It has operation calculateResult(). It taken 5 MatrixParameters

Those are: sid, sname, eng, mat, science(marks-3 Subjects)

Calculations:

Totoal=sum(eng,mat,sci);

Avg=total/3;

Grade =A+ if avg>=85

A if avg<85 and avg>=60

B if avg<60 and avg>=45

C if avg<45 and avg>=35

Print Final Message:

Hello <StudentName>, with ID:<sid>,

Total marks are: <total>,

Avg is: <avg> and Grade is: <grade>.

Hint: use MatrixParam

Solution::

-----Producer Code-----

Note:: It is a Maven WebApp

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
</dependencies>
```

In web.xml file::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In StudentProducerController::

```
package in.nit.controller;
import javax.ws.rs.GET;
import javax.ws.rs.MatrixParam;
import javax.ws.rs.Path;
@Path("/std")
public class StudentProcuderController {
 //.../std/cal;sid=101;sname=Sankar;eng=93;math=94;sci=91

 @Path("/cal")
 @GET
 public String calculateResult(
 @MatrixParam("sid")int sid,
 @MatrixParam("sname")String sname,
 @MatrixParam("eng")int eng,
 @MatrixParam("mat")int mat,
 @MatrixParam("sci")int sci
) {
```

```
int total=eng+mat+sci;
double avg=total/3.0;
String grade=null;
if(avg>=85) {
 grade="A+";
}
else if(avg<85 && avg>=60) {
 grade="A";
}
else if(avg<60 && avg>=45) {
 grade="B";
}
else if(avg>45 && avg>35) {
 grade="C";
}
else {
 grade="Un Defined";
}
StringBuffer sb=new StringBuffer();

sb.append("Hello: ").append(sname)//String+String=>append
.append(" , With ID: ").append(sid)
.append(" , Total Marks: ").append(total)
.append(" , Average is : ").append(avg)
.append(" , Grade is : ").append(grade);

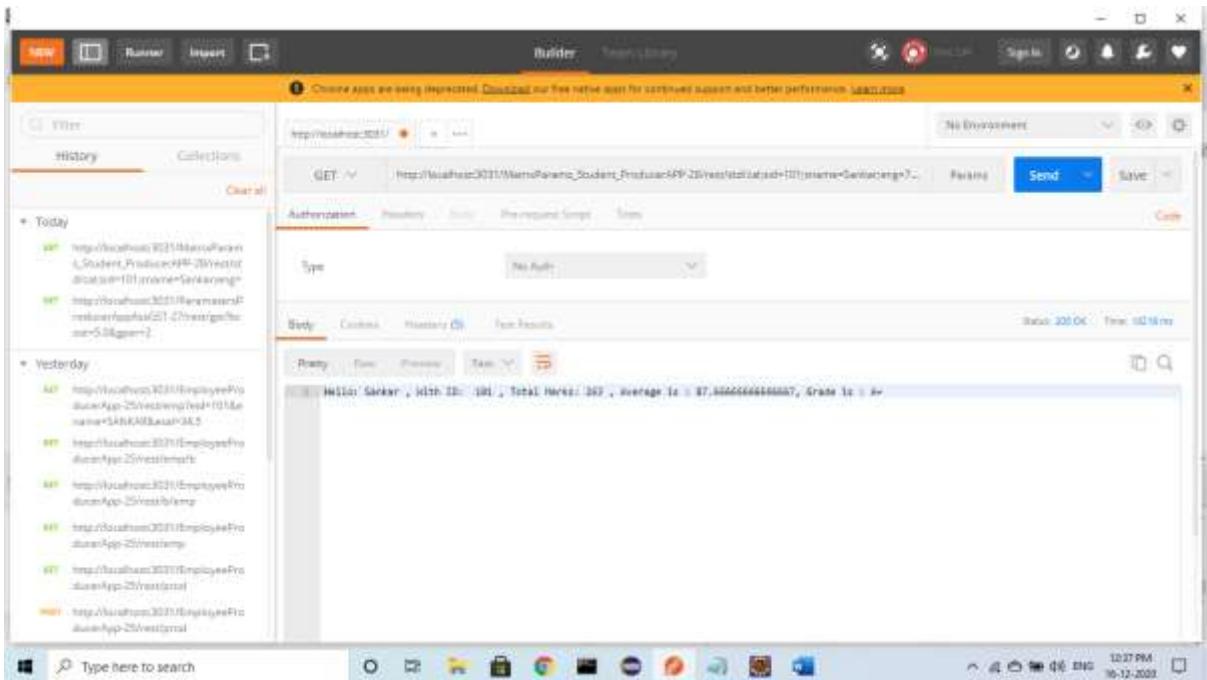
String message=sb.toString();

return message;
}
```

Output in Chrome::

[http://localhost:3031/MatrixParams\\_Student\\_ProducerAPP-28/rest/std/cal;sid=101;sname=Sankar;eng=78;mat=87;sci=98](http://localhost:3031/MatrixParams_Student_ProducerAPP-28/rest/std/cal;sid=101;sname=Sankar;eng=78;mat=87;sci=98)

Hello: Sankar , With ID: 101 , Total Marks: 263 , Average is :  
87.66666666666667, Grade is : A+



[http://localhost:3031/MatrixParams\\_Student\\_ProducerAPP-28/rest/std/cal;sid=101;sname=Sankar;eng=8;mat=7;sci=9](http://localhost:3031/MatrixParams_Student_ProducerAPP-28/rest/std/cal;sid=101;sname=Sankar;eng=8;mat=7;sci=9)

Hello: Sankar , With ID: 101 , Total Marks: 24 , Average is : 8.0,  
Grade is : Un Defined

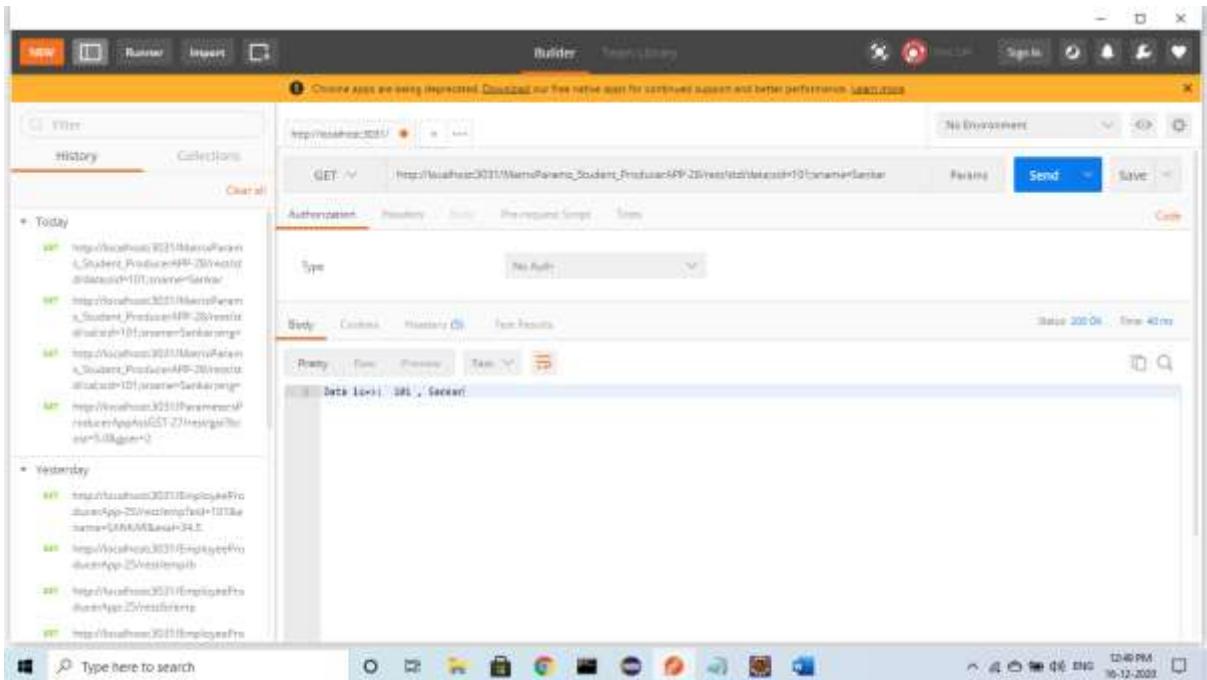
→ To get sid and sname then use another method::

```
@Path("/data")
@GET
public String getData(
 @MatrixParam("sid")int sid,
 @MatrixParam("sname")String sname
) {
 return "Data is=>: "+sid+" "+sname;
}
```

Output in Chrome::

[http://localhost:3031/MatrixParams\\_Student\\_ProducerAPP-28/rest/std/data;sid=101;sname=Sankar](http://localhost:3031/MatrixParams_Student_ProducerAPP-28/rest/std/data;sid=101;sname=Sankar)

Data is=>: 101 , Sankar



### -----Consumer Code-----

Note: it is Simple Maven App

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
```

<https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet> -->

```
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
```

```
</dependency>
<!--

ConsumerStudentTest:


```

```
package in.nit.test;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTestAssignmentStudent {

 public static void main(String[] args) {
 String
uri="http://localhost:3031/MatrixParams_Student_ProducerAPP-
28";
 String path="/rest/std/cal";
 try {

 //1. Create Client Object
 Client c=ClientBuilder.newClient();

 //2. Add target and path
 WebTarget wt=c.target(uri).path(path);

 //3. Add parameters Here to Request
 //MatrixParam(key,value)
 wt=wt.matrixParam("sid","101");
 wt=wt.matrixParam("sname","Sname");
 wt=wt.matrixParam("eng",98);
 wt=wt.matrixParam("mat",96);
 wt=wt.matrixParam("sci",94);
```

```
//4. Convert to Request
Invocation.Builder builder=wt.request();

//5. Provide method type
Response resp=builder.get();

//6. Print Entity and Status
System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));
} catch (Exception e) {
 e.printStackTrace();
}
}
}
}
```

Execution Process::

1. Producer App(it is Web App)
2. Consumer App( it is Simple Maven App)

Output::

```
200
OK
Hello: Sname , With ID: 101 , Total Marks: 288 , Average is : 96.0,
Grade is : A+
```

---

### Assignment (Token3):

Define one Application using JAX-RS  
Module: Product having path: /prod  
It has method calculateBill()  
It takes 5 Matrix Params like: Product Code (PCODE),  
Product Cost (PCOST), discount, GST, vendor name(VNAME).

Calculations:

disAmt=PCOST\*discount/100.0;  
GSTAmt=PCOST \* GST/100.0;

finalAmt=pcost+gstAmt-disAmt;

Message: Product : <PCODE>, Final amount is : <filaAmt> Given by : <vendor>, with discount amount : <disAmt> and GST Amount : <gstAmt>

Solution::

-----Producer App-----

Note: it is WebApp

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
```

```
<version>2.32</version>
</dependency>

</dependencies>
```

In web.xml file::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In ProcuderController

```
package in.nit.controller;
import javax.ws.rs.GET;
import javax.ws.rs.MatrixParam;
import javax.ws.rs.Path;
@Path("/prod")
public class ProcuderController {
 //.../prod;pcode=1301A;pcost=65.00;discount=2;gst=1.4;vname=S
 ankar

 @GET
 public String calculateBill(
 @MatrixParam("pcode")String pcode,
 @MatrixParam("pcost")double pcost,
 @MatrixParam("discount")int discount,
 @MatrixParam("gst")int gst,
```

```
@MatrixParam("vname")int vname
) {
double disAmt=pcost*discount/100.0;

double gstAmt=pcost*gst/100.0;

double finalAmt=pcost+gstAmt-disAmt;

StringBuffer sb=new StringBuffer();

sb.append(" Product :
").append(pcode)//String+String=>append
.append(" ,Final Amount is : ").append(finalAmt)
.append(" , Given By Vendor : ").append(vname)
.append(" , with discount Amount : ").append(disAmt)
.append(", And GST Amount : ").append(gstAmt);

String message=sb.toString();

return message;
}
}
```

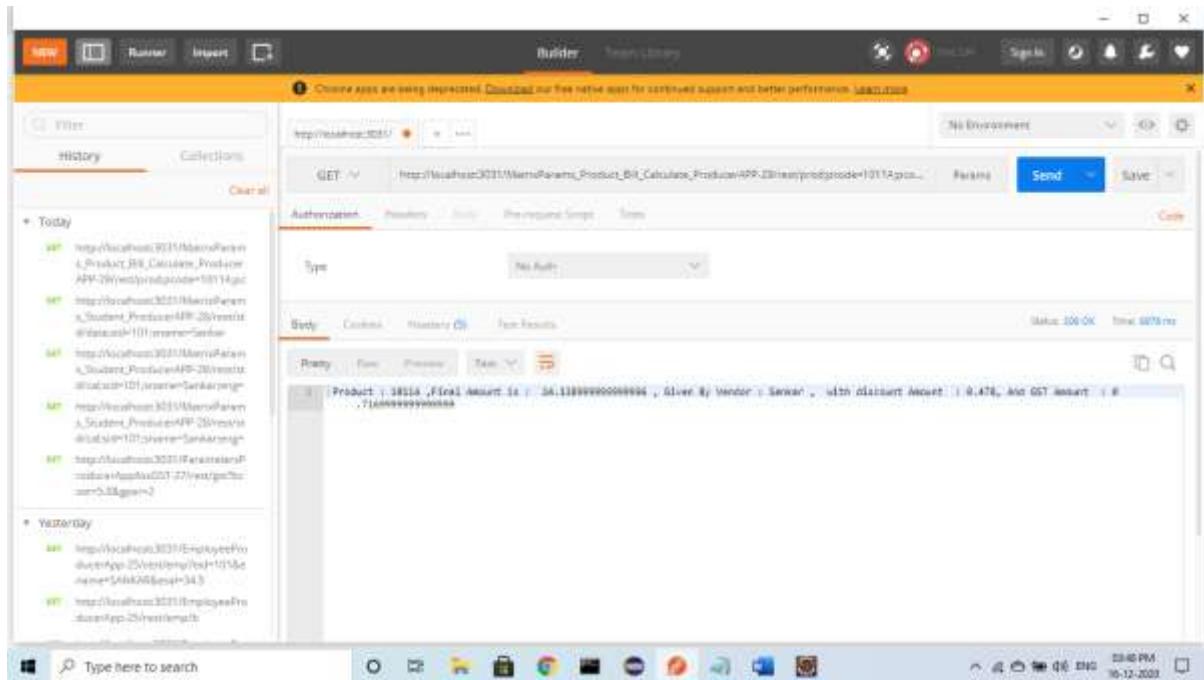
Output::

In chrome:

[http://localhost:3031/MatrixParams\\_Product\\_Bill\\_Calculate\\_ProducerAP\\_P-29/rest/prod;pcode=1011A;pcost=23.9;discount=2;gst=3;vname=Sankar](http://localhost:3031/MatrixParams_Product_Bill_Calculate_ProducerAP_P-29/rest/prod;pcode=1011A;pcost=23.9;discount=2;gst=3;vname=Sankar)

Product : 1011A ,Final Amount is : 24.138999999999996 , Given By Vendor : Sankar , with discount Amount : 0.478, And GST Amount : 0.7169999999999999

-----POSTMAN SCREEN-----



### -----Consumer Code-----

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
```

<https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet> -->

```
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<version>2.32</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

</dependencies>
```

## In ConsumerBillTest

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTestAssignmentProductBill {

 public static void main(String[] args) {
 String
uri="http://localhost:3031/MatrixParams_Product_Bill_Calculate_Produc
erAPP-29";
 String path="/rest/prod";
 try {
 //1. Create Client Object
```

```
Client c=ClientBuilder.newClient();

//2. Add target and path
WebTarget wt=c.target(uri).path(path);

//3. Add parameters Here to Request
//MatrixParam(key,value)
wt=wt.matrixParam("PCODE","1011A");
wt=wt.matrixParam("PCOST",23.43);
wt=wt.matrixParam("DISCOUNT",4);
wt=wt.matrixParam("GST",3);
wt=wt.matrixParam("VNAME","Sankar");

//4. Convert to Request
Invocation.Builder builder=wt.request();

//5. Provide method type
Response resp=builder.get();

//6. Print Entity and Status
System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));

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

}
```

Output::

Execution Steps:

1. Run Producer App (it is WebApp)
2. Run Consumer App (it is Simple Maven App)

200

OK

Product : 1011A ,Final Amount is : 14.7609 , Given By Vendor : Sankar , with discount Amount : 9.372, And GST Amount : 0.7029

---

### Assignment (Token4):

Define one Application using JAX-RS

Module name: Recharge having path: /rech

It has method payAmt() it takes Matrix Parameters.

Like serviceId, billAmt, custId, discountVal.

Calculation:

discountVal = 1-10% (Generate Random)

disAmt = billAmt \* discountVal / 100.0;

finalAmt = billAmt - disAmt;

Hint: Use class Random and Method nextInt(n)

Message:

Hello: <custId>, thank you for <serviced>, Actual Bill is <billAmt>,  
Congrats!! Discount Provided : <disAmt>.   
Final Pay Amount is : <finalAmt>

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
</dependencies>
```

In web.xml File:::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
```

```
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In ProducerRechargeController

```
package in.nit.controller;
import java.util.Random;

import javax.ws.rs.GET;
import javax.ws.rs.MatrixParam;
import javax.ws.rs.Path;
@Path("/rech")
public class RechargeController {
 //.../rech;serviceId=1301265;billAmt=65.00;customerId=2;gst=1.4

 @GET
 public String calculateBill(
 @MatrixParam("serviceId")Long serviceId,
 @MatrixParam("billAmt")double billAmt,
 @MatrixParam("customerId")int customerId
) {

 int discountVal=new Random().nextInt(10);

 double disAmt= billAmt*discountVal/100.0;

 double finalAmt=billAmt-disAmt;

 StringBuffer sb=new StringBuffer();

 sb.append(" Hello :");
 ").append(customerId)//String+String=>append
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
.append(" , Thank you for : ").append(serviceId)
.append(" , Acutal Bill Is : ").append(billAmt)
.append(" , Congrats!! discount provided :
").append(disAmt)
.append(" , Final Pay Amount Is : ").append(finalAmt);

String message=sb.toString();

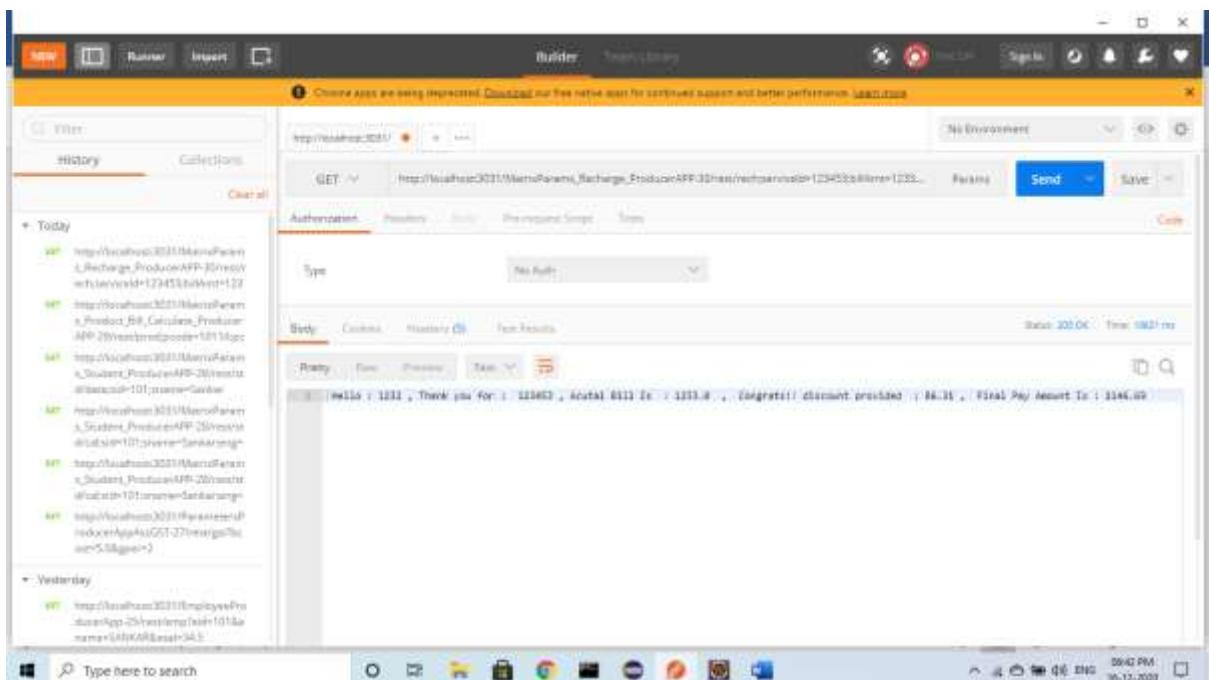
return message;
}
}
```

Output::

In chrome:

[http://localhost:3031/MatrixParams\\_Recharge\\_ProducerAPP-30/rest/rech;serviceId=123453;billAmt=1233;custId=1232](http://localhost:3031/MatrixParams_Recharge_ProducerAPP-30/rest/rech;serviceId=123453;billAmt=1233;custId=1232)

Hello : 1232 , Thank you for : 123453 , Acutal Bill Is : 1233.0 , Congrats!!  
discount provided : 98.64 , Final Pay Amount Is : 1134.36



-----Consumer Code-----

It is Simple Maven Project

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
</dependencies>
```

In RechargeConsumerTest::

```
package in.nit.test;
```

```
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTestAssignmentRecharge {

 public static void main(String[] args) {
 String
uri="http://localhost:3031/MatrixParams_Recharge_ProducerAPP-
30";
 String path="/rest/rech";
 try {

 //1. Create Client Object
 Client c=ClientBuilder.newClient();

 //2. Add target and path
 WebTarget wt=c.target(uri).path(path);

 //3. Add parameters Here to Request
 //MatrixParam(key,value)
 wt=wt.matrixParam("serviceId","1011232");
 wt=wt.matrixParam("billAmt",23.43);
 wt=wt.matrixParam("custId",4231);
 //4. Convert to Request
 Invocation.Builder builder=wt.request();

 //5. Provide method type
 Response resp=builder.get();

 //6. Print Entity and Status
 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 System.out.println(resp.readEntity(String.class));
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

}

Output::

200

OK

Hello : 4231 , Thank you for : 1011232 , Acutal Bill Is : 23.43 ,  
Congrats!! discount provided : 1.4058 , Final Pay Amount Is : 22.0242

---

## Implementation of Webservices:

To do WebServices coding, we should define two applications.

- 1) Producer Application
- 2) Consumer Application

# 1) Producer Application(used Jersey2.32):

It must be a web application.

This application works based on FC (FrontController) Design Pattern.

Here we should define two files for providing service.

- a) web.xml
  - b) Producer Class
- a) web.xml:

In this file we should configure a pre-defined Servlet using Directory Match URL Pattern (Example: /rest/\*).

In case of Jersey1.x vendor, FrontController(FC) Name is: ServletContainer given in package:

“org.glassfish.jersey.servlet”.

FrontController(FC) will navigate request to one Producer class method and takes response back, at last sends to client.

b) Producer class:

This class will do actual work for Producer Application.  
It is also called as Resource or  
Service Provider class.

It must have unique path for every class and method.

Use ReST annotations, which are given in package: “javax.ws.rs”, over this class and methods.

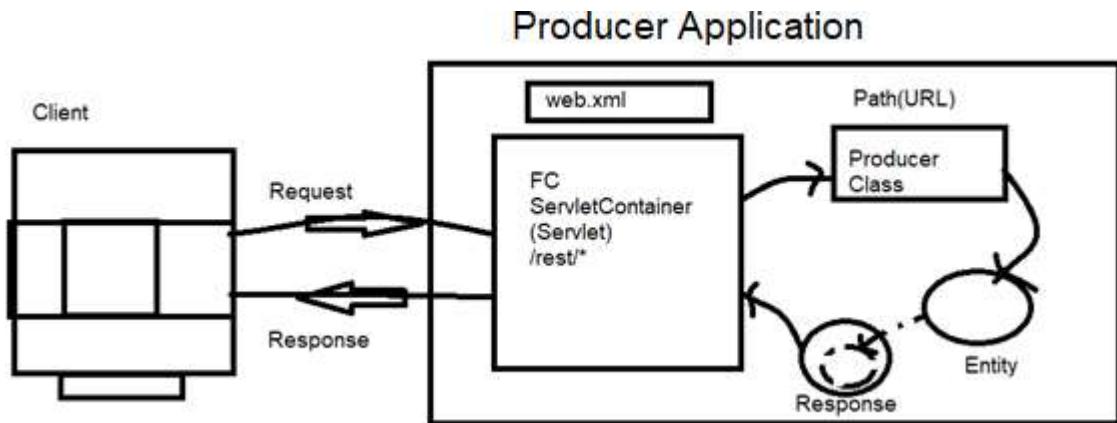
## Example for ReST annotations are:

1. @Path
2. @GET
3. @Consumes
4. @Produces etc.....

- a) Provide @Path(“/url”) at class level.
- b) Define at least one method and must have Annotation of HTTP Method (7 =GET, POST, PUT, DELETE, TRACE, OPTIONS, HEAD).

Note: There is no default type in ReST WebServices.

## FrontController Design for Producer Application:



Example Code:

---

-----Producer Code-----Jersey2.32-----

Software's: JDK15  
Tomcat9

In pom.xml File:

```

<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.32</version>
</dependency>

<dependency>
```

In web.xml file

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

**package** in.nit.controller;

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/prod")
public class ProducerController {
 @Path("/s")
 @GET
 public String messageOne() {

 return "message displayed..";
 }
}
```

## Output::

<http://localhost:3031/JerseyTestApp/rest/prod/s>  
message displayed..

-----Consumer Jersey2.32-----

In pom.xml File::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>

 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
```

```
</dependency>
</dependencies>
```

### In ConsumerTest

```
package in.nit.test;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTest {
 public static void main(String[] args) {
 try {
 String url="http://localhost:3031/ProducerJersey2_32-31";
 String path="/rest/prod/s";

 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(url).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.get();
 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 System.out.println(resp.readEntity(String.class));
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 }
}
```

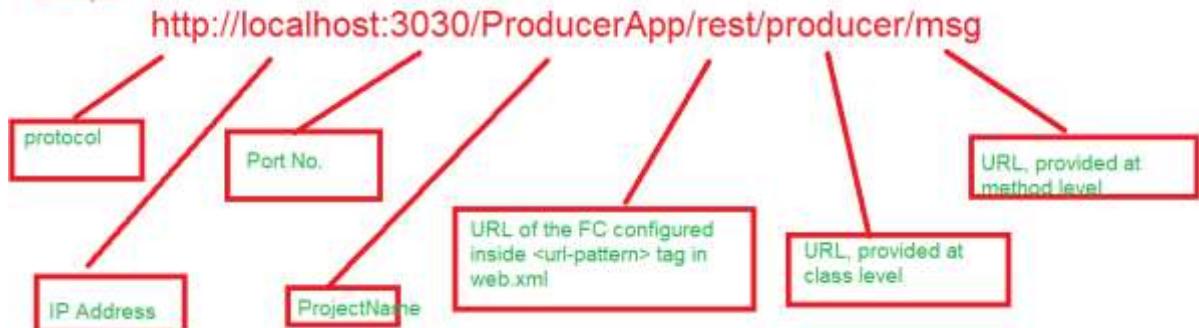
Output:

200  
OK  
Message Displayed...

---

Protocol://IPAddress:portNo/ProjectName/FCURLWebxml/Classpath/MethodPath

Example:



## HTTP Method Types used in ReST Webservices:

Used(4)+Un\_used(5)=total(9)

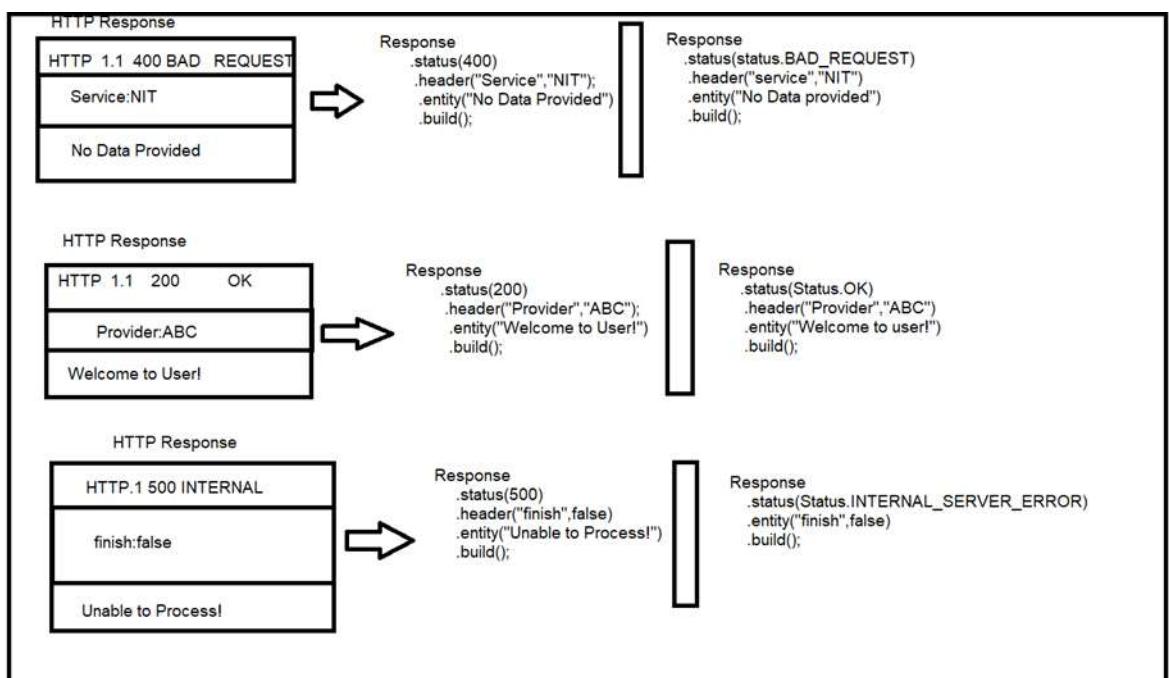
| sl.no | Method Type | Description                                                                                                                                                                                                                                                                                              |
|-------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | GET         | This option is used to indicate <b>fetching</b> resource from Server to Client                                                                                                                                                                                                                           |
| 2     | POST        | It indicates create new resource at Server Side.                                                                                                                                                                                                                                                         |
| 3     | PUT         | It indicates <b>modify</b> existed Resource at Server Side.                                                                                                                                                                                                                                              |
| 4     | DELETE      | It indicates remove existed Resource at Server Side.                                                                                                                                                                                                                                                     |
| 5     | HEAD        | It indicates do some work at Server Side and return nothing (empty body to Client).                                                                                                                                                                                                                      |
| 6     | OPTIONS     | It indicates execute multiple blocks or multiple tasks or multiplexing.                                                                                                                                                                                                                                  |
| 7     | TRACE       | The HTTP TRACE method performs a message loop-back test along the path to the target resource, providing a useful debugging mechanism.                                                                                                                                                                   |
| 8     | PATCH       | HTTP PATCH requests are to make partial update on a resource. If you see PUT requests also modify a resource entity, so to make more clear – PATCH method is the correct choice for partially updating an existing resource, and PUT should only be used if you're replacing a resource in its entirety. |
| 9     | CONNECT     | The HTTP CONNECT method starts two-way communications with the requested resource. It can be used to open a tunnel.                                                                                                                                                                                      |

Here Resource means=> It can be a File/Audio/Video/Image/Text etc.....

HEAD, TRACE, CONNECT, OPTIONS are not used in ReST application development.

Client Tools:

1. POSTMAN TOOL
2. Advanced ReST Client
3. Insomnia.



## Response Construction in JAX-RS

- Every Service Producer method return type can be Response.
- Here Response indicates HTTP Response given back to Consumer.

It contains mainly 3 things. They are

- a) HTTP Response Status
  - b) HTTP Header Param
  - c) HTTP Response Body.
- a) Hader Response Status
- |     |                   |
|-----|-------------------|
| 1xx | Information       |
| 2xx | Success           |
| 3xx | Redirect          |
| 4xx | Client-Side Error |
| 5xx | Server-Side Error |

\*\* Syntax to create Response Object:

Response

```
.status(code)
.header(k,v)
.entity(data)
.build();
```

---

#### -----Notes-----

- ➔ Response it is an abstract class
- ➔ It is having a static (Overloader) method
  - status(Status enum)
  - status(int code)
- ➔ status() methods returns ResponseBuilder
- ➔ ResponseBuilder is having supportive methods

Like header(), entity(), type().....etc

- ➔ To add data to Response(Body) Object use method entity(Object).
- ➔ Finally call build() method converts ResponseBuilder to Response Object.

---

#### -----Example-----

In pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
```

```
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### In ProducerController

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Response;

@Path("/resp")
public class ProducerResponseConstrcutorController {

 @Path("/r")
 @GET
 public Response doOperation(
 @QueryParam("input") Integer value
) {
 Response resp=null;
 try {

 if(value==null) {
 resp=Response
 .status(400)
 .header("result: ", "Not Executed")
 .entity("No Data given")
 .build();
 }
 else {
 int res=10/value;
 resp=Response
 .status(200)
 .header("result : ", "GOOD")
 }
 }
 }
}
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
 .entity("result is : "+res)
 .build();
 }

} catch (Exception e) {

 resp=Response
 .status(500)
 .header("Rslt : ","Faild")
 .entity("Unable to Process Request!! : ")
 .build();
 e.printStackTrace();
}

return resp;
}
}
```

Output:

In chrome:

[http://localhost:3031/ResponseConstruction\\_JAX\\_RS\\_32/rest/resp/r](http://localhost:3031/ResponseConstruction_JAX_RS_32/rest/resp/r)

No Data given

[http://localhost:3031/ResponseConstruction\\_JAX\\_RS\\_32/rest/resp/r?input=](http://localhost:3031/ResponseConstruction_JAX_RS_32/rest/resp/r?input=)

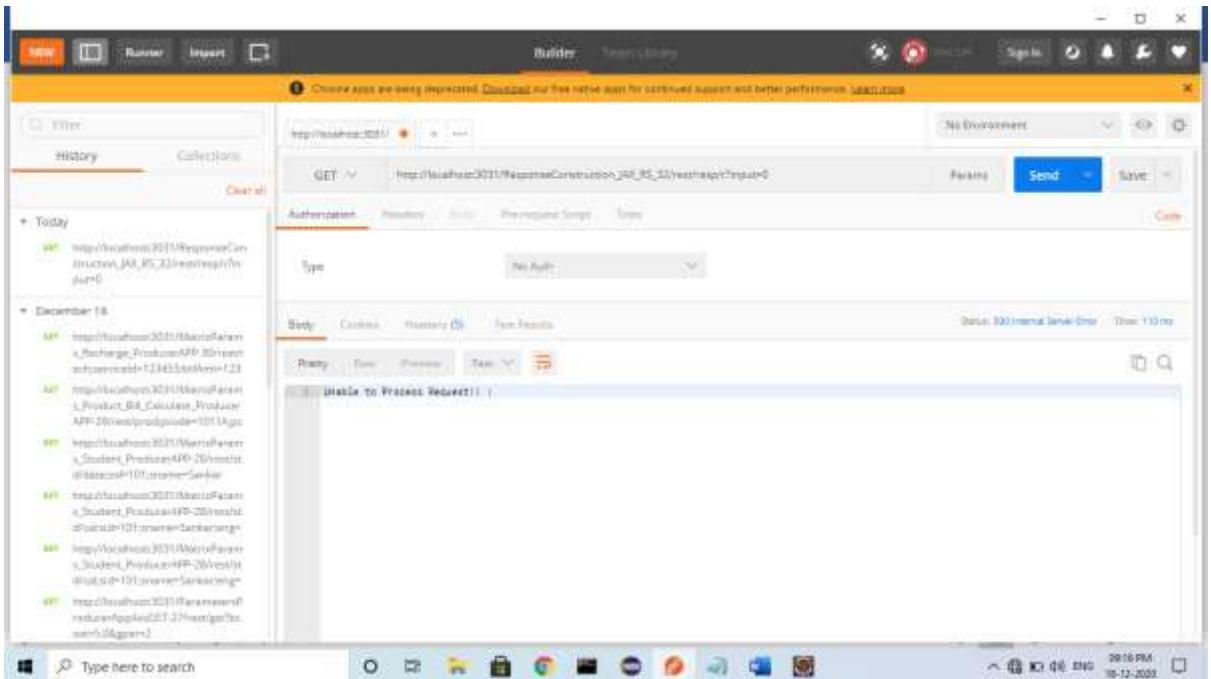
No Data given

[http://localhost:3031/ResponseConstruction\\_JAX\\_RS\\_32/rest/resp/r?input=10](http://localhost:3031/ResponseConstruction_JAX_RS_32/rest/resp/r?input=10)

result is : 1

[http://localhost:3031/ResponseConstruction\\_JAX\\_RS\\_32/rest/resp/r?input=0](http://localhost:3031/ResponseConstruction_JAX_RS_32/rest/resp/r?input=0)

Unable to Process Request!! :



## Response as ReturnType in Producer App

---

Response [abstract class] Define in package: javax.ws.rs.core.

Response it is like HTTP Response which holds details like  
(Response=) Status Details+HTTP Headers+body(entity)

### Syntax-----

Response res=Response.status(\_).header(k,v).entity(data).build();

## 3. Header Parameters(key=Value)

[HTTPResponse Head]

( used for Security details like username/password/token/otp..... )

- In ReST, we can send secure details like username, password, OTP, Token , ....etc using HTTP Request Header from Consumer to Producer.
- You can Test in Producer Application also. Producer is a web App.

@HeaderParam("key")DataType LocalVariable

### Assignment(Token5):

Develop one JAX-RS Producer App

Module Name: User

Method name: validateUser();

That taken 2 Header Params user, pwd.

If They are null/empty REQUEST, No Data Provided) →Response(400-BAD)

If user/pwd values are developer/Sankar →Response(200-OK,Welcome to Admin)

Else AUTHORIZED, Invalid user/pwd). →Response(401-UNAUTHORIZED)

Solution::

-----Producer App-----

It is web App:

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/* </url-pattern>
```

```
</servlet-mapping>
</web-app>
```

In UserProducerController:

string trim() method

Eliminates leading and trailing Spaces.

trim() method checks spaces in value before and after value, if it exists then removes the spaces and return the String.

```
package com.nt.test;
public class StringTrimTest {
 public static void main(String[] args) {
 //String is a group of characters
 String person_names= " ravi sankar jagadesh ";
 //String length
 System.out.println(person_names.length());
 // trim
 String t=person_names.trim();
 System.out.println(t);
 // after trim, spaces removed
 System.out.println(t.length());
 }
}
```

Output::

```
22
ravi sankar jagadesh
20
```

---

```
javax.ws.rs.core.Response.Status (Status is inner class)
Status is enum class, enum is a group of named constants.

OK(200, "OK")
CREATED(201, "Created")
ACCEPTED(202, "Accepted")
NO_CONTENT(204, "No Content")
RESET_CONTENT(205, "Reset Content")
PARTIAL_CONTENT(206, "Partial Content")
MOVED_PERMANENTLY(301, "Moved Permanently")
FOUND(302, "Found")
SEE_OTHER(303, "See Other")
NOT_MODIFIED(304, "Not Modified")
USE_PROXY(305, "Use Proxy")
TEMPORARY_REDIRECT(307, "Temporary Redirect")
BAD_REQUEST(400, "Bad Request")
UNAUTHORIZED(401, "Unauthorized")
PAYMENT_REQUIRED(402, "Payment Required")
FORBIDDEN(403, "Forbidden")
NOT_FOUND(404, "Not Found")
METHOD_NOT_ALLOWED(405, "Method Not Allowed")
NOT_ACCEPTABLE(406, "Not Acceptable")
PROXY_AUTHENTICATION_REQUIRED(407, "Proxy Authentication Required")
```

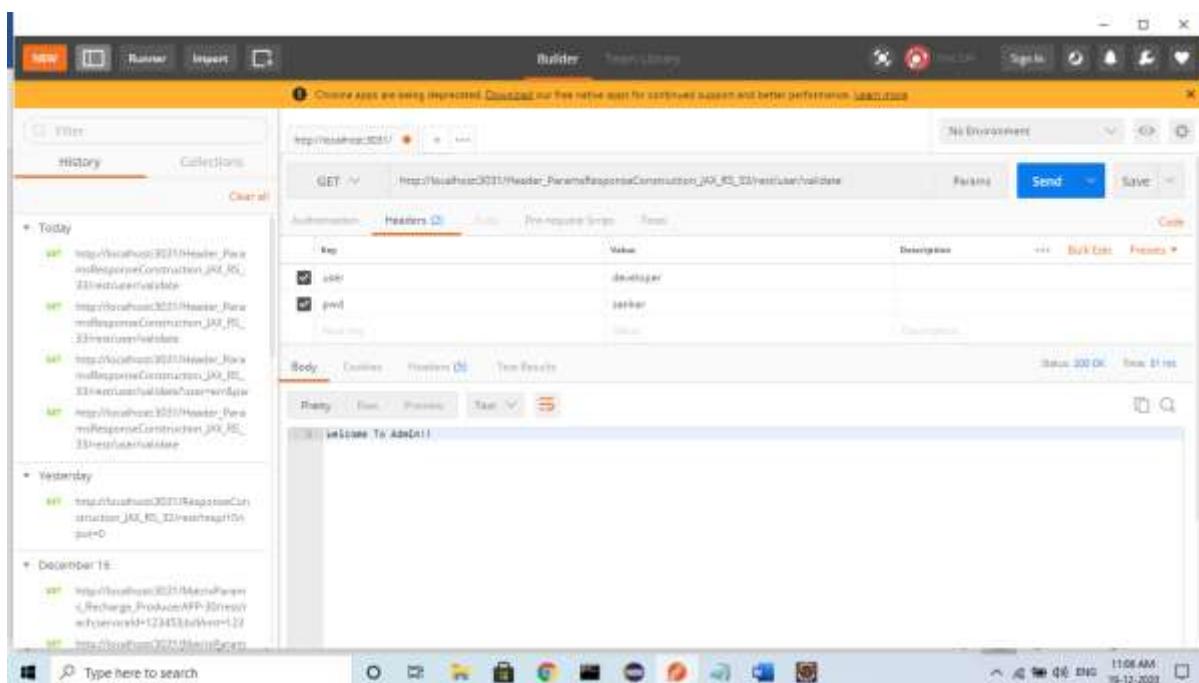
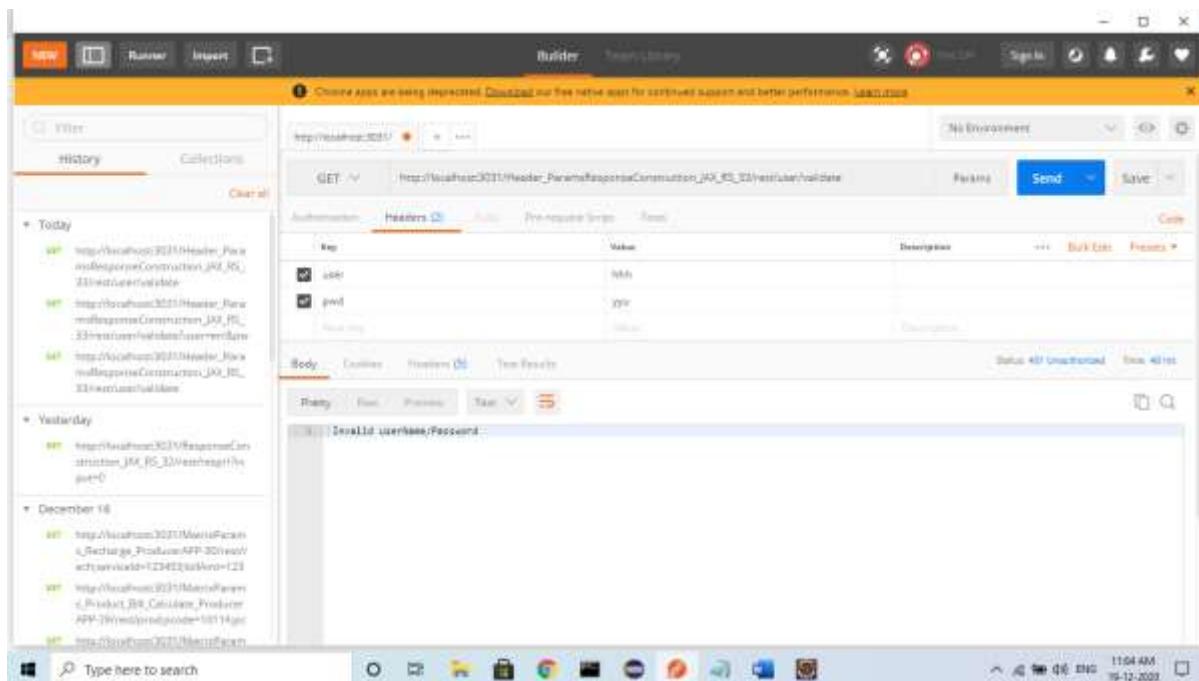
```
REQUEST_TIMEOUT(408, "Request Timeout")
CONFLICT(409, "Conflict")
GONE(410, "Gone")
LENGTH_REQUIRED(411, "Length Required")
PRECONDITION_FAILED(412, "Precondition Failed")
REQUEST_ENTITY_TOO_LARGE(413, "Request Entity Too Large")
REQUEST_URI_TOO_LONG(414, "Request-URI Too Long")
UNSUPPORTED_MEDIA_TYPE(415, "Unsupported Media Type")
REQUESTED_RANGE_NOT_SATISFIABLE(416, "Requested Range Not Satisfiable")
EXPECTATION_FAILED(417, "Expectation Failed")
PRECONDITION_REQUIRED(428, "Precondition Required")
TOO_MANY_REQUESTS(429, "Too Many Requests")
REQUEST_HEADER_FIELDS_TOO_LARGE(431, "Request Header Fields Too Large")
INTERNAL_SERVER_ERROR(500, "Internal Server Error")
NOT_IMPLEMENTED(501, "Not Implemented")
BAD_GATEWAY(502, "Bad Gateway")
SERVICE_UNAVAILABLE(503, "Service Unavailable")
GATEWAY_TIMEOUT(504, "Gateway Timeout")
HTTP_VERSION_NOT_SUPPORTED(505, "HTTP Version Not Supported")
NETWORK_AUTHENTICATION_REQUIRED(511, "Network Authentication Required")
```

---

Output::

In Chrome HeaderParm output not possible to see

So use POSTMAN TOOL.



**package** in.nit.controller;

```
import javax.ws.rs.GET;
import javax.ws.rs.HeaderParam;
```

```
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

@Path("/user")
public class UserProducerResponseConstrutorController {
 @Path("/validate")
 @GET
 public Response validateUser(
 @HeaderParam("user")String user,
 @HeaderParam("pwd")String pwd
) {
 Response resp=null;
 try {
 if(user==null || "" .equals(user.trim()) || pwd==null ||
"" .equals(user.trim())) {
 //resp=Response.status(400).entity("No Data
Provided").build();
 //or

 resp=Response.status(Status.BAD_REQUEST).entity("No Data
Provided").build();
 }
 else if(user.equals("developer") &&
pwd.equals("sankar")) {
 //resp=Response.status(200).entity("welcome To
Admin!!").build();
 //or

 //resp=Response.status(Status.OK).entity("welcome To Admin!!
").build();
 //or
 resp=Response.ok("welcome To Admin!!
").build();
 }
 else {
 //resp=Response.status(401).entity(" Invalid
userName/Password ").build();
 //or

 resp=Response.status(Status.UNAUTHORIZED).entity(" Invalid
userName/Password ").build();
 }
 }
 }
}
```

```
 } catch (Exception e) {
 e.printStackTrace();
 }
 return resp;
 }
}
```

-----POSTMAN TOOL-----

GET [http://localhost:3031/Header\\_ParamsResponseConstruction\\_JAX\\_RS\\_33/rest/user/validate](http://localhost:3031/Header_ParamsResponseConstruction_JAX_RS_33/rest/user/validate) SEND

Headers

| Key  | value     |
|------|-----------|
| user | developer |
| pwd  | Sankar    |

- In Consumer use method header(String key, Object value) defined  
In Invocation.Builder that returns same build type only.
- Provide this code in Consumer after creating builder Object and  
before calling method type(get() / post() / put().....etc]

-----Consumer Code for Header Params-----

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>

```

```
</dependency>
<!--

<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
</dependencies>
```

In ConsumerUserTest

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerUserTest {

 public static void main(String[] args) {
 String url="http://localhost:3031/Header_ParamsResponseProducer_JAX_RS_33";
 String path="rest/user/validate";
 try {

 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(url).path(path);

 Invocation.Builder builder=wt.request();
 builder=builder.header("user","developer");
 }
 }
}
```

```
builder=builder.header("pwd","sankar");
Response resp=builder.get();

System.out.println(resp.getStatus());//200
System.out.println(resp.getStatusInfo());//ok
System.out.println(resp.readEntity(String.class));

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

}
```

Execution Steps::

- Producer app it is web app.
- Consumer app it is Simple Maven App

Output:

---

```
200
OK
welcome To Admin!!
```

-----Short Code for Header Params-----

- We can define same Code in short format using Builder reference before calling method type.

```
Response resp=builder.header(key,value)
 .header(key,value)
 .header(key,value).....
 .get()/post().....;
```

```
package in.nit.test;
```

```
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
```

```
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerUserTest {

 public static void main(String[] args) {
 String url="http://localhost:3031/Header_ParamsResponseProducer_JAX_RS_33";
 String path="rest/user/validate";
 try {

 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(url).path(path);

 Invocation.Builder builder=wt.request();
 /*
 * builder=builder.header("user","developer");
 * builder=builder.header("pwd","sankar");
 * Response resp=builder.get();
 */
 Response resp=builder
 .header("user","developer")
 .header("pwd","sankar")
 .get();

 System.out.println(resp.getStatus());//200
 System.out.println(resp.getStatusInfo());//ok
 System.out.println(resp.readEntity(String.class));

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

---

-----CO\_DEC-----

## [Encoding and Decoding]

CO=enCODing and DEC=DECoding

Encoding=Readable Context is converted to Un-readable format

Example: Hello Admin!=> hshj\$%2kjhg>, \*

Decoding=Encoded content is converted back to Readable format

Example: Hshj\$%2k1jhgg,\* => Hello Admin!

**API Name: Apache Commons Codec.**

This API is used to perform Encoding and Decoding by using Java classes/methods.

class: Base64

package: org.apache.commons.codec.binary

-----Static Methods-----

encodeBase64(byte[] normalData): byte[] encodedData

decodeBase64(byte[] encodeedData): byte[] normalData.

---

Encryption:

the process of converting information or data into a code, especially to prevent unauthorized access.

**package** in.nit.test;

**import** java.util.Base64;

//core API

**public class** Encoding\_Decoding\_Test {

**public static void** main(String[] args) {

```
String sentence="you are good boy...";

//Create Encoder Object
Base64.Encoder en_Base=Base64.getEncoder();

//Encoding String
String encode=en_Base.encodeToString(sentence.getBytes());

//Encoded String
System.out.println("Encoded String : "+encode);

System.out.println("-----");

//Decoding Object
Base64.Decoder de_Base=Base64.getDecoder();

//DeCoded String
String decode=new String(de_Base.decode(encode.getBytes()));

//decoded String
System.out.println("Decoded String : "+decode);
}
}
```

Output::

```
Encoded String : eW91IGFyZSBnb29kIGJveS4uLg==

Decoded String : you are good boy...
```

---

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!-- https://mvnrepository.com/artifact/commons-
codec/commons-codec -->
 <dependency>
 <groupId>commons-codec</groupId>
 <artifactId>commons-codec</artifactId>
 <version>1.15</version>
 </dependency>
</dependencies>
```

In Test class

```
package in.nit.test;

import org.apache.commons.codec.binary.Base64;

public class EnDecUsingCommonDecoTest {

 public static void main(String[] args) {

 //Original String or group of characters
 String welcome_Message="Welcome to Java Coding!";

 //converting String to bytes
 byte[] input=welcome_Message.getBytes();

 //Encode
 byte[] output=Base64.encodeBase64(input);

 //convert Encode to String
 String encode=new String(output);
 //Encoded String
 System.out.println(encode);
```

```
//Decode
byte[] output1=Base64.decodeBase64(output);

//convert Decode to String
String decode=new String(output1);

//Original String
System.out.println(decode);

}

}
```

### Output::

V2VsY29tZSB0byBKYXZhIENvZGluZyE=  
 Welcome to Java Coding!

```
package in.nit.test;
import java.util.Base64;
public class Encoding_Decoding_Test {
 public static void main(String[] args) {
 String sentence="you are good boy..";
 //Create Encoder Object
 Base64.Encoder en_Base64=Base64.getEncoder();
 //Encoding String
 String encode=en_Base64.encodeToString(sentence.getBytes());
 //Encoded String
 System.out.println("Encoded String "+encode);
 System.out.println("-----");
 //Decoding Object
 Base64.Decoder de_Base64=Base64.getDecoder();
 //Decoded String
 String decode=new String(de_Base64.decode(encode.getBytes()));
 //decoded String
 System.out.println("Decoded String "+decode);
 }
}
```

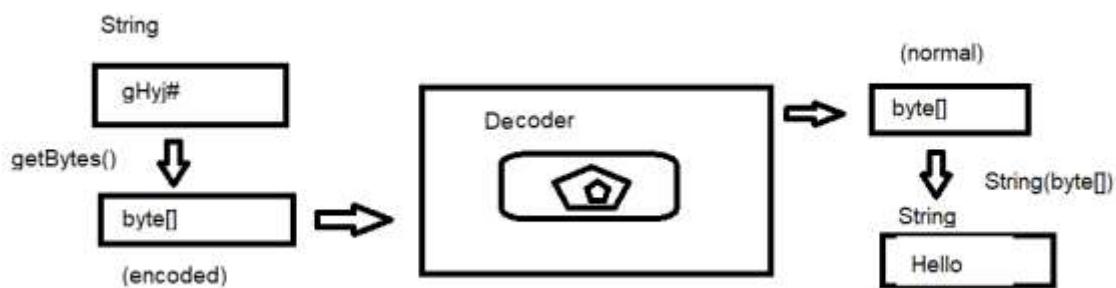
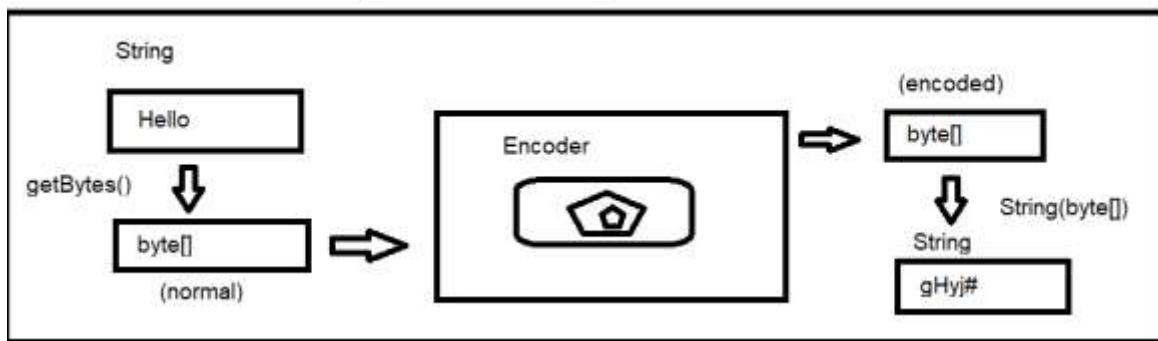
**Encryption**  
 The process of converting information or data into a code, especially to prevent unauthorized access.

```
package in.nit.test;
import org.apache.commons.codec.binary.Base64;
public class EnDecUsingCommonDecoTest {
 public static void main(String[] args) {
 //Original String or group of characters
 String welcome_Message="Welcome to Java Coding!";
 //Converting String to bytes
 byte[] inst=welcome_Message.getBytes();
 //Encode
 byte[] output=Base64.encodeBase64(inst);
 //Convert Encode to String
 String encode=new String(output);
 //Encoded String
 System.out.println(encode);
 //Decode
 byte[] output1=Base64.decodeBase64(output);
 //Convert Decode to String
 String decode=new String(output1);
 //Original String
 System.out.println(decode);
 }
}
```

Output:  
 V2VsY29tZSB0byBKYXZhIENvZGluZyE=  
 Welcome to Java Coding!

Output:  
 Encoded String : eW91IGFyZSB0by29hIGJve54uLg==  
 Decoded String : you are good boy..

### CODEC WORK FLOW



# CODEC

## enCODing and DECOding

Encoding: Converting readable data into unreadable format.

Example: sam-> Gbsf\$6f2>...Fr2

Decoding: Converting encoded data( or unreadable format)

Back to readable format is called as Decoding.

Example: Gbsf\$6f2>...Fr2-> sam

API: Apache Commons CODEC is a open source and light weight API used to perform CODEC operations.

Package: org.apache.commons.codec.binary

Class: Base64

static Methods: encodeBase64(byte[] normal): byte[] encoded

decodeBase64(byte[] encoded): byte[] normal

In pom.xml file:

```
<!-- https://mvnrepository.com/artifact/commons-codec/commons-codec
-->
<dependency>
 <groupId>commons-codec</groupId>
 <artifactId>commons-codec</artifactId>
 <version>1.15</version>
</dependency>
```

Note:

- To convert String Object to byte[] use method getBytes()  
Example: String s=\_\_\_\_\_;  
byte[] arr=s.getBytes();
- To convert byte[] to String use Param Constructor String(byte[])  
Example: byte[] arr=\_\_\_\_\_;  
String s=new String(arr);

-----Example Project-----

1. Create Simple Maven Project:

2. Add below dependency in pom.xml file

```
<!-- https://mvnrepository.com/artifact/commons-
codec/commons-codec -->
```

```
<dependency>
 <groupId>commons-codec</groupId>
 <artifactId>commons-codec</artifactId>
 <version>1.15</version>
</dependency>
```

3. Define code for CODEC

**package** in.nit.test;

**import** org.apache.commons.codec.binary.Base64;

```
public class TestCodec {

 public static void main(String[] args) {
 //1. Take One input String
 String india="India is a Multi region Country";

 //2.Convert to byte[] format
 byte[] characters=india.getBytes();

 //3.Give data to Encoder
 byte[] encode=Base64.encodeBase64(characters);

 //4.convert Output byte[] to String
 String converted_String=new String(encode);

 //5.Print Encoding String
 System.out.println(converted_String);

 //-----
 //-----//

 //6.Read encoded String and convert to byte[]
 byte[] decode=converted_String.getBytes();

 //7.Decode byte[] given
 byte[] decoded_String=Base64.decodeBase64(decode);

 //8.Convert back to String Data Type
 String original_String=new String(decoded_String);

 //9.Print Decoded Data
 System.out.println(original_String);
 }
}
```

Output::

---

SW5kaWEgaXMgYSBNdWx0aSByZWdpb24gQ291bnRyeQ==  
India is a Multi region Country

---

-----JDK8 Based CODEC API-----

API Details(as overview)

```
package java.util;
public class Base64{
 //To get Encoder Object
 public static Encoder getEncoder(){...}
 //To get Decoder Object
 public static Decoder getDecoder(){...}

 //two static inner classes
 public static class Encoder{
 Public byte[] encode(byte[] src){...}
 }

 public static class Decoder{
 public byte[] decode(byte[] src){...}
 }
}
```

-----Note-----

→ Compared to JDK1.8 CODEC, Apache Commons CODEC is Easy to write, light weight(taken less memory), works faster.

-----Example Code using JDK1.8 CODEC-----

```
package in.nit.test;

import java.util.Base64;

public class TestCodec1 {

 public static void main(String[] args) {
 //1. Take One input String
 String india="Naresh I Technologies";

 //2.Convert to byte[] format
 byte[] characters=india.getBytes();

 //3.Give data to Encoder
 Base64.Encoder encode=Base64.getEncoder();
```

```
//4.Encode Data
byte[] data=encode.encode(characters);

//5.Covert Encode Data to String
String encode_Data=new String(data);

//6.Print Data
System.out.println(encode_Data);

//-----
-----//

//6.Read encoded String and convert to byte[]
byte[] decode=encode_Data.getBytes();

//7.Decode byte[] given
Base64.Decoder decoded_String=Base64.getDecoder();

//8.Convert to byte[]
byte[] decode_Data=decoded_String.decode(decode);

//9.Convert back to String Data Type
String orignal_String=new String(decode_Data);

//10.Print Decoded Data
System.out.println(orignal_String);
}

}
```

Output::

TmFyZXNoIEkgVGVjaG5vbG9naWVz  
Naresh I Technologies

---

-----Short Code-----

package in.nit.test;

```
import java.util.Base64;

public class EnCoDecTest {

 public static void main(String[] args) {
 String message_name="Welcome To All!";
 byte[]
encode=Base64.getEncoder().encode(message_name.getBytes());
 String data=new String(encode);
 System.out.println(data);
 //-----

 byte[]
decode=Base64.getDecoder().decode(data.getBytes());
 String data1=new String(decode);
 System.out.println(data1);

 }
}
```

Output::

---

V2VsY29tZSBubyBBbGwh  
Welcome To All!

---

## 4. Form Params:---action="\_\_\_\_\_";

---

Form Params in JAX-RS(RestFul Webservices)

---

- To send large data in hidden mode, using HTTP Body we can go for Form Parameters.
- It is also following key=value format, both are by default String type.

Here Data can be sent using:

1. HTML Form(also called as Physical Form)
2. Form class (also called as Logical Form)

→ \*\* Mostly used one in real time is Form class.

Notes:

1. It follows key=value format.
  2. It is mainly sent using POST Type. Use HTTP Body
  3. To read data at Producer use @FormParam.
- 

Case#1: EndUser to Producer Data Sending

Generally, Consumer application given data to Producer.

But this time EndUser wants to give data to Producer application.

In such case use FormParams.

Example::

Consumer App: BookMyshow

Producer App: PayTM

EndUser: Sam

While doing Payment using PayTM, SAM has chosen CC/DC  
Payment to add money to PayTM, in this case end user has to fill  
A HTML Form and submit.

i.e EndUser(SAM) is given data to Producer App(PayTM)

---

Case#2 Consumer sending Bulk Data to Producer

Consumer App wants to send multiple values as Input data to  
Producer App then Form Param concept is used.

\*\*Note:

Query Param and Matrix Param are sent using URL section  
even Header Params are sent using Header Section which has a  
limited data(max 4kb or 128 chars).

Form Params are sent using HTTP Request Body where as Body  
has no limit. So, Form Params can be unlimited(i.e more data/Bulk  
Data).

---

Form Params are sent to Producer in 2 different ways. Those are

1. Physical Form(HTML-UI/Browser)

EndUser sending data using HTML Form from Browser to  
Producer App.

2. Logical Form (Form class Object/Consumer App)  
Consumer Application is creating object using class Form  
And adding data to it. Finally sending to Producer App.

→ To read above type of Data in Producer App Syntax is:

`@FormParam("key")DataType LocalVariable`

-----Example1: Physical Form-----

Note: must use same dependencies Otherwise may be change not work.

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
```

```
</dependency>
```

```
</dependencies>
```

In web.xml File::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In ProducerController

```
package in.nit.controller;

import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

@Path("/payment")
public class ProducerController {

 @Path("/user")
 @POST
 public String message(
 @FormParam("u")String name
) {

 return "hello: "+name;
 }
}
```

}

In index.jsp(under WebContent or webapp).

```
<form action="rest/payment/user" method="post">
name:: <input type="text" name="u">
<input type="submit" value="Enter">
</form>
```

Output:

In chrome :

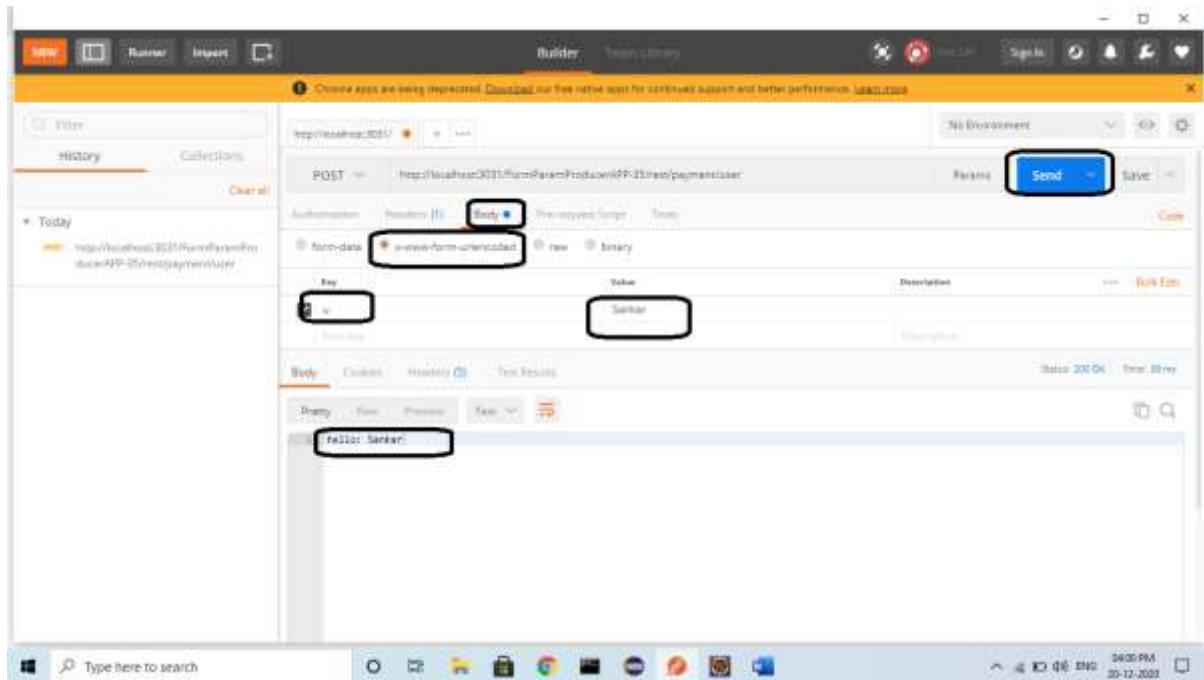
<http://localhost:3031/FormParamProducerAPP-35/rest/payment/user>

hello: Sankar

In POSTMAN TOOL:

- Here choose POST headers-> Content-type **application/x-www-form-urlencoded**
- Body -> key , value Send

The screenshot shows the Postman application window. In the top header, 'Builder' is selected. The main area shows a POST request to 'http://localhost:3031/FormParamProducerAPP-35/rest/payment/user'. The 'Headers' tab is selected, displaying a single header 'Content-Type: application/x-www-form-urlencoded'. The 'Body' tab shows a key-value pair 'hello: Sankar123'. The status bar at the bottom indicates a 'Status: 200 OK' response.



### -----Consumer App-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.33</version>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.33</version>
</dependency>

</dependencies>
```

In Test Class::

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.Response;

public class Testclass {
 public static void main(String[] args) {
 String URI="http://localhost:3031/FormParamProducerAPP-
35";
 String path="/rest/payment/user";
 try {
 //Creating Form Class object
 Form f=new Form();
 //adding dats to Form object
```

```
f.param("u", "ABC-123");

Client c=ClientBuilder.newClient();
WebTarget wt=c.target(URI).path(path);

Invocation.Builder builder=wt.request();
//sending data using Request Body
(Entity.form(formObj))

Response resp=builder.post(Entity.form(f));

System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));

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

}
```

Output::

---

```
200
OK
hello: ABC-123
```

Note:

1. Key=value can be sent in any order, Data binding is done based on key, not on order.

2. Sending extra key=value will be Ignored by FrontController(FC).
3. Auto Type conversion is supported by @FormParam.
4. If no key is provided in Request, FrontController(FC) returns default values.
5. Returns HTTP-404, If invalid DataType provided.

-----Example2: Physical Form-----

→ Producer App should read: cardNum, cardName, expMonth, expYear, CVV, amt by using Form Param.

In pom.xml file::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
```

```
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In web.xml::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
 </web-app>
```

In ProducerController:

```
package in.nit.controller;

import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/payment")
```

```
public class FormParamPaymentProducer {

 @Path("/card")
 @POST //if using form must take POST
 public String doPayment(
 @FormParam("cardNum")String cardNumber,
 @FormParam("cardName")String cardHolderName,
 @FormParam("expMonth")String expMonth,
 @FormParam("expYear")int expYear,
 @FormParam("cvv")String cvv,
 @FormParam("amt")double amt
) {

 StringBuffer sb=new StringBuffer();

 String message=sb
 .append("Hello Mr/Mrs/Ms :
").append(cardHolderName)
 .append(" , Amount : ").append(amt)
 .append(" , is paid from your Card :
").append(cardNumber)
 .append(" , Having Details : CVV=").append(cvv)
 .append(" , Date :
").append(expMonth).append("/").append(expYear)
 .toString();

 return message;
 }
}
```

In index.jsp file(Under webapp folder or WebContext Folder)

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
 pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Payment</title>
```

```
</head>
<body>
<h1>Payment Screen Here!</h1>
<form action="rest/payment/card" method="POST">
<pre>
CARD NUMBER: <input type="text" name="cardNum"
placeholder="xxxx-xxxx-xxxx">
CARD HOLDER NAME: <input type="text" name="cardName"
placeholder="Enter your Name">
EXP MONTH: <select name="expMonth">
<option>JAN</option>
<option>FEB</option>
<option>MAR</option>
<option>APR</option>
<option>MAY</option>
<option>JUN</option>
<option>JUL</option>
<option>AUG</option>
<option>SEP</option>
<option>OCT</option>
<option>NOV</option>
<option>DEC</option>
</select>
EXP YEAR: <select name="expYear">
<option>2021</option>
<option>2022</option>
<option>2023</option>
<option>2024</option>
<option>2025</option>
<option>2026</option>
<option>2027</option>
<option>2028</option>
<option>2029</option>
</select>
CVV: <input type="text" name="cvv"
placeholder="***">
AMOUNT: <input type="text" name="amt"
placeholder="23.33">
<input type="submit" value="Pay Now..."/>
</pre>
</form>
</body>
</html>
```

-----Output-----

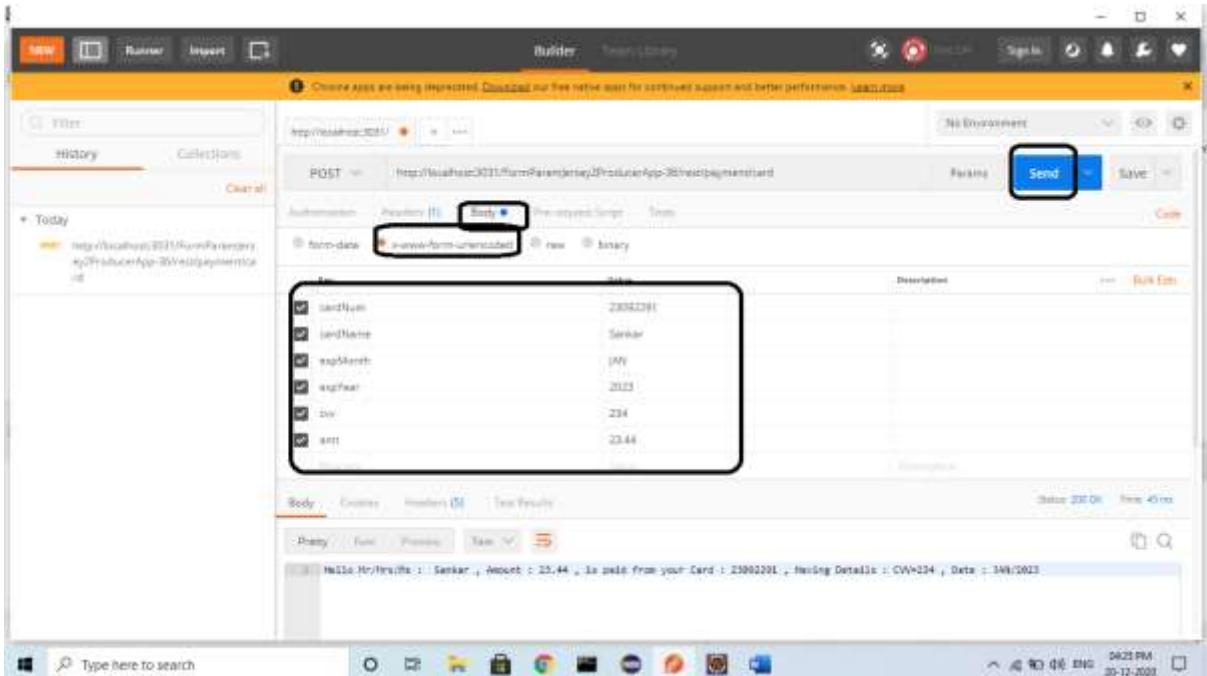
<http://localhost:3031/FormParamJersey2ProducerApp-36/rest/payment/card>

Hello Mr/Mrs/Ms : sankar , Amount : 12.0 , is paid from your Card :  
23092321 , Having Details : CVV=123 , Date : FEB/2024

-----POSTMAN SCREEN-----

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** <http://localhost:3031/FormParamJersey2ProducerApp-36/rest/payment/card>
- Headers:** Content-Type: application/x-www-form-urlencoded
- Body:** Hello: sankar
- Status:** 200 OK



---

### -----ConsumerJersey2.30\_App-----

Note: it is a Simple Maven APP

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
```

```
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

-----Logical Form-----

Consumer is sending data to Producer using Form Concept.

→ In Consumer Application add below steps extra

a) Create Form Class Object

```
Form f=new Form(); [javax.ws.rs.core package]
```

b) Add data to Form class Object

```
f.param(key,value);
```

c) Send Form data using Request Body with POST method

```
builder.post(Entity.form(f));
```

---

In ConsumerTest:

```
package in.nit.test;
```

```
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Form;
```

```
import javax.ws.rs.core.Response;

public class ConsumerJerseyConsumerTest {

 public static void main(String[] args) {
 String url="http://localhost:3031/FormParamJersey2ProducerApp-36";
 String path="/rest/payment/card";
 try {
 //Creating Form class Object
 Form f=new Form();

 //Adding DATA to Form Object
 f.param("cardNum","12334");
 f.param("cardName","Sankar");
 f.param("expMonth","JAN");
 f.param("cardYear","2023");
 f.param("cvv","123");
 f.param("amt","123.445");

 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(url).path(path);

 Invocation.Builder builder=wt.request();

 //Sending Data Using Request
 Body(Entity.form(form(Obj)))
 Response resp=builder.post(Entity.form(f));

 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 System.out.println(resp.readEntity(String.class));
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 }
}
```

Output::

200

OK

Hello Mr/Mrs/Ms : Sankar , Amount : 123.445 , is paid from your Card : 12334 , Having Details : CVV=123 , Date : JAN/0

---

-----Producer UserInfo App-----

It is a Web App:

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
```

```
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.30</version>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In ProducerUserInfo::

```
package in.nit.controller;

import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

@Path("/user")
public class UserInfoProducerController {

 @Path("/info")
 @POST
 public String readUserInfo(
 @FormParam("unqCode")String code,
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
@FormParam("secureId")String id,
@FormParam("accessToken")String token,
@FormParam("permitRole")String role
) {
 StringBuffer sb=new StringBuffer();
 String message=sb
 .append(" Code Is : ").append(code)
 .append(" , Secure ID IS : ").append(id)
 .append(" , Token Is : ").append(token)
 .append(" , Role : ").append(role)
 .toString();
 return message;
}
}
```

#### -----Output-----

In Chrome::

In chrome Not Possible. To see Output use postman tool.

#### -----PostMan Scree-----

In PostMan Tool Possible to see Output:

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

The screenshot shows the Postman application interface. A POST request is being prepared to the URL `http://localhost:3001/UserInfoProducer/app-17/testUserinfo`. In the Headers section, there is a single entry for `Content-Type` set to `application/x-www-form-urlencoded`. The `Send` button at the top right of the request panel is highlighted with a blue rectangle.

The screenshot shows the same Postman session after the `Send` button was clicked. The response pane now displays the JSON data sent in the request body. The response status is `200 OK` and the time taken is `101 ms`. The response content is: `Code Is : 123 , Secure ID IS : 357 , Toke Is : 1234, Role : Admin`.

Code Is : 123 , Secure ID IS : 357 , Toke Is : 1234, Role : Admin

-----Consumer App Jersey2.30-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

In UserInfoConsumerApp::

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
```

```
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.Response;

public class ConsumerJerseyConsumerTest {

 public static void main(String[] args) {
 String url="http://localhost:3031/UserInfoProducerApp-37";
 String path="/rest/user/info";
 try {
 //Creating Form class Object
 Form f=new Form();

 //Adding DATA to Form Object
 f.param("unqCode","12334");
 f.param("secureId","Sankar");
 f.param("accessToken","JAN");
 f.param("permitRole","2023");

 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(url).path(path);

 Invocation.Builder builder=wt.request();

 //Sending Data Using Request
 Body(Entity.form(form(Obj)))
 }
 }
}
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
Response resp=builder.post(Entity.form(f));

 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 System.out.println(resp.readEntity(String.class));
 }

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

Output::

```
200
OK
Code Is : 12334 , Secure ID IS : Sankar , Token Is : JAN, Role : 2023
```

---

-----Addition Producer App-----

It is web app:

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
```

```
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
 class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
```

```
<servlet-mapping>
< servlet-name > sample </servlet-name >
< url-pattern > /rest/* </url-pattern >
</servlet-mapping>
< welcome-file-list >
< welcome-file > input.jsp </welcome-file >
</ welcome-file-list >
</ web-app >
```

In Producer App:

```
package in.nit.controller;

import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

@Path("/math")
public class UserInfoProducerController {

 @Path("/add")
 @POST
 public String readUserInfo(
 @FormParam("a") Integer a,
 @FormParam("b") Integer b,
 @FormParam("c") Integer c
) {
 Integer total = a + b + c;
 StringBuffer sb = new StringBuffer();
 String message = sb
```

```
 .append(" A : ").append(a)
 .append(" , B : ").append(b)
 .append(" , C : ").append(c)
 .append(", Total = ").append(total)
 .toString();

 return message;
}

}
```

In input.jsp: ( Under WebContext Folder)

Note: Tomcat9 server automatically detects index.html or index.jsp

If you changed file to like input.jsp you must add entries in web.xml

```
<welcome-file-list>
 <welcome-file>input.jsp</welcome-file>
</welcome-file-list>
```

or you maual type /input.jsp

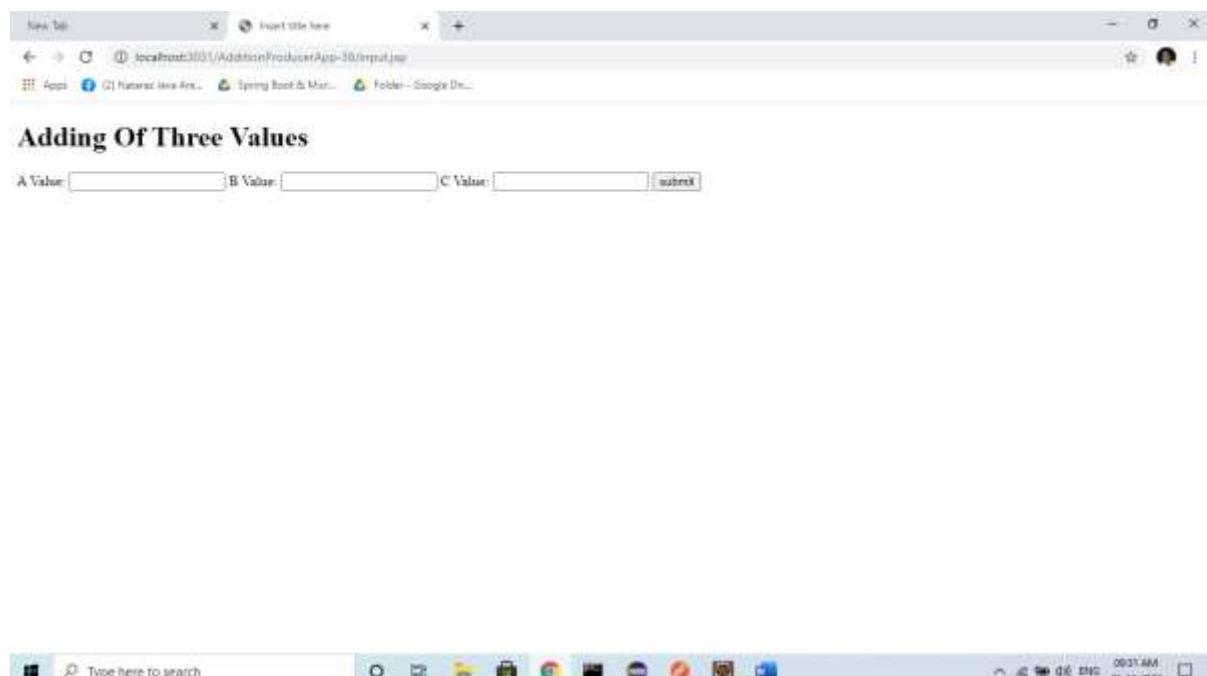
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Adding Of Three Values</h1>
<form action="rest/math/add" method="post">
A Value: <input type="text" name="a">
B Value: <input type="text" name="b">
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
C Value: <input type="text" name="c">
<input type="submit" value="submit">
</form>
</body>
</html>
```

In Chrome:

Input.jsp File:

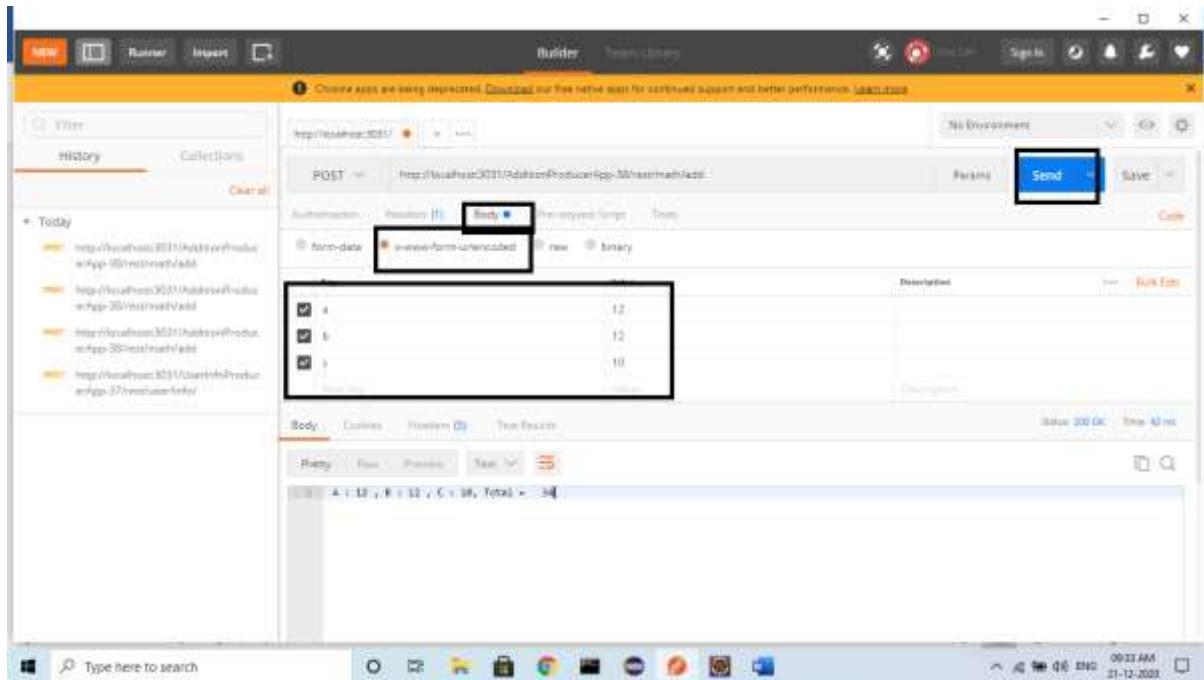


For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>



## -----PostMan Tool-----

A screenshot of the Postman tool interface. The top navigation bar includes "NEW", "Runner", "Import", "Collection", "Builder", "Team Library", and "Sign In". The main area shows a "POST" request to "http://localhost:3031/AdditionProducer" with the "Headers" tab selected. A "Content-Type" header is set to "application/x-www-form-urlencoded". The "Body" tab shows a raw JSON payload: {"A":12,"B":12,"C":12,"Total":36}. The status bar at the bottom shows "Status: 200 OK | Time: 41 ms".




---

A : 12 , B : 12 , C : 10, Total = 34

---

-----Consumer App-----

```

<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>

```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In Consumer Test:

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.Response;

public class ConsumerJerseyConsumerTest {

 public static void main(String[] args) {
 String url="http://localhost:3031/AdditionProducerApp-38";
 String path="/rest/math/add";
 try {
 //Creating Form class Object
```

```
Form f=new Form();

//Adding DATA to Form Object
f.param("a","60");
f.param("b","60");
f.param("c","60");

Client c=ClientBuilder.newClient();
WebTarget wt=c.target(url).path(path);

Invocation.Builder builder=wt.request();

//Sending Data Using Request
Body(Entity.form(form(Obj))
Response resp=builder.post(Entity.form(f));

System.out.println(resp.getStatus());
System.out.println(resp.getStatusInfo());
System.out.println(resp.readEntity(String.class));
}

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

-----Output-----
```

OK

A : 60 , B : 60 , C : 60, Total = 180

---

Assignment(Token6):: you can try

Token6:

Write Jax-RS(Jersey2.30) Producer

Module name: addition

method name: addOfthree

Calculations: total=a+b+c;

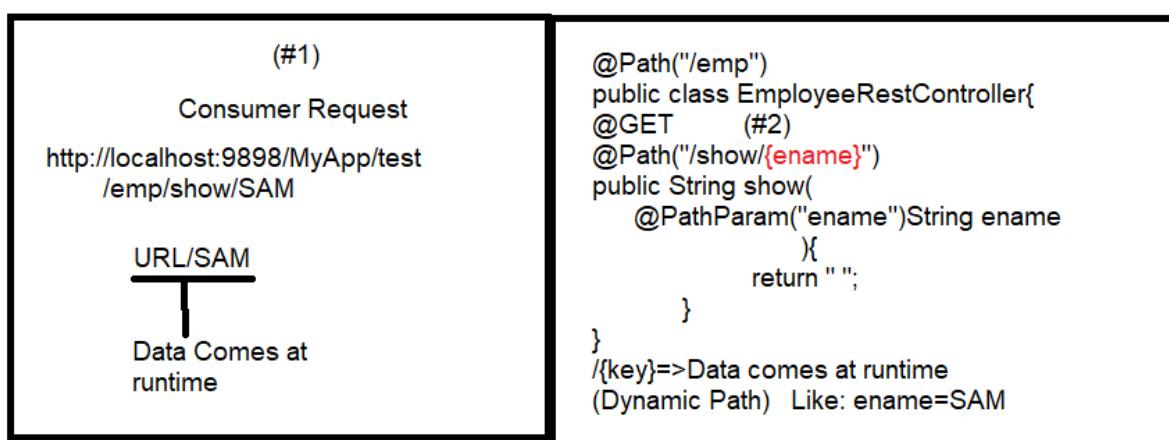
Result like: A value: <a> B Value: <b> C Value: <c> Total is: <total>

Example: A value: 10 B Value: 10 C Value: 10 Total is: 30

Hint:use @FormParam

Test in three Browser and PostMan Tool and Consumer App

---



## 5.Path Parameters

# /Template/ Path Variable

-> Here Path indicates URI Pattern given to one resource or a unique name/identity given to resource.

- \*\* Here Resource means, It can folder/file/project/data/code files
- Sending primitive data from Consumer to Producer can be done along with Path( as data ) is known as PathVariable, which works faster than all other Parameters (i.e @QueryParam,@MatrixParam,@FormParam,@HeaderParam).
- It is also called as "PathParameter in ReST.

Example1:

<http://localhost:8080/App/rest/msg/show/10/Sankar/10.5>

Here 10/Sankar/10.5 are Path Parameters.

- Consider one Example2 URL as given below

<http://localhost:9090/MyApp/rest/user/show>

/MyApp -is a PATH given to project

/rest -is a PATH given to FC(Servlet)

/user -is a PATH given to RestController class

/show -is a PATH given to method in RestController

Path:

Path Indicates Identity of a resource which can be a class or a method in Programming. We can provide multilevel Path in programming.

Example: @Path("/emp/contract")

```
public class EmpContract {
 @Path("/delete/emp/byId")
 public void show(){

 }
}
```

Here "/emp/contract" provided at class level & "/delete/emp/byId" provided at method level are Multilevel Paths.

This Path can be Static/Dynamic.

Dynamic Path is used to send Data (path parameter).

Dynamic Path:

To send a value along with URL without any key(Only value)  
Dynamic path is Used.

It can be specified using symbol {}.

Example: /{empId}

If a Request is made then highest priority should be given to Static Path.

If no matching found then go to Dynamic Path.

@) Why Path?

A) Consumer App will access our Producer App resource  
(files/codes) by using unique Path.

---

## Types Of Path in Programming:

→ There are two types of PATHs. They are given as:

- a) static Path(normal PATH): It indicates a unique name/URL pattern given to resource.
- b) \*\* Dynamic Path: It behaves like PATH but used to transfer data from Consumer App to Producer App.  
Syntax: at RestController Level: /{key}

Example:

<http://localhost:8080/MyApp/rest/user/show/10>

/user -class level path  
/show -method level path  
/10 -Behaves like path, but 10 is data given to  
RestController.(/user/{num})

→ \*\* Sending data using Path from consumer to Producer is called as  
@PathParam.

- ➔ @PathParam is faster compared to Query/Matrix params.
- ➔ (#1) Data sending Syntax: URL/value/value.....
- ➔ (#2) Method level Path Specification: @Path("/url/{key}")
- ➔ (#3) Data Reading Syntax: @PathParam("key")Data Type Local Variable.

-----Example#1 Producer Code-----

Note: it is web App

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
```

</dependencies>

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/* </url-pattern>
 </servlet-mapping>
</web-app>
```

In PathParamProducerController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;

@Path("/emp")
public class PathParamProducer {

 @GET
 @Path("/data/{name}")
 public String show(
 @PathParam("name")String name
) {
```

```
 return "Name is : "+name;
 }
}
```

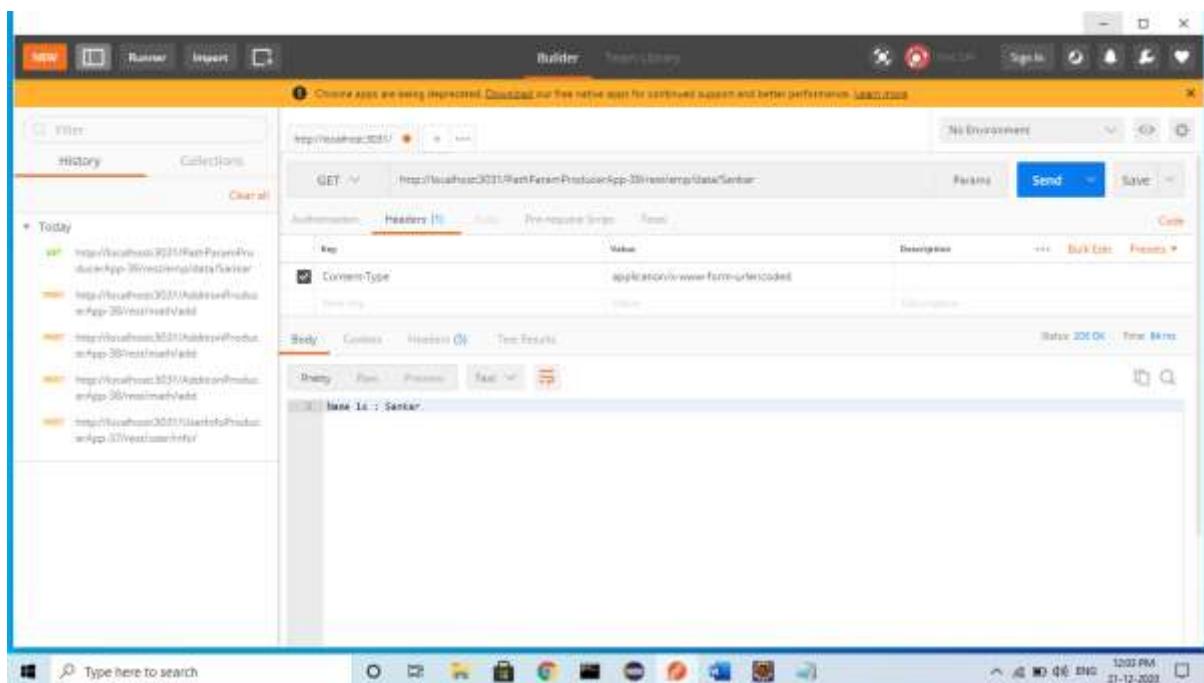
-----OutPut-----

In Chrome:

<http://localhost:3031/PathParamProducerApp-39/rest/emp/data/Sankar>

Name is : Sankar

-----POSTMAN TOOL-----



- If we are sending multiple values using @PathParam,  
Then order must be followed while Sending Data.
  - Because, in request URL there will be no key, Path contains only data.
  - While receiving data in Code it contains key.
- 
- \*\*\* Request URL must have all levels of data that matches with Path.

Example: URL=> /a/b/c/d/e (5 Levels are there).  
Then Request URL should have Data for 5 Levels.

-----Multiple @PathParam Producer App-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
```

```
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### In MultiplePathParamController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;

@Path("/emp")
public class PathParamProducer {

 @GET
 @Path("/data/{eid}/{ename}/{esal}")
 public String fetchAll(
 @PathParam("eid")int eid,
 @PathParam("ename")String ename,
 @PathParam("esal")double esal
) {

 return " Employee id: "+eid+"Name is:"+ename+" Salary
is : "+esal;
 }
}
```

-----Output-----

In Chrome(or any Browser)

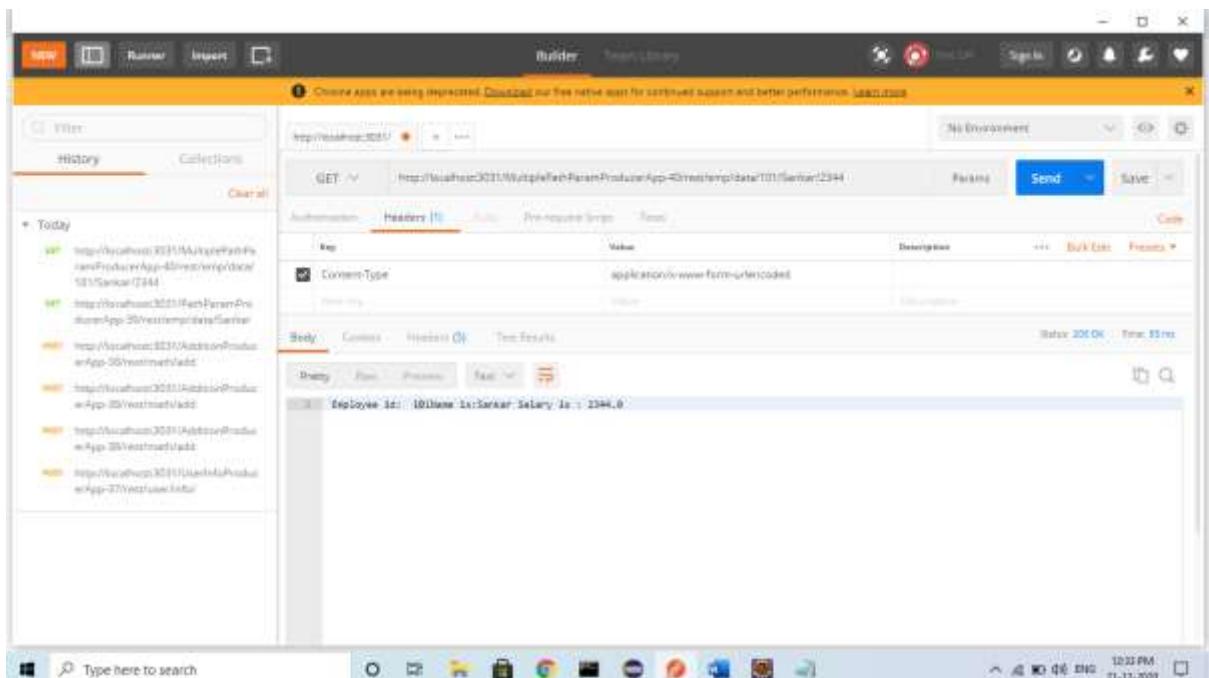
<http://localhost:3031/MultiplePathParamProducerApp-40/rest/emp/data/101/Sankar/2344>

Employee id: 101Name is:Sankar Salary is : 2344.0

Note: pass params exactly .Otherwise you get 404-Error.

-----Error if give any other URL 404-----

<http://localhost:3031/MultiplePathParamProducerApp-40/rest/emp/data/eid/ename/esal/101/Sankar/2344>  
404-Not Found.



-----EmployeeProducerPathParam-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
```

```
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
 class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
```

```
< servlet-name > sample </ servlet-name >
< url-pattern > /rest/* </ url-pattern >
</ servlet-mapping >
</ web-app >
```

In EmployeePathParamProducer:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rsPathParam;

@Path("/emp")
public class PathParamProducer {

 @GET
 @Path("/data/code")
 public String showA() {
 return " From All Static Data.... ";
 }
 @GET
 @Path("/data/{code}")
 public String showB(
 @PathParam("code")String code
) {
 return " From Dynamic code is :" + code;
 }
}
```

-----Output-----

In chrome1:

<http://localhost:3031/MultiplePathParamProducerApp-40/rest/emp/data/code>

From All Static Data....

In chrome2:

<http://localhost:3031/MultiplePathParamProducerApp-40/rest/emp/data/Sankar>

From Dynamic code is :Sankar

---

Q) What is order of writing static and dynamic paths in code?

A) There is no order required to write static and dynamic paths in code.  
Those can be in any Order.

-----PathParamProducer App-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
```

```
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In web.xml File:::

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
 </web-app>
```

In ProducerController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rsPathParam;

@Path("/emp")
public class PathParamProducer {

 @GET
 @Path("/data/{data}/yy/{name}")
 public String showB(
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
@PathParam("data")String data,
@PathParam("name")String name
) {

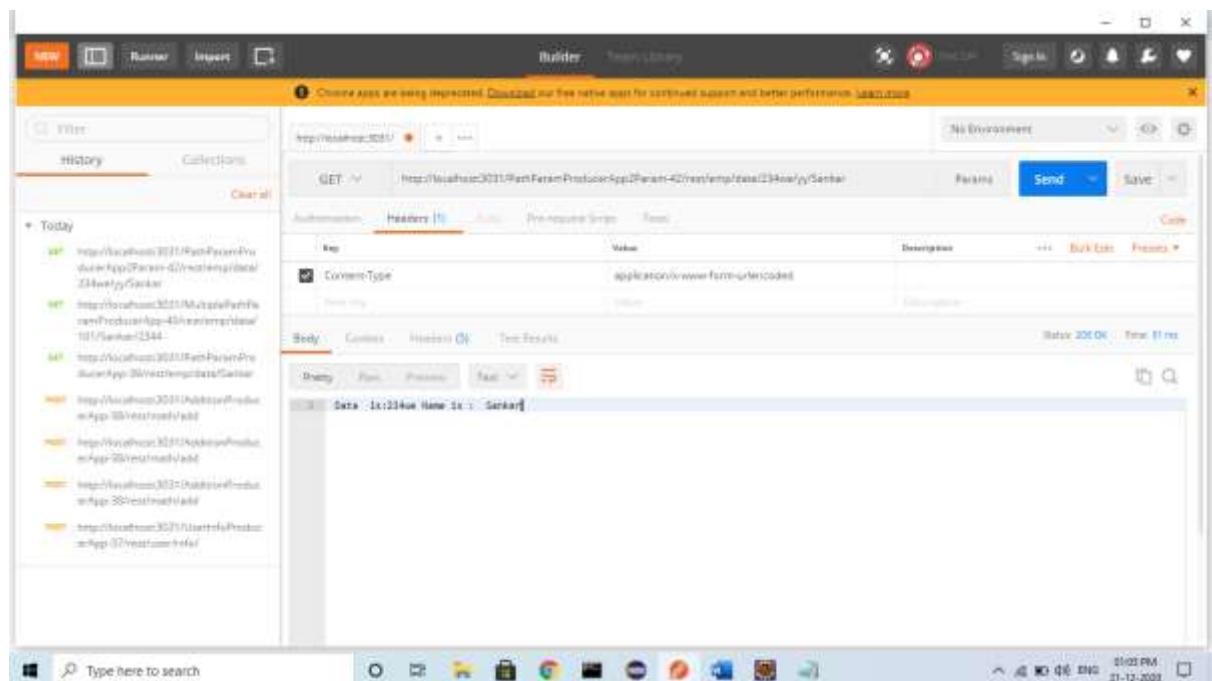
 return " Data is:"+data+" Name is : "+name;
}
}
```

-----Output-----

<http://localhost:3031/PathParamProducerApp2Param-42/rest/emp/data/234we/yy/Sankar>

Data is:234we Name is : Sankar

-----POSTMAN TOOL-----



The screenshot shows the Postman application window. In the top bar, there are tabs for 'Builder', 'Collection', 'Library', and 'Sign In'. Below the tabs, the URL is set to 'http://localhost:3031/PathParamProducerApp2Param-42/rest/emp/data/234we/yy/Sankar'. The 'Headers' tab is selected, showing a single header 'Content-Type: application/x-www-form-urlencoded'. The 'Body' tab shows the response body: 'Data : 234we Name is : Sankar'. The status bar at the bottom right indicates 'Status: 200 OK' and 'Time: 81 ms'.

-----PathParamConsumerApp-----

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

## In ConsumerTest

```
package in.nit.test;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class ConsumerTest {

 public static void main(String[] args) {
 String url="http://localhost:3031/PathParamProducerApp2Param-
42";
```

```
String path="/rest/emp/data/234we/yy/Sankar";
try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(url).path(path);

 Invocation.Builder builder=wt.request();
 Response resp=builder.get();

 System.out.println(resp.getStatus());
 System.out.println(resp.getStatusInfo());
 System.out.println(resp.readEntity(String.class));

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

}
```

-----Output-----

200  
OK  
Data is:234we Name is : Sankar

---

| Methods | Paths                 |
|---------|-----------------------|
| M1()    | /data/find/info       |
| M2()    | /data/find/{info}     |
| M3()    | /data/{find}/{info}   |
| M4()    | /{data}/{find}/info   |
| M5()    | /{data}/find/{info}   |
| M6()    | /{data}/{find}/{info} |

-----Request URLs-----

#1 ...../data/55/20

Matching Methods: m3(), m6().  
Finally selected is: m3().

#2. ....../ABC/find/info

Matching Methods: m4(), m5(), m6()

Finally Selected is: m5().

---

M1() -----/abc/xyz/mno  
M2()-----/abc/xyz/{mno}  
M3()-----/abc/{xyz}/{mno}  
M4()-----/{abc}/{xyz}/{mno}

- One Request URL can be matched with multiple methods, but priority for execution is given for a greater number of static paths.

-----Request URLs-----

1..../abc/25/mno  
2..../ABC/XYZ/MNO  
3..../hello/hi/mno  
4..../abc/xyz/mno  
5..../abc/XYZ/mno

-----PathParamProducerApp-----

-> In pom.xml File:

---

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In web.xml File:

---

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In PathParamRestController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rsPathParam;

@Path("/emp")
public class PathParamRestController {

 @GET
 @Path("/msg/empId/empName/empSal")
 public String show() {

 return "Hello#1: ";
 }

 @GET
 @Path("/msg/empId/empName/{empSal}")
 public String show2(@PathParam("empSal")Double empSal) {

 return "Hello#2: "+empSal;
 }

 @GET
 @Path("/msg/empId/{empName}/{empSal}")
 public String show3(
 @PathParam("empName")String empName,
 @PathParam("empSal")Double empSal) {

 return "Hello#3: "+empName+" , "+empSal;
 }
}
```

-----Output-----

In chrome:

<http://localhost:3031/PathParamProducerApp-67/rest/emp/msg/empId/empName/empSal>

Hello#1:

<http://localhost:3031/PathParamProducerApp-67/rest/emp/msg/empId/empName/23.00>

Hello#2: 23.0

<http://localhost:3031/PathParamProducerApp-67/rest/emp/msg/empId/Sankar/23.00>

Hello#3: Sankar , 23.0

-----POSTMAN SCREEN-----

The screenshot shows the Postman interface with a GET request to `http://localhost:3031/PathParamProducerApp-67/rest/emp/msg/empId/Sankar/23.00`. The response body contains the text `Hello#3: Sankar , 23.0`.

Note:

1. Path Param follows order without Key.
  2. If extra/less values are Sent or Invalid data sent then http-404.
  3. Highest Priority is given to static.
  4. No default value concept is provided
  5. To read data, use @PaathParam Annotation.
- 

- \* Consumer has to provide to Producer Application as Inputs using Parameters.
- \* If Consumer is not sending data, at producer side FrontController(FC) will provide default values.
- Like int-0, double-0.0, String-null.....etc.
- At Producer code level use **@DefaultValue** annotation to modify default values for Parameters that are given by FrontController(FC),  
If data not given by Consumer.

-----Example ProdcuerRestController-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In web.xml:

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/* </url-pattern>
 </servlet-mapping>
</web-app>
```

In DefaultValueProducerController:

```
package in.nit.controller;

import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

@Path("/info")
public class DefaultValueProducer {

 @GET
 public String readData(
 @DefaultValue("10") @QueryParam("id")int id,
 @DefaultValue(" No Name")
 @QueryParam("name")String name,
 @QueryParam("fees")double fees
) {

 return "Data Given is: => "+id+"-"+name+"-"+fees;
 }
}
```

-----output-----

In chrome:

<http://localhost:3031/DefaultValueProducerApp2Param-43/rest/info?id=101&name=Sankar&fees=23.44>

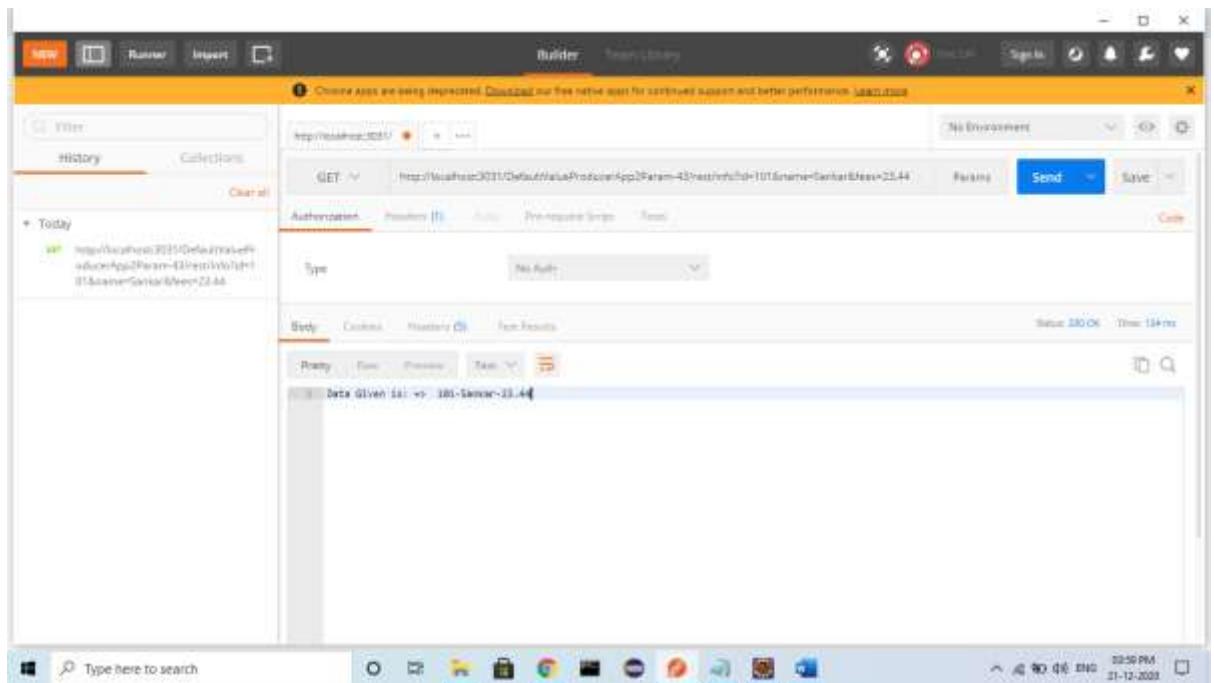
Data Given is: => 101- No Name-23.44

→ Now I am given wrong key name to names then getting default Value:

<http://localhost:3031/DefaultValueProducerApp2Param-43/rest/info?id=101&names=Sankar&fees=23.44>

Data Given is: => 101- No Name-23.44

-----PostMan Screen-----



- \*\* Note: Default values we can also use for other params even like  
 @DefaultValue("NO NAME") @MatrixParam("cname")String cname;

### Assignment(Token7)::

- \* Define one JAX-RS application using Module Product.  
 It should have operation getBillDetails().  
 It takes 4parms, productId, code, cost and discount(as percentage).
- \*\* If Discount is 0 then default discount as 3%.

-----Product DefaultValue Producer App-----

Note: It is Web App:

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
```

```
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In ProductDefaultValueProducerController:

```
package in.nit.controller;
```

```
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.MatrixParam;
import javax.ws.rs.Path;

@Path("/product")
public class DefaultValueProducer {

 @GET
 @Path("/invoice")
 public String getBillDetails(
 @MatrixParam("pid")int pid,
 @MatrixParam("PCODE")String pcode,
 @MatrixParam("PCOST")double amt,
 @DefaultValue("3") @MatrixParam("DISCOUNT")int
discount
) {

 double discountAmt=amt*discount/100.0;
 double finalAmt=amt-discountAmt;
 return new StringBuffer()
 .append("For Product: ").append(pcode)
 .append(" , Having Id: ").append(pid)
 .append(" , Actual Amount: ").append(amt)
 .append(" , Discount Amount:
").append(discountAmt)
 .append(" , Final Cost Is: ").append(finalAmt)
 }
}
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
 .toString();
 }
}
```

//@MatrixParam();

-----Output-----

In chrome:

<http://localhost:3031/DefaultValueProducerApp2Param-43/rest/product/invoice;pid=101;pcode=2we333;pcost=34>

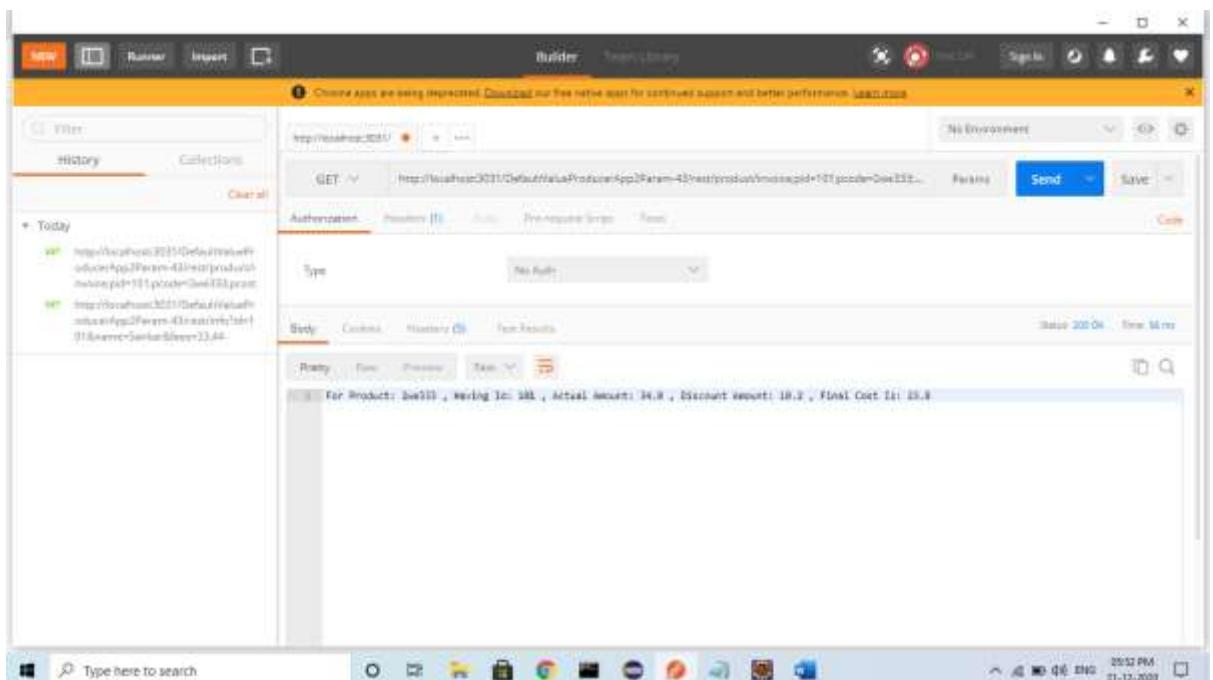
For Product: 2we333 , Having Id: 101 , Actual Amount: 34.0 , Discount Amount: 1.02 , Final Cost Is: 32.98

Or

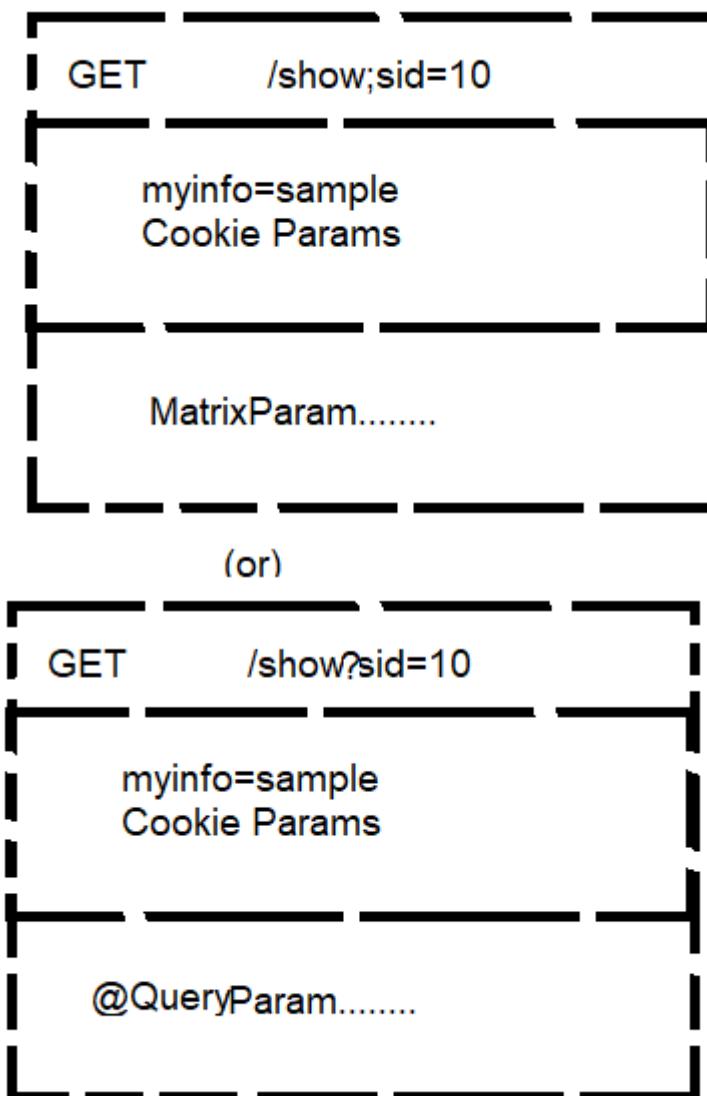
<http://localhost:3031/DefaultValueProducerApp2Param-43/rest/product/invoice;pid=101;pcode=2we333;pcost=34;discount=30>

For Product: 2we333 , Having Id: 101 , Actual Amount: 34.0 , Discount Amount: 10.2 , Final Cost Is: 23.8

-----POSTMAN SCREEN-----



- Q) Can we use mixed params in JAX-RS (Restful webservices)?  
A) Yes, It possible, but limitations is  
    -> We can pass/use at a time either @QueryParam or  
        @MatrixParam  
    -> Because both uses same section (i.e HTTP Head Initialization Line)



-----mixed Params ProducerApp-----

Note: It is WebApp

In pom.xml File:

<properties>

```
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
```

```
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In MixedProducerController:

```
package in.nit.controller;
import javax.ws.rs.FormParam;
import javax.ws.rs.HeaderParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

@Path("/info")
public class MixedParamProducer {
 @POST
 //Consumes("application/x-www-form-urlencoded") optional
 @Consumes("application/x-www-form-urlencoded")
 public String readData(
 @QueryParam("id")int id,
 @HeaderParam("name")String name,
 @FormParam("fee")double fee
) {

 return "Data Given Is => : "+id+"-"+name+"-"+fee;
 }
}
```

Note::

Content-type: application/x-www-form-urlencoded copy from Postman tool

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'New', 'Runner', 'Import', and other icons. Below it, a banner says 'Create apps using improved Dashboard or the new app for continued support and better performance. [Learn more](#)'.

The main area has a sidebar on the left with 'History' and 'Collections' sections, and a 'Create API' button. The 'APIs' collection is selected, listing two items:

- <http://localhost:3010/Default/HelloWorld>
- <http://localhost:3010/Default/HelloWorld?param=123>

For the first item, the 'Headers' tab is active, showing a single header:

| Key          | Value                             | Description |
|--------------|-----------------------------------|-------------|
| Content-Type | application/x-www-form-urlencoded |             |

Below the table, there are 'Body', 'Cookies', 'Headers (2)', and 'Test Results' tabs. On the right, there are 'Send', 'Save', and 'Cancel' buttons.

## -Output-

In chrome:

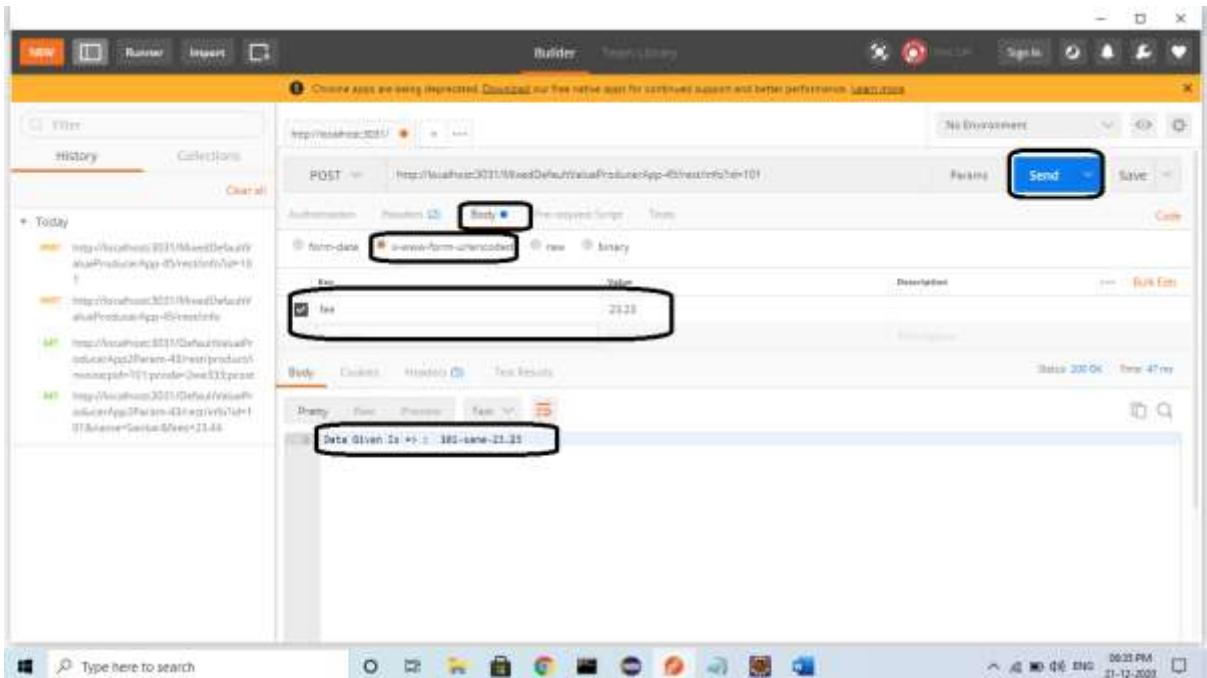
Not possible to see Output here.

-POSTMAN TOOL

The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Runner', 'Import', 'Builder', 'Recent Library', 'Find API', 'Sign In', and a user icon. A banner at the top says 'Choose API and API endpoint. Download our free native app for continued support and better performance.' with a 'GET IT' button. The left sidebar has 'History' selected under 'Collections' and shows a list of recent requests:

- http://localhost:8081/WoodDefault/ValueProducerApp-0/test/info?n=101
- http://localhost:8081/WoodDefault/ValueProducerApp-0/test/info
- http://localhost:8081/Gateway/wood/producer/api?param=101&providerName=S333&producerName=S333&producerName=S333
- http://localhost:8081/Gateway/wood/producer/api?param=101&providerName=S333&producerName=S333&producerName=S333

The main workspace shows a POST request to 'http://localhost:8081/WoodDefault/ValueProducerApp-0/test/info?n=101'. The 'Headers' tab is selected, showing two entries: 'Content-Type' with value 'application/x-www-form-urlencoded' and 'name' with value 'Jane'. The 'Body' tab is also visible. On the right, there are buttons for 'Send', 'Save', 'Cancel', 'No Environment', 'Params', 'Bulk Edit', and 'Pretty'.



Data Given Is => : 101-sane-23.23

Note:

- >Content-Type default is: application/json (default)
- >a resource and then passed back to the client is specified as a MIME media type in the headers of an HTTP request or response
- > Context type passing in Producer App option.
- >Which indicates which type of data Providing from Request to Response.
- > using the POST method, then its data is sent as part of the body of the HTTP request.
- >@FormParam and @headerParam not possible to send in chrome.

-----Default Parm ProducerApp-----

It is web App::

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
 class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
```

```
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In ProducerController:

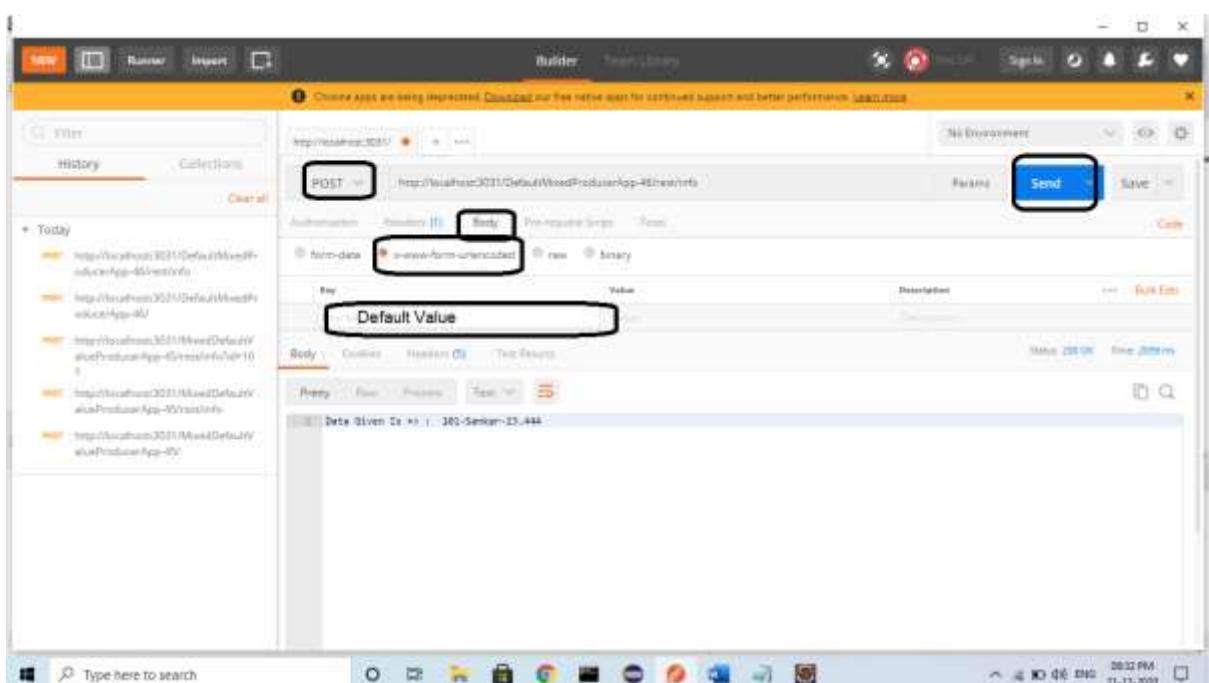
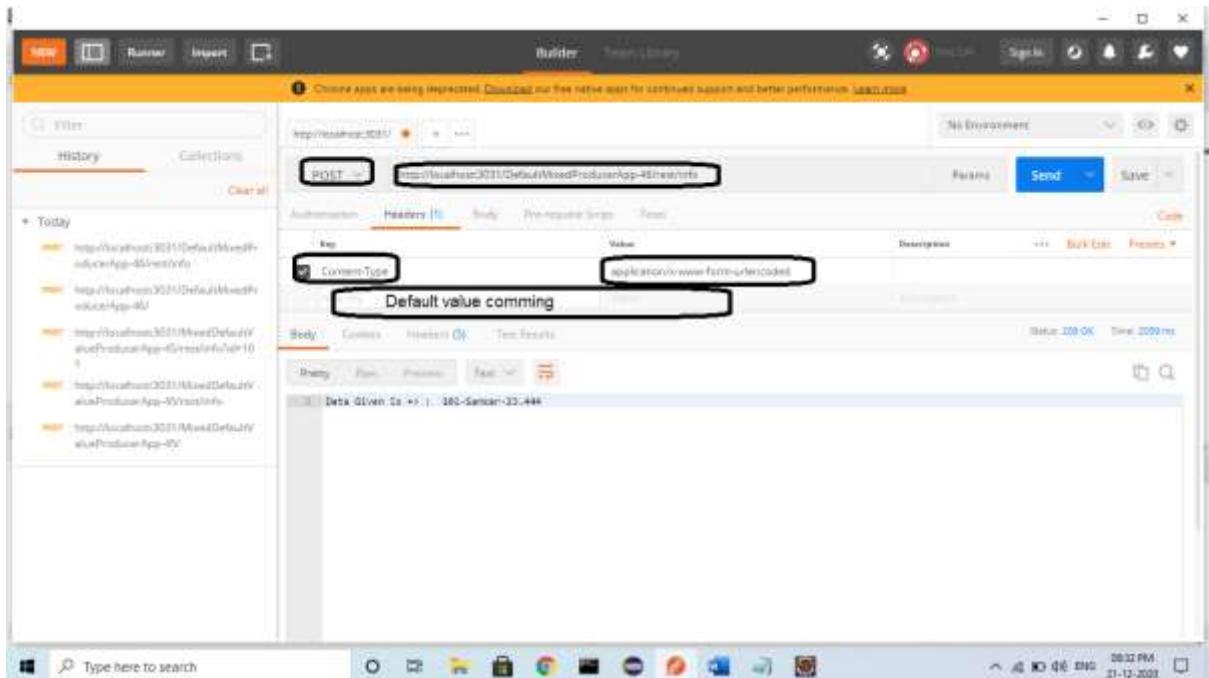
```
package in.nit.controller;
import javax.ws.rs.Consumes;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.FormParam;
import javax.ws.rs.HeaderParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

@Path("/info")
public class MixedParamProducer {
 @POST
 // @Consumes("application/x-www-form-urlencoded") optional
 @Consumes("application/x-www-form-urlencoded")
 public String readData(
 @DefaultValue("101") @QueryParam("id") int id,
 @DefaultValue("Sankar")
 @HeaderParam("name") String name,
 @DefaultValue("23.444") @FormParam("fee") double fee
) {
 return "Data Given Is => : "+id+"-"+name+"-"+fee;
 }
}
```

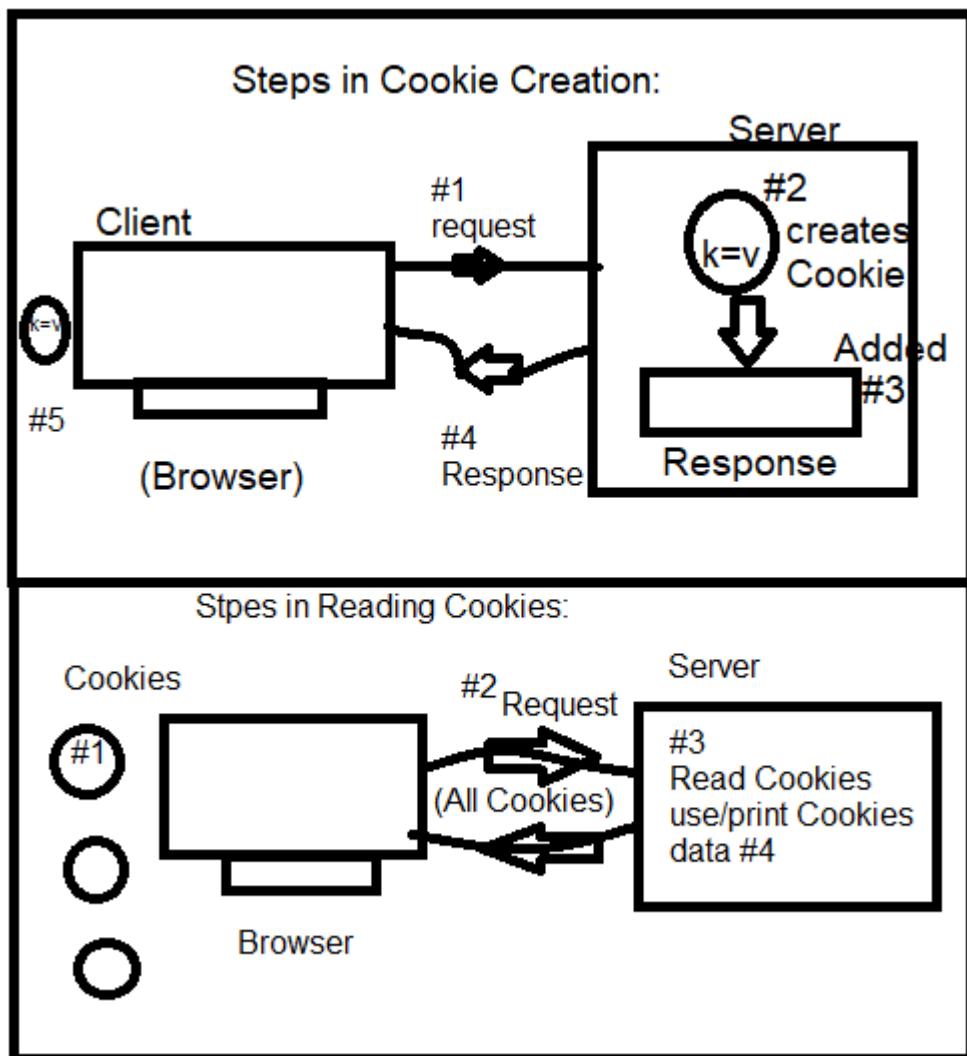
-----Output-----

In chrome not possible to see output.

-----POSTMAN TOOL-----



## 6.Cookie:



- It is a memory that holds data in key=value format which is stored at browser.
- If Consumer App or Producer App wants to store data for some time at browser, then Cookies Params are used.
- It is like EndUser Specific data.

Example: langCode=HINDI, loc=HYD.

#### -----Working on Cookies in JAX-RS-----

1. Creating Cookie and sending to Client Machine
  - a. Client Machine(browser) will make request to Server
  - b. Server will create one new Cookie with Key=Value.
  - c. Server will add Cookie to Response
  - d. Response is sent to Client Machine
  - e. Client will store Cookie.
- \*\*\*\* Cookies are created by Servers and send browsers where they are stored.
2. Reading Cookie from Client Machine

- a. Client Machine holds Cookies given by Server.
- b. When Client is making request even Cookies also sent to Server.
- c. Server will read Cookies.
- d. Server can use/Print Cookies Data.

-----Syntax for Cookie In JAX-RS-----

1. Creating Cookie in JAX-RS(ReSTful webservices)

By using class: NewCookie

Define in package: javax.ws.rs.core

Having constructor: NewCookie(name,value)

Example:

```
NewCookie c1=new NewCookie("LangCode","ENG");
```

→ To send Cookie to Browser, It must be added to Response.

Use method cookie(newCookie...cks) to add Cookies to Response.

```
Response.status(-).entity(_).cookie(c1,c2,c3,.....).build();
```

---

2. Reading Cookie in RestController, by using Annotation  
@CookieParam

Example:

```
@cookieParam("key")DataType LocalVariable
```

---

-----Cookies in Servlet-----

-> Cookie is a memory created by server and stored in Browser.

-> Cookies stores data (at max 4KB) only text data.

->Cookies allocate memory at client.

->Cookies are Specific to Browser, if you are fill form data in one browser to see those Cookies in another Browser not possible.

-> A cookie created by server will be send to browser using HTTP Response Head Area.

->All Cookies will be submitted along with your request using HTTP Request Head.

->Cookie store data in key=value format. Those are also called as Attribute Name and Attribute Value.

->Every Cookie will be having max life time (expiry time). Default is-1 indicates on browser close all Cookies are removed from browser Cache.

->We/Programmer can set life time of a Cookie using method setMaxAge in sec (int value) values.

Example: 60sec=1min.

->A Cookie created by Server one can only read/modify/deleted by same Server one only. not by default other servers.

->remembering client data at client side across multiple requests during session.

-> it is Statefull/State management.

->Statefull: Storing Client related data at server for long time (Login=Logout time) is known as Session.

---

### Steps to Create Cookie

---

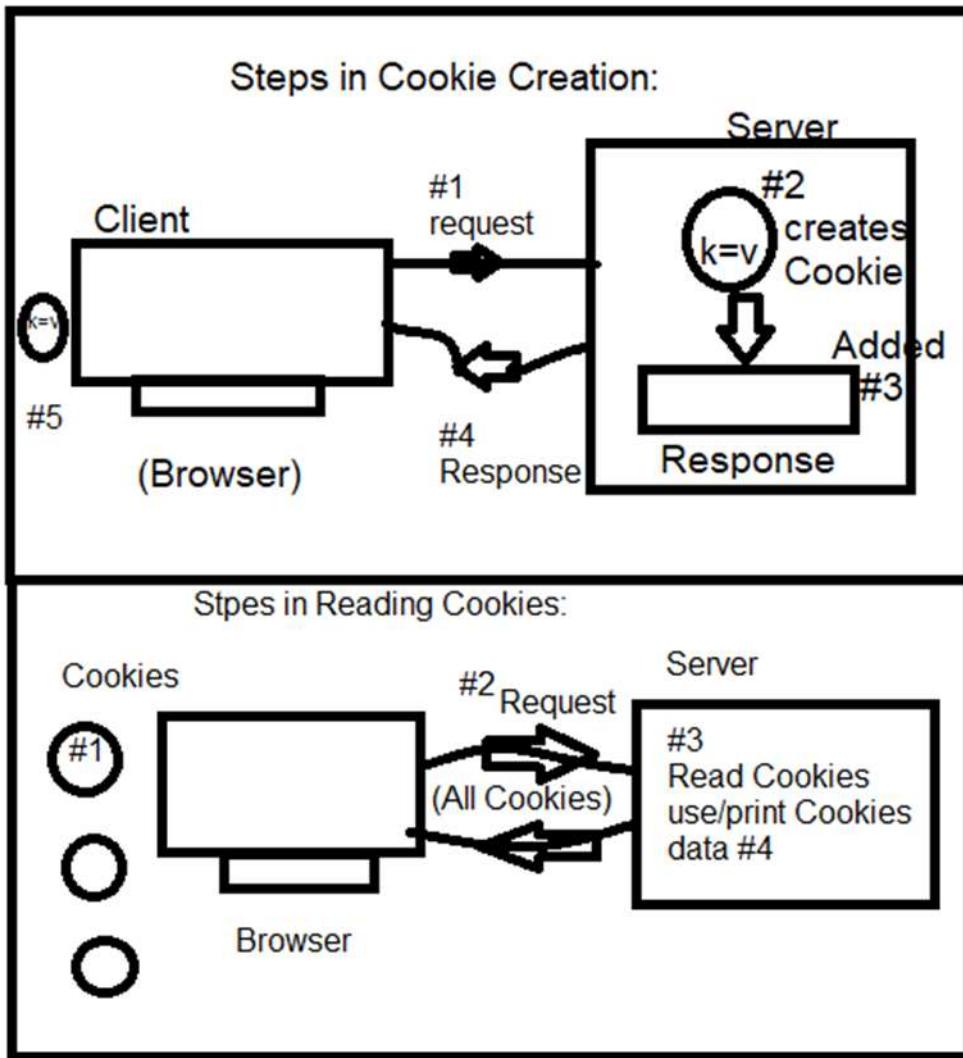
1. Make New/First Request using Browser.
2. Server creates new Cookie and adds to HTTP Response (Head).
3. Response send back to the Browser.
4. Browser creates all new Cookies in Browser Cache.
5. Browser submits all Cookies related to same Server using HTTP Request Head.

---

### Limitations-

---

- In Browser we can disable Cookie concept, then application may not work properly.
- Stores only Text data, not binary data like image, Audio, Video...etc.
- Secure data cannot be stored in Cookies.



### Working on Cookie

1. To create one Cookie, use class `Cookie` given by Servlet HTTP API in package “`javax/jakarta.servlet.http`.” use parameterized Constructor.

```
Cooke c1=new Cookie((name,value));
```

- Name and value both are String type only.
- Set Lifetime of Cookie using method `setMaxAge(int);` in seconds.

2. Send Cookie from Server to Browser using HTTP Response Head.  
`response.addCookie(c1);`
3. On request made, Browser submits all Cookies to server using HTTP Request Head. To read those codes is:

```
Cookie[] cks=request.getCookies();
```

4. To read one Cookie data use index number And methods like  
`getName(),cks[0].getName();cks[0].getValue();`

#### -----Types of Cookies-----

1. SessionCookies/Browsing Session cookies:

- A Cookie which do not have expiration As soon as then web Browser is closed this Cookie gets Destroyed.
- 2. Persistent Cookie:
- A Cookie is created with life -time and destroyed based on the expire time.

#### -----Example Dynamic Web project-----

- This is Dynamic Web Project so not require pom.xml File
- In web.xml file:  

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
 http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
 id="WebApp_ID" version="4.0">
 <display-name>ServletCookies-47</display-name>
 <welcome-file-list>
 <welcome-file>index.html</welcome-file>
 </welcome-file-list>
</web-app>
```

#### In CreateCookiesServlet:

```
package in.nit.servlet;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/create")
public class CreateCookieServlet extends HttpServlet {
 protected void doGet(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException {

 //1. Reading data from Form
 String id=req.getParameter("id");
 String name=req.getParameter("name");

 //2. Create Cookies
 Cookie c=new Cookie(id,"101");
 Cookie c1=new Cookie(name,"Sankar");

 //set Max Life Time in seconds
 c1.setMaxAge(20);// in seconds

 //3. Add to Response
 res.addCookie(c);
 res.addCookie(c1);

 //4. Print Done Message
 PrintWriter pw=res.getWriter();
 pw.print("Cookies Created");

 }
 protected void doPost(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException {
 doGet(req, res);
 }
}
```

In ReadCookieServlet:

```
package in.nit.servlet;
import java.io.IOException;
```

```
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/read")
public class ReadCookieServlet extends HttpServlet {
 protected void doGet(HttpServletRequest req,
 HttpServletResponse res) throws ServletException, IOException {

 //1.Reading all Cookies from request-head
 Cookie[] cks=req.getCookies();

 //2.Print Cookie name, Value
 PrintWriter pw=res.getWriter();
 if(cks!=null&& cks.length!=0) {
 for(int i=0;i<cks.length;i++) {
 pw.print(" Cookie Name=
"+cks[i].getName()+" ,Value= "+cks[i].getValue());
 }
 } else {
 pw.print("No Cookies Found ");
 }

 }

 protected void doPost(HttpServletRequest req,
 HttpServletResponse res) throws ServletException, IOException {
 doGet(req, res);
 }
}
```

Under WebContent:

Index.html:

```
<form action="create" method="post">
<h1>Cookie Parameters....</h1>
<pre>
id: <input type="text" name="id">

Name: <input type="text" name="name">

<input type="submit" value="Enter..">
</pre>
</form>
```

-----Output-----

In Chrome :

Fill id, name:

<http://localhost:3031/ServletCookies-47/>

submit:

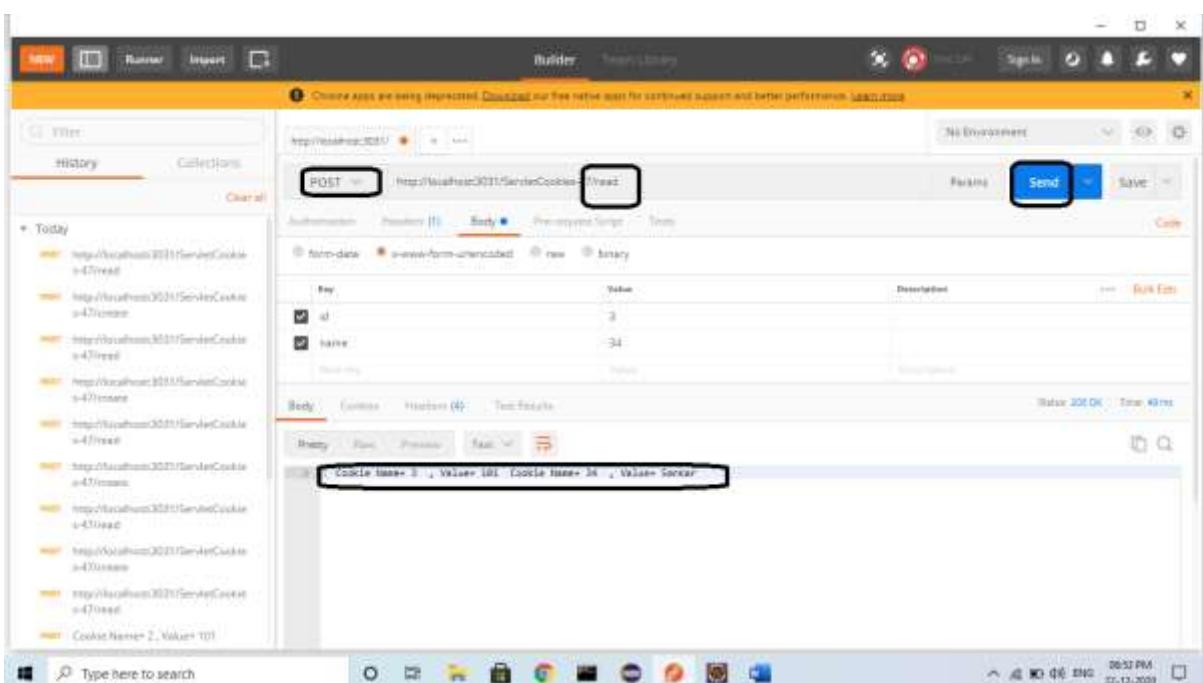
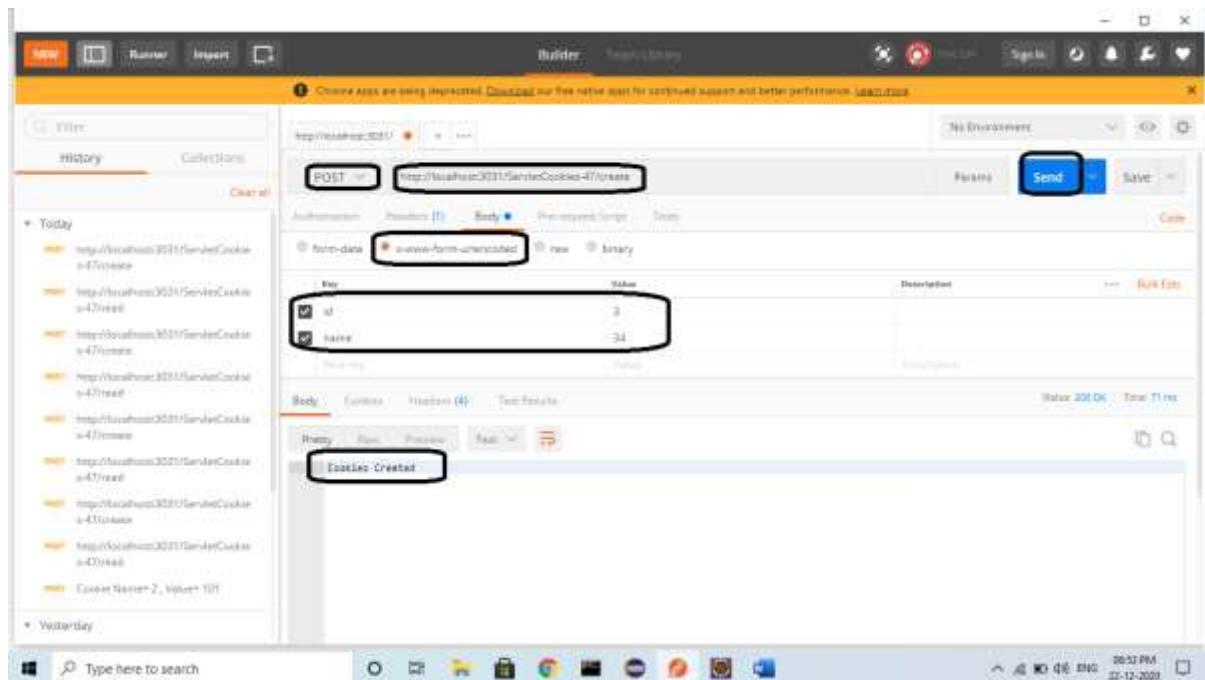
<http://localhost:3031/ServletCookies-47/create>

Cookies Created.

<http://localhost:3031/ServletCookies-47/read>

Cookie Name= 2 , Value= 101

-----PostMan Screen-----



---

### -----Cookies in JAX-RS-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>

 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
```

```
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### In ProducerController:

```
package in.nit.controller;

import javax.ws.rs.CookieParam;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.NewCookie;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

@Path("/cookie")
public class CreateCookieRestController {

 @GET
 @Path("/create")
 public Response createCookie() {

 //create one new Cookie Object
 NewCookie cookie=new NewCookie("LANG","HINDI");
 NewCookie cookie1=new NewCookie("LOC","HYD");
 NewCookie cookie2=new
 NewCookie("COMPANY","dell",null,null,"SAMPLE",30,false);
 //Cookie Added to Response
 Response resp=Response
 .status(Status.OK)
 .entity("Cookies are Created!")
 .cookie(cookie,cookie1,cookie2)
 //or

 //.cookie(cookie).cookie(cookie1).cookie(cookies)2
 .build();
 }
}
```

```
//Response Sent to Client Machine

 return resp;
 }
 @GET
 @Path("/read")
 public String readCookies(
 @CookieParam("LANG")String lang,
 @CookieParam("LOC")String loc,
 @CookieParam("COMPANY")String company
) {
 StringBuffer sb=new StringBuffer();

 String message=sb
 .append("Cookies Data=> ")
 .append(" Language Code: ").append(lang)
 .append(" , Location ").append(loc)
 .append(" company: ").append(company)
 .toString();

 return message;
 }
}
```

-----Output-----

In Chrome

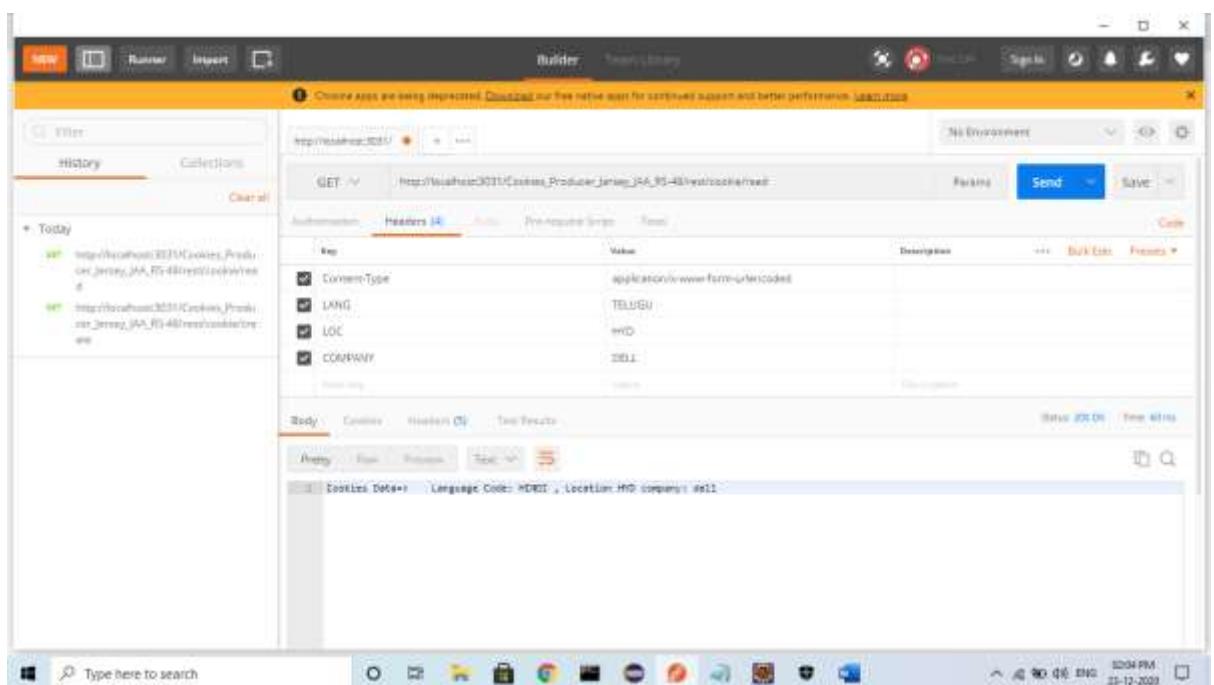
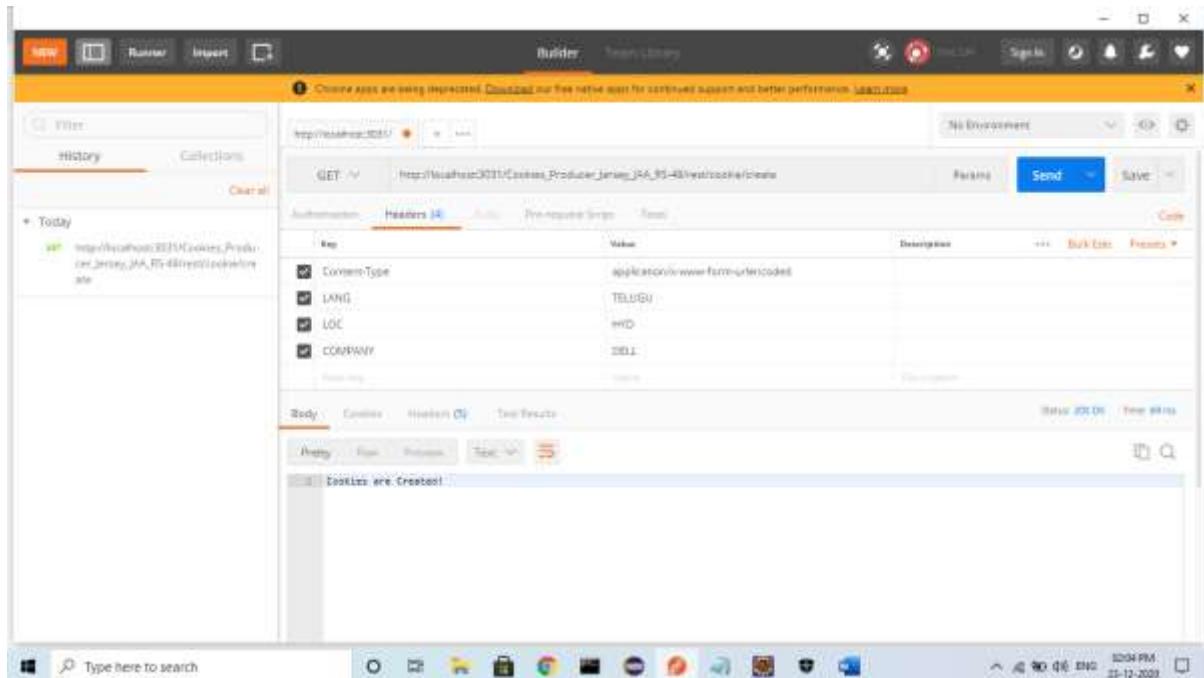
[http://localhost:3031/Cookies\\_Producer\\_Jersey\\_JAA\\_RS-48/rest/cookie/create](http://localhost:3031/Cookies_Producer_Jersey_JAA_RS-48/rest/cookie/create)

Cookies are Created!

[http://localhost:3031/Cookies\\_Producer\\_Jersey\\_JAA\\_RS-48/rest/cookie/read](http://localhost:3031/Cookies_Producer_Jersey_JAA_RS-48/rest/cookie/read)

Cookies Data=> Language Code: HINDI , Location HYD company: dell

-----In POSTMAN SCREEN-----  
Note: No need to pass key-value in body of postman tool ..but I am passed.



- \*\* Cookies created by default with Expire date/time as -1.  
That means Cookies are stored in browser until browser gets closed.  
We can even specify time limit using maxAge(in Sec)  
60sec=1min  
600sec=10min

Example:

```
NewCookie cookie2=new NewCookie(
 "loc","HYD",
 Null,null,"SAMPLE",
 600, //time in sec.
 False);
```

---

### Assignment(Token6):

#### Task:

Create 3 cookies  
userType = ADMIN  
format = RED  
accessMode= Enable

Read then using JAX-RS code

Specify time limit for Expire for any one Cookie.

Using

```
newCookie cookie2=new
NewCookie("accessMode",null,null,null,null,20,false);
```

-----Cookies Assignment Producer App-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
```

```
<scope>test</scope>
</dependency>

<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
 </web-app>
```

In ProducerApp:

```
package in.nit.controller;
import javax.ws.rs.CookieParam;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.NewCookie;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

@Path("/cookie")
public class CreateCookieRestController {

 @GET
 @Path("/create")
 public Response createCookie() {

 //create one new Cookie Object
 NewCookie cookie=new NewCookie("userType","ADMIN");
 NewCookie cookie1=new NewCookie("format","RED");
 NewCookie cookie2=new
 NewCookie("accessMode","Enable",null,null,null,20,false); //20
 seconds after removed
 //Cookie Added to Response
 Response resp=Response
 .status(Status.OK)
 .entity("Cookies are Created!")
 .cookie(cookie,cookie1,cookie2)
 .build();
 //Response Sent to Client Machine

 return resp;
 }
 @GET
 @Path("/read")
 public String readCookies(
 @CookieParam("userType")String user,
 @CookieParam("format")String color,
 @CookieParam("accessMode")String mode
) {
```

```
StringBuffer sb=new StringBuffer();

String message=sb
 .append("Cookies Data=> ")
 .append(" User Type Is : ").append(user)
 .append(" , Format Is: ").append(color)
 .append(" AccessMode Is : ").append(mode)
 .toString();

return message;
}
```

-----output-----

In Chrome:

[http://localhost:3031/AssignmentCookies\\_Producer\\_Jersey\\_JAA\\_RS-49/rest/cookie/create](http://localhost:3031/AssignmentCookies_Producer_Jersey_JAA_RS-49/rest/cookie/create)

Cookies are Created!

[http://localhost:3031/AssignmentCookies\\_Producer\\_Jersey\\_JAA\\_RS-49/rest/cookie/read](http://localhost:3031/AssignmentCookies_Producer_Jersey_JAA_RS-49/rest/cookie/read)

Cookies Data=> User Type Is : ADMIN , Format Is: RED AccessMode Is : Enable

Refresh after 20 sec

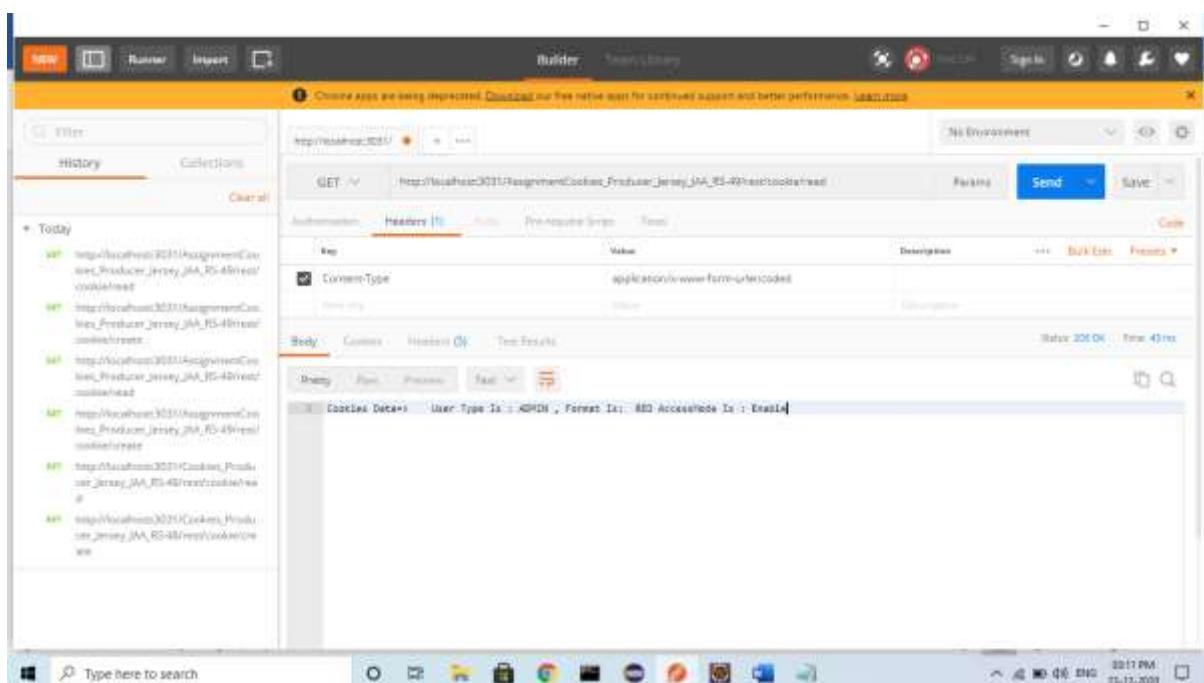
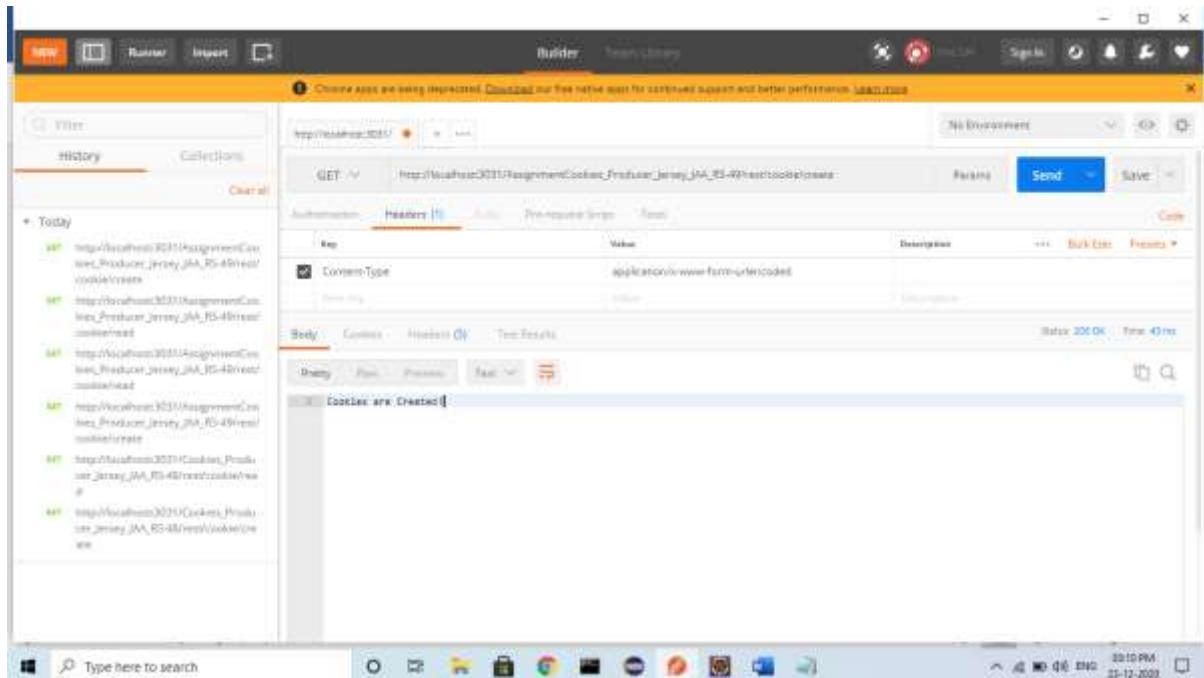
[http://localhost:3031/AssignmentCookies\\_Producer\\_Jersey\\_JAA\\_RS-49/rest/cookie/read](http://localhost:3031/AssignmentCookies_Producer_Jersey_JAA_RS-49/rest/cookie/read)

Cookies Data=> User Type Is : ADMIN , Format Is: RED AccessMode Is : null

-----POSTMAN TOOL-----

Note:

No need to pass key-value in body of postman tool:



Cookies Data=> User Type Is : ADMIN , Format Is: RED AccessMode Is : Enable

## 7.Bean Parameters:

-> `@BeanParam`: This concept is used to convert Input Params into Object(Bean) Format.

-> @BeanParam it will read input parameters entered in request and convert them into one class Object( as object oriented type).

-> We can apply this over any type of parameter in ReST.

Example:

3 Params (id, name, sal) as Input to RestController.

Above 3 values can be converted into one object like: Employee Object.

---

Step#1: Define one Model class with variable and also set/get methods, toString.

Step#2: On top of variable Specify @\_\_Param("key")

Step#3: In RestController method, read Input/Params as One Object using below Syntax. **@BeanParam** ModelClass ObjectName.

-----BeanParamProducerJerseyApp-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
```

[https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->](https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet)

```
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
```

</web-app>

In Model Class:

```
package in.nit.model;

import javax.ws.rs.QueryParam;

import lombok.Data;

@Data
public class Employee {
 @QueryParam("eid")
 private int empld;
 @QueryParam("ename")
 private String empName;
 @QueryParam("esal")
 private double empSal;
}
```

In BeanParamProducerController:

```
package in.nit.controller;

import javax.ws.rs.BeanParam;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

import in.nit.model.Employee;

@Path("emp")
public class BeanParamProcuderController {
 //.../rest/emp?eid=202&ename=Sankar&esal=23.44

 @GET
 @Path("/data")
 public String readInputs(
 @BeanParam Employee employee
) {

 return "Data is => "+employee;
 }
}
```

```
}
```

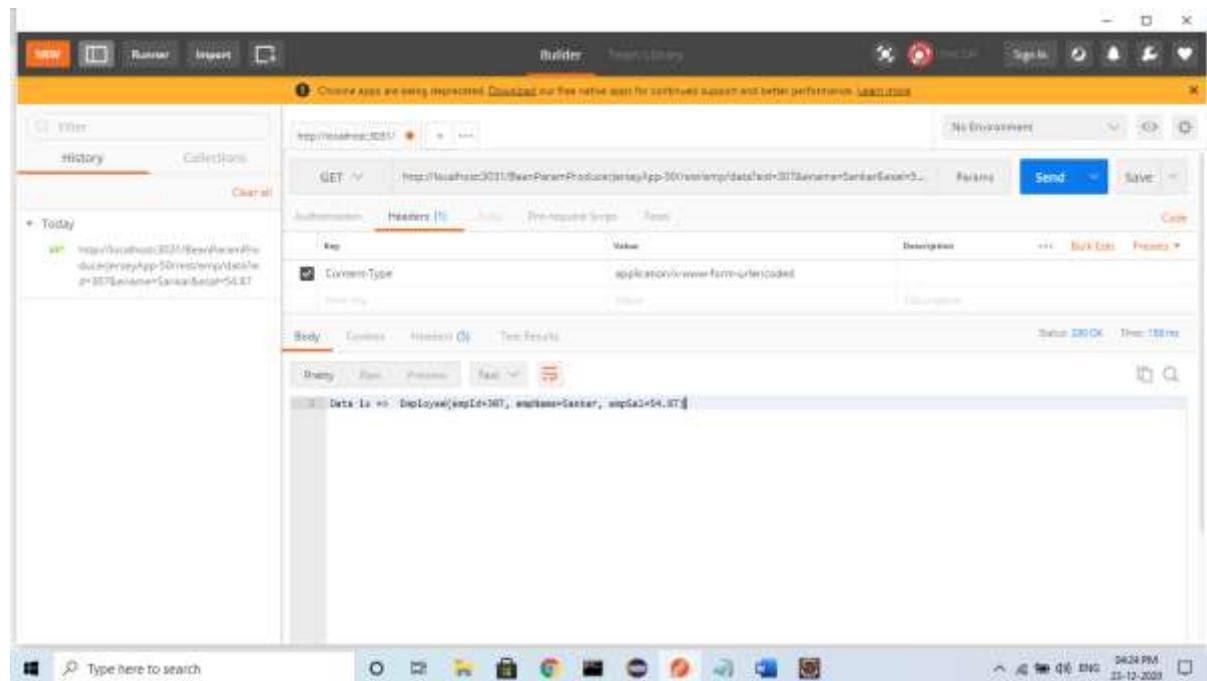
-----output-----

In chrome:

<http://localhost:3031/BeanParamProducerJerseyApp-50/rest/emp/data?eid=307&ename=Sankar&esal=54.87>

Data is => Employee(emplId=307, empName=Sankar, empSal=54.87)

Note: you must pass @QuerParam names not pass field(Variable) names.



-----BeanParamTypesOfParamtersProducerApp-----

→ \*\* Bean Param also works on different types of Params to convert data into Object.

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
 </dependency>
</dependencies>
```

### In web.xml File:

```
<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### In Model class:

```
package in.nit.model;

import javax.ws.rs.FormParam;
import javax.ws.rs.HeaderParam;
import javax.ws.rs.QueryParam;

import lombok.Data;

@Data
public class Employee {
 @QueryParam("eid")
 private int emplId;
 @HeaderParam("ename")
 private String empName;
 @FormParam("esal")
 private double empSal;
}
```

### In BeanParamProducerController:

Note: if you are using @FormParam must use @POST.

```
package in.nit.controller;

import javax.ws.rs.BeanParam;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.MediaType;

import in.nit.model.Employee;

@Path("/emp")
public class BeanParamProcuderController {

 @POST
 @Path("/data")
 // @Consumes(MediaType.APPLICATION_FORM_URLENCODED
 // it is optional
 public String readInputs(
 @BeanParam Employee employee
) {

 return "Data is => "+employee;
 }
}
```

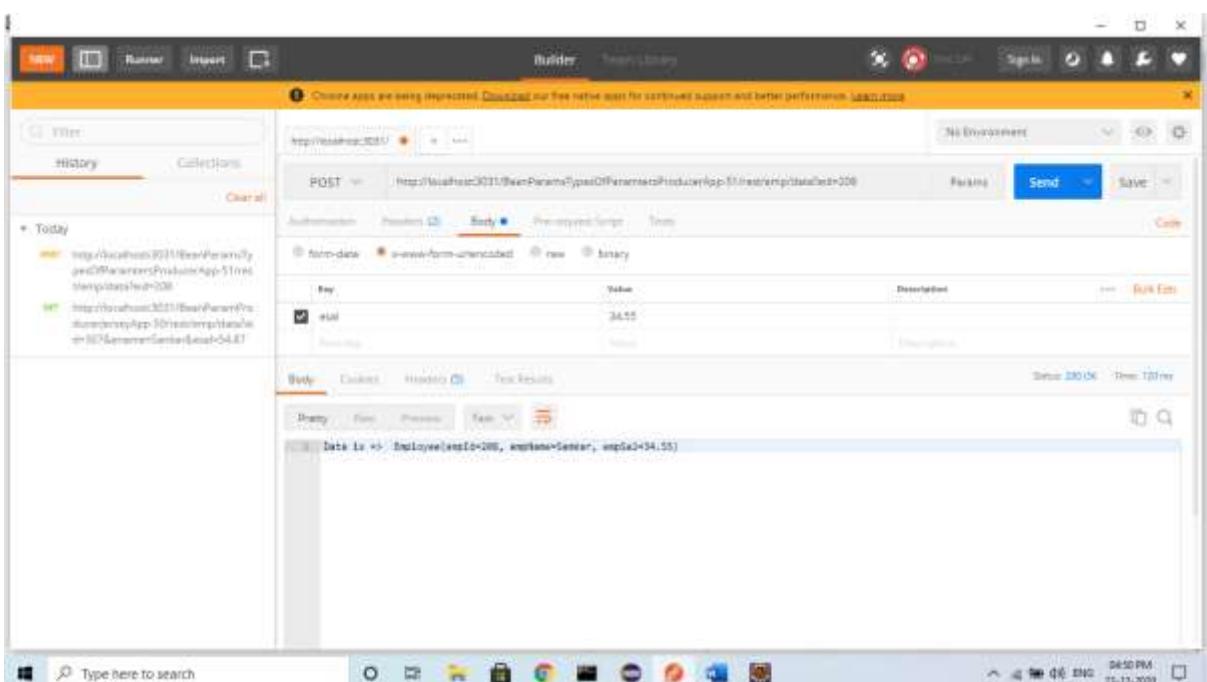
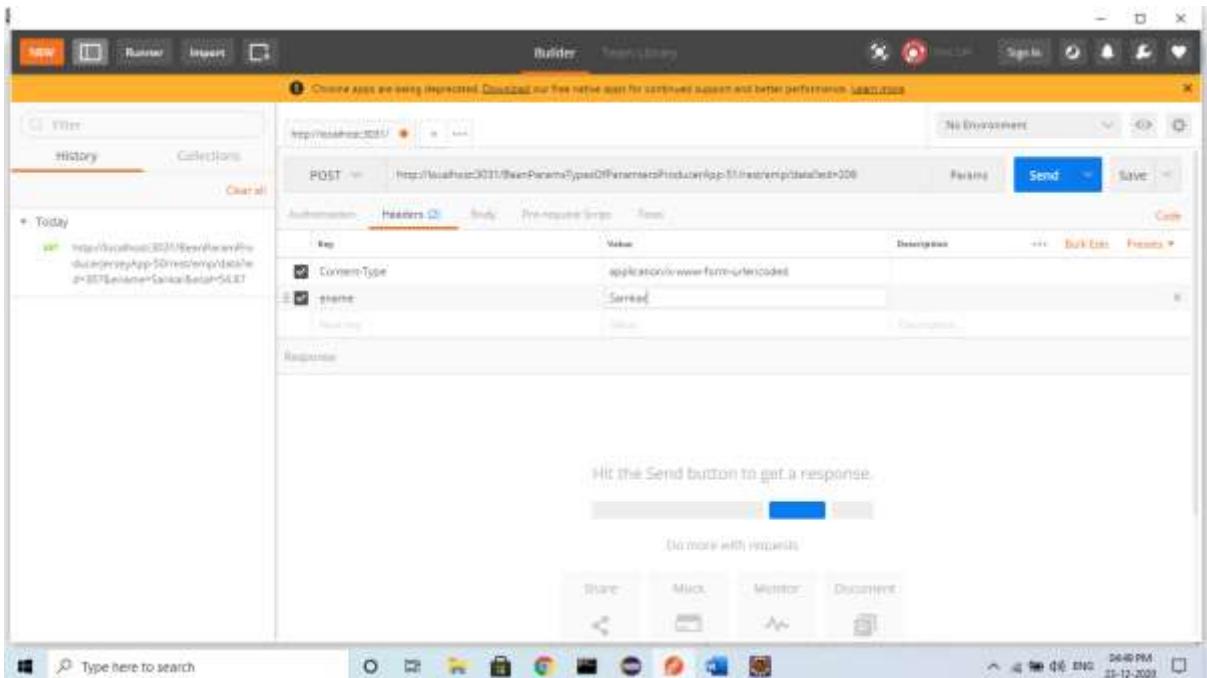
-----output-----

In Chrome

In chrome not possible to see output.

-----POSTMAN TOOL-----

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>



## Example:

All @QueryParams converted to one Employee class Object.

-----@BeanParam\_Jersey2.x\_Archetype\_ProducerApp-----

In pom.xml File:

```
<build>
 <finalName>BeanParam_Jersey2.x_Archetype_ProducerApp-69</finalName>
 <plugins>
 <plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>2.5.1</version>
 <inherited>true</inherited>
 <configuration>
 <source>15</source>
 <target>15</target>
 </configuration>
 </plugin>
 </plugins>
</build>

<dependencyManagement>
 <dependencies>
 <dependency>
 <groupId>org.glassfish.jersey</groupId>
 <artifactId>jersey-bom</artifactId>
 <version>${jersey.version}</version>
 <type>pom</type>
 <scope>import</scope>
 </dependency>
 </dependencies>
</dependencyManagement>

<dependencies>
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet-core</artifactId>
 <!-- use the following artifactId if you don't need servlet 2.x compatibility -->
 <!-- artifactId>jersey-container-servlet</artifactId -->
 </dependency>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 </dependency>
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
```

```
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.16</version>
<scope>provided</scope>
</dependency>
 <!-- uncomment this to get JSON support
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-binding</artifactId>
 </dependency>
 -->
</dependencies>
<properties>
 <jersey.version>2.30</jersey.version>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 </properties>
```

In web.xml File:

```
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 <servlet>
 <servlet-name>Jersey Web Application</servlet-name>
 <servlet-
class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-
name>
 <param-value>in.nit</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
 <servlet-name>Jersey Web Application</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model class:

```
package in.nit.model;

import javax.ws.rs.QueryParam;
import lombok.Data;
@Data
public class Employee {

 private @QueryParam("eid")Integer eid;
 private @QueryParam("ename")String ename;
 private @QueryParam("esal")Double esal;
}
```

In RestController:

```
package in.nit;

import javax.ws.rs.BeanParam;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import in.nit.model.Employee;

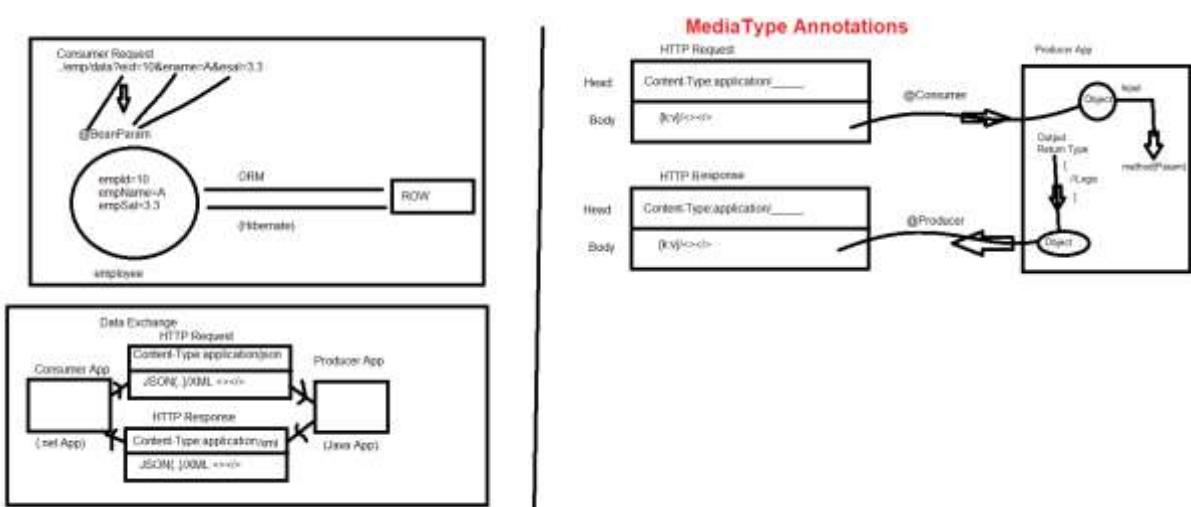
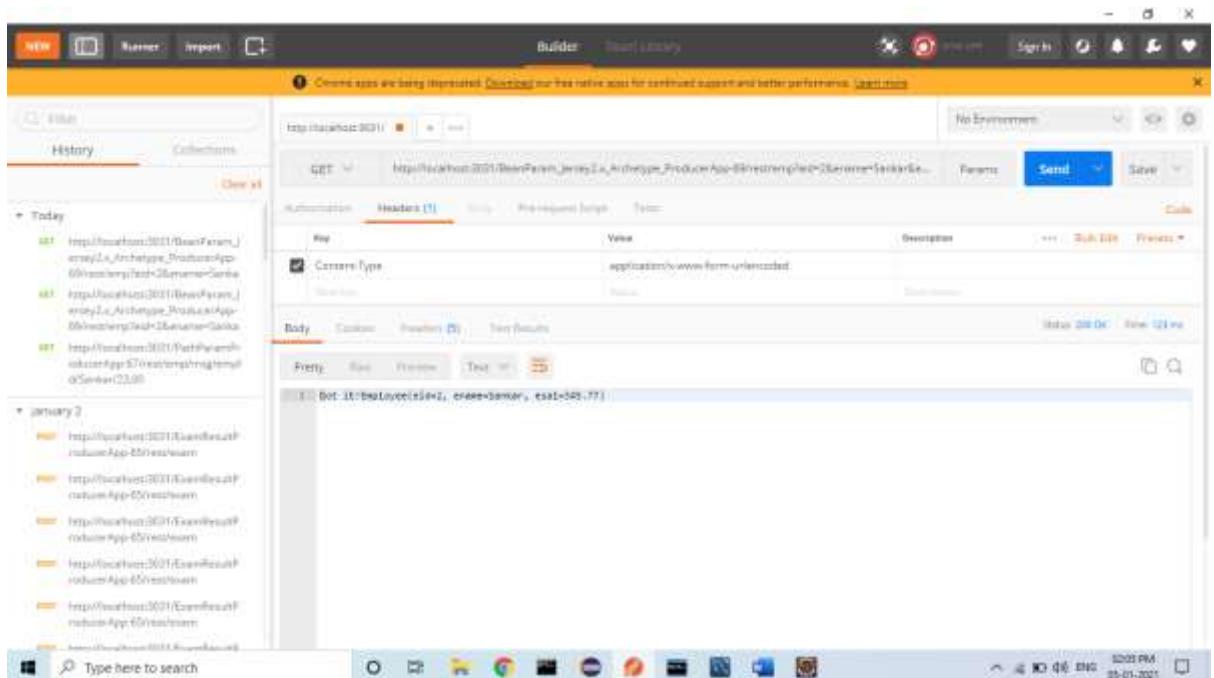
@Path("/emp")
public class BeanParamRestController {
 @GET
 @Produces(MediaType.TEXT_PLAIN)
 public String getIt(@BeanParam Employee e) {
 return "Got it!"+e;
 }
}
```

-----Output-----

In Chrome:

[http://localhost:3031/BeanParam\\_Jersey2.x\\_Archetype\\_Producer\\_App-69/rest/emp?eid=2&ename=Sankar&esal=345.77](http://localhost:3031/BeanParam_Jersey2.x_Archetype_Producer_App-69/rest/emp?eid=2&ename=Sankar&esal=345.77)

Got it! Employee(eid=2, ename=Sankar, esal=345.77)



## MediaTypes/Global DataTypes:

- In real time, webservices applications i.e Producer and Consumer may be different language applications.
- In this case data exchange is done using XML or JSON format.
- There are called as Global Data Types or MediaTypes.
- Here, XML or JSON can be understand by any High-Level Languages like Java, .net, PHP, Python.... etc
- Consumer will send HTTP Request that contains Head and Body.
- Body always contains Data (Content).
- Headers contains one Header Param ‘Content-Type: \_\_\_\_\_’.
- It indicates what type of data (Content) exist in Body.

Example:

If Body has JSON data then Header Param will be  
Content-type: application/json  
else if data is XML in Body then header Param will be  
Content-type : application-xml.

- In same way Producer will process request and gives Response that holds Global Data format back to Consumer.

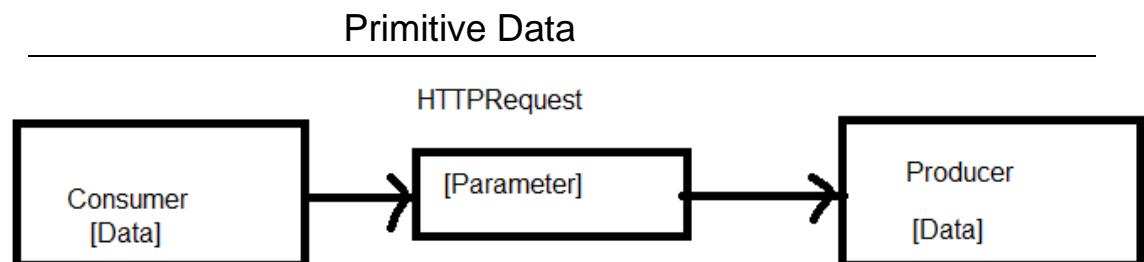
## Data Exchange Concept in ReST Webservices:

ReST supports data exchange in two Formats:

1. Primitive Type Data.
2. Global Type Data(JSON/XML).
1. Primitive Type Data (One Value):

From Consumer to Producer primitive Data like: 10(int),  
“Sankar”(String), 2.3(double),.....

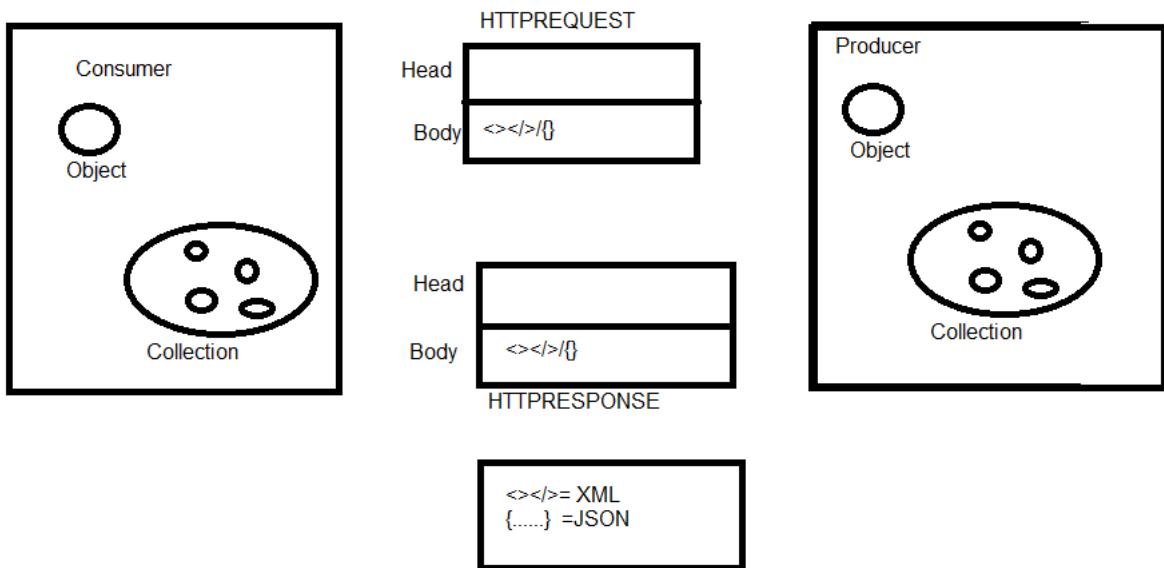
Will be send in “String Format” only, using HTTPRequest  
(URL/Head/Body).



### 2. Global Type Data:

- To Send Objects and Collection data from Consumer to producer and provided to Consumer, ReST uses global Formats like XML/JSON.

- It uses HTTPRequest/HTTPResponse Body to place JSON/XML Data.
- Its look like:



## MediaType Annotations

JAX-RS has provided two MediaType annotations.

Those are:

1. @Consumes
2. @Produces

### 1. @Consumes: It works on Request/Input/Parameter

This annotation will read HTTP Request.

First it checks Header Param 'Content-Type: \_\_\_\_\_',

Example:

Content-Type:application/json

Then JSON Data is converted into Object

This Object is given as Input to RestController  
i.e Method Parameter.

### 2. @Produces: It works on Response/Output/ReturnType

After Executing RestController method,

Consider it returns one Object i.e Output.

This Object is converted into Global Format EX: XML.

Then same data is placed into HTTP Response Body.

It also updates Response Header i.e

Content-type: application/xml

→ To work with JSON, in pom.xml we need to add one Dependency:

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey
-media-json-jackson -->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
```

Example:

```
@Consumes("application/json")
(or)
```

We can use class MediaType (javax.ws.rs.core)

Is having public final static String i.e like

APPLICATION\_JSON="application/json"

```
@Consumes(MediaType.APPLICATION_JSON)
```

In same way to work with Produces

```
@Produces("application/json")
(or)
```

```
@Produces(MediaType.APPLICATION_JSON)
```

→ \*\* At a time, we can use either @Consumes or @Produces even both.

→ \*\*\* To work with XML, add below dependencies

```
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
```

```
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-impl --
->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-core --
->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>
```

-----MediaTypes/Gobal DataTypesProducerAPP-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with JSON -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson -->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime -->
>
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
```

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

Note: give ni.nit param value, so detect sub two packages

In Model class:

**package** in.nit.model;

**import** javax.xml.bind.annotation.XmlRootElement;

**import** lombok.Data;

```
@Data
@XmlRootElement
public class Student {
 private int sld;
 private String sName;
 private double sfees;
}
```

In MediaTypesProducerController:

Controller1:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import in.nit.model.Student;

@Path("/std")
public class MediaTypesProducerController {

 /* here, we are working on return type only */
 @GET
 @Path("/info")
 @Produces("application/json")
 public Student showData() {
 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setSfees(34.56);
 return s;
 }
}
```

-----output-----

In chrome:

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/info>

```
{"sfees":34.56,"sid":202,"sname":"Sankar"}
```

---

In Controller2:

```
package in.nit.controller;
```

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import in.nit.model.Student;

@Path("/std")
public class MediaTypesProducerController {

 /* here, we are working on return type only */
 @Path("/info")
 @GET
 //@Produces("application/json")
 // or
 @Produces(MediaType.APPLICATION_JSON)
 public Student showData() {
 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setSfees(34.56);
 return s;
 }
}
```

-----output-----

In chrome :

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/info>  
{ "sfees":34.56, "sid":202, "sname":"Sankar" }

---

In Controller3:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
```

```
import in.nit.model.Student;

@Path("/std")
public class MediaTypesProducerController {

 /* here, we are working on return type only */
 @Path("/info")
 @GET
 //@Produces("application/json")
 // or
 //@Produces(MediaType.APPLICATION_JSON)

 @Produces("application/xml")
 public Student showData() {
 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setSfees(34.56);
 return s;
 }
}
```

-----output-----

In chrome:

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/info>

```
<student>
<SId>202</SId>
<SName>Sankar</SName>
<sfees>34.56</sfees>
</student>
```

In Controller4:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
```

```
import in.nit.model.Student;

@Path("/std")
public class MediaTypesProducerController {

 /* here, we are working on return type only */
 @Path("/info")
 @GET
 //@Produces("application/json")
 // or
 //@Produces(MediaType.APPLICATION_JSON)

 //@Produces("application/xml")
 //or
 @Produces(MediaType.APPLICATION_XML)
public Student showData() {
 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setSfees(34.56);
 return s;
}
}
```

-----output-----

In chrome:

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/info>

```
<student>
<SId>202</SId>
<SName>Sankar</SName>
<sfees>34.56</sfees>
</student>
```

Note by default coming xml data as output:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
```

```
import javax.ws.rs.core.MediaType;
import in.nit.model.Student;

@Path("/std")
public class MediaTypesProducerController {

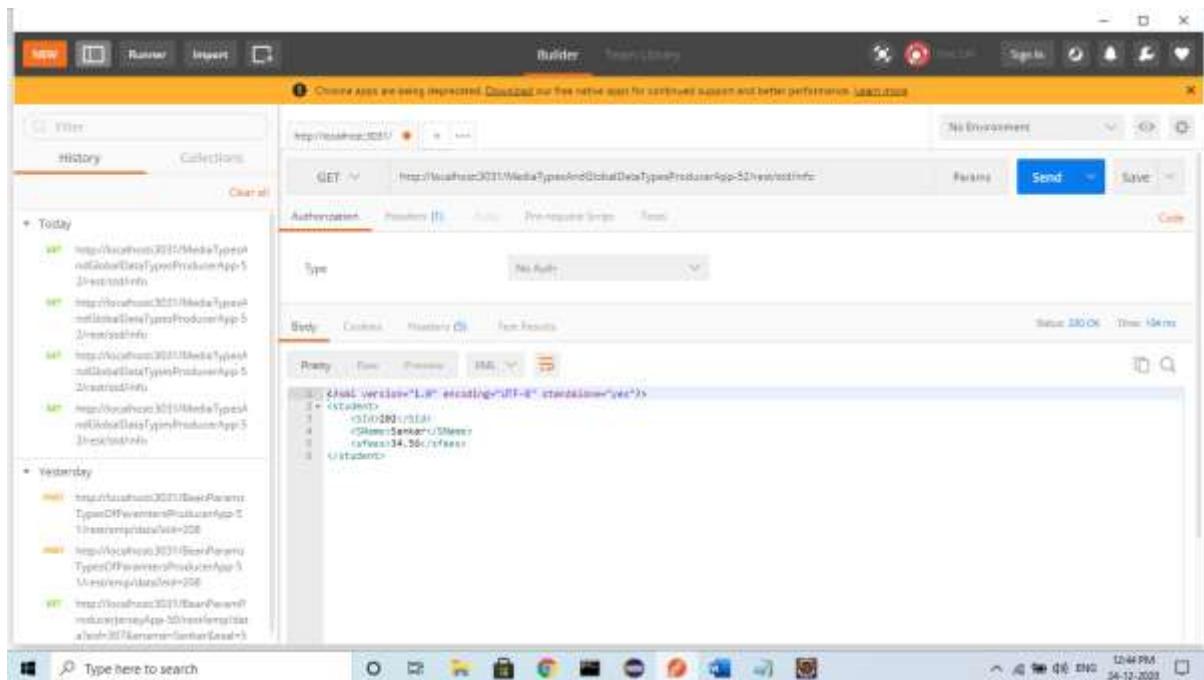
 /* here, we are working on return type only */
 @Path("/info")
 @GET
 //@Produces("application/json")
 // or
 //@Produces(MediaType.APPLICATION_JSON)

 //@Produces("application/xml")
 //or
 //@Produces(MediaType.APPLICATION_XML)
 public Student showData() {
 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setSfees(34.56);
 return s;
 }
}

<student>
<SId>202</SId>
<SName>Sankar</SName>
<sfees>34.56</sfees>
</student>
```

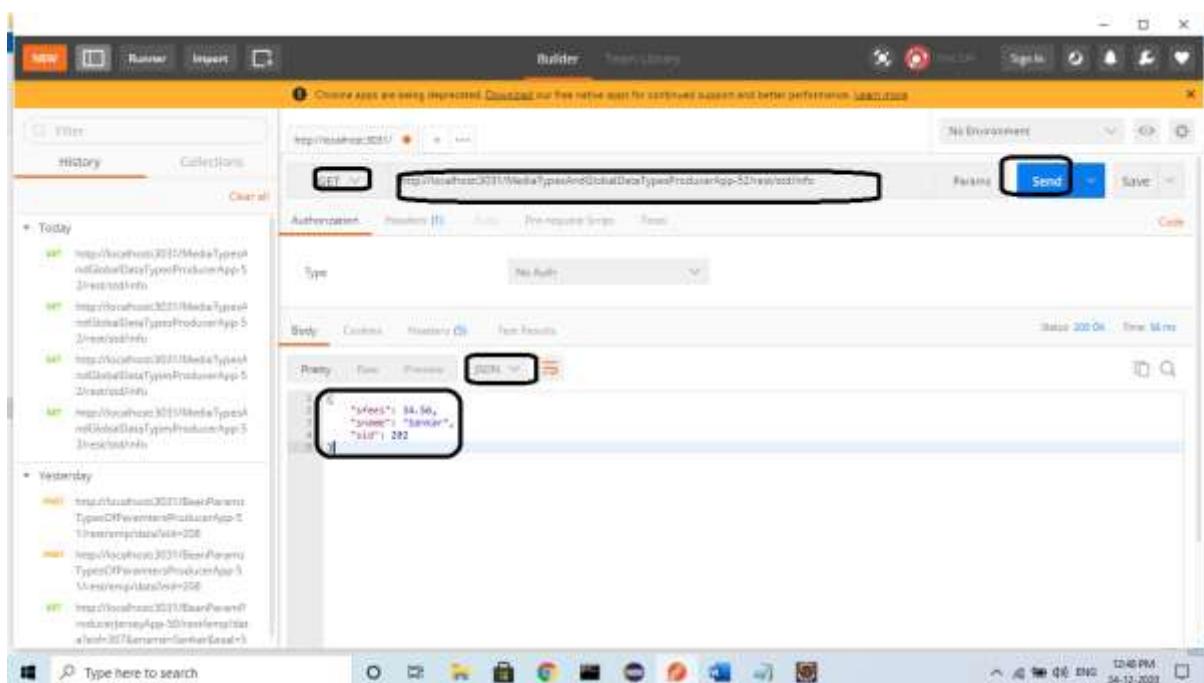
#### -----POSTMAN TOOL-----

Note: xml output is by default coming:  
No need `@Produces("application/xml")` in controller



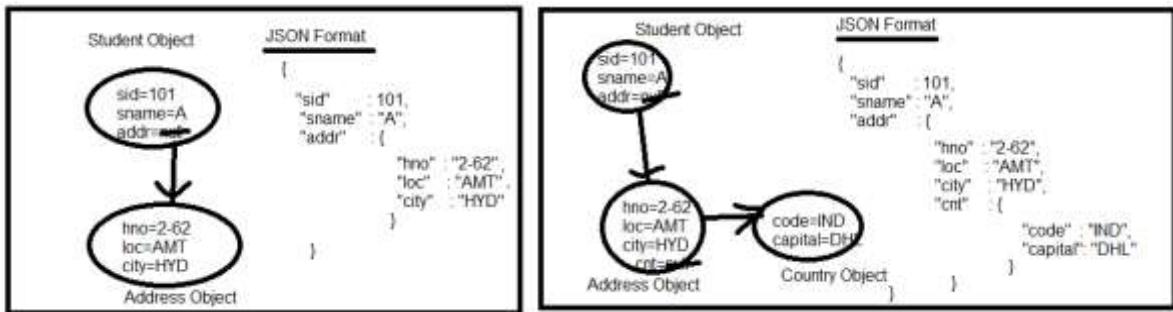
Note: JSON output is not come by default :

Must use `@Produces("application/json")` in controller:



Note:

- a) To work with JSON, we must add **jersey-media-json-jackson** dependency in pom.xml also provide media type as:  
**Application/json or MediaType.APPLICATION\_JSON.**
  - b) To work with XML, we must add jaxb dependencies in pom.xml  
 Also provide media type as:  
**Application/xml or MediaType.APPLICATION\_XML**
- 



@Produces Part#2:

RestController method return Object->JSON/XML->HTTP Response Body.

- \*\* Working on multiple classes connected using HAS-A relation\*\*
1. Student HAS-A Address (Address class Object is used in Student class).

```

{
 //std JSON
 "hasVariable" : { //addr JSON
 ...
 }
}

```

---

-----MediaTypesHAS-A\_RelationProducerApp-----

In pom.xml File:

```

<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

```

```
<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with JSON -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with XML -->
 <!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
 <dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
 </dependency>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
 </web-app>
```

In Model classes: Student and Address:

In Address:

Note: In dependent class no need to mention  
`@XmlRootElement`.

```
package in.nit.model;
import lombok.Data;
```

```
@Data
public class Address {
 private String hNo;
 private String loc;
 private double city;
}
```

In Student class:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.AllArgsConstructorConstructor;
import lombok.Data;
import lombok.NoArgsConstructorConstructor;
```

```
@Data
@XmlRootElement
@AllArgsConstructorConstructor
@NoArgsConstructorConstructor
public class Student {
 private int sld;
 private String sName;
 //one class has DataType -> Creating variable
 private Address addr;//HAS-A Relation
}
```

In StudentProducerController:

```
package in.nit.controller;

import javax.ws.rs.GET;
```

```
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import in.nit.model.Address;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 @Produces("application/json")
 //@Produces("application/xml")
 public Student showData() {

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");

 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setAddr(addr);//Link between Objects.
 return s;
 }
}
```

-----Output-----

In chrome:

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/link>

{"addr":{"loc":"HYD","city":"Ammerpet","hno":"2-62"}, "sname":"Sankar", "sid":202}

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
```

```
import in.nit.model.Address;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 //@Produces("application/json")
 @Produces("application/xml")
 public Student showData() {

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");

 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setAddr(addr);//Link between Objects.
 return s;
 }
}
```

-----output-----

In chrome:

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/link>

```
<student>
<addr>
<city>Ammerpet</city>
<HNo>2-62</HNo>
<loc>HYD</loc>
</addr>
<SId>202</SId>
<SName>Sankar</SName>
</student>
```

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
```

```
import javax.ws.rs.Produces;

import in.nit.model.Address;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 //@Produces("application/json")
 //@Produces("application/xml")
 //Request (header Param: Accept = application/xml)
 @Produces({"application/json","application/xml"})
 public Student showData() {

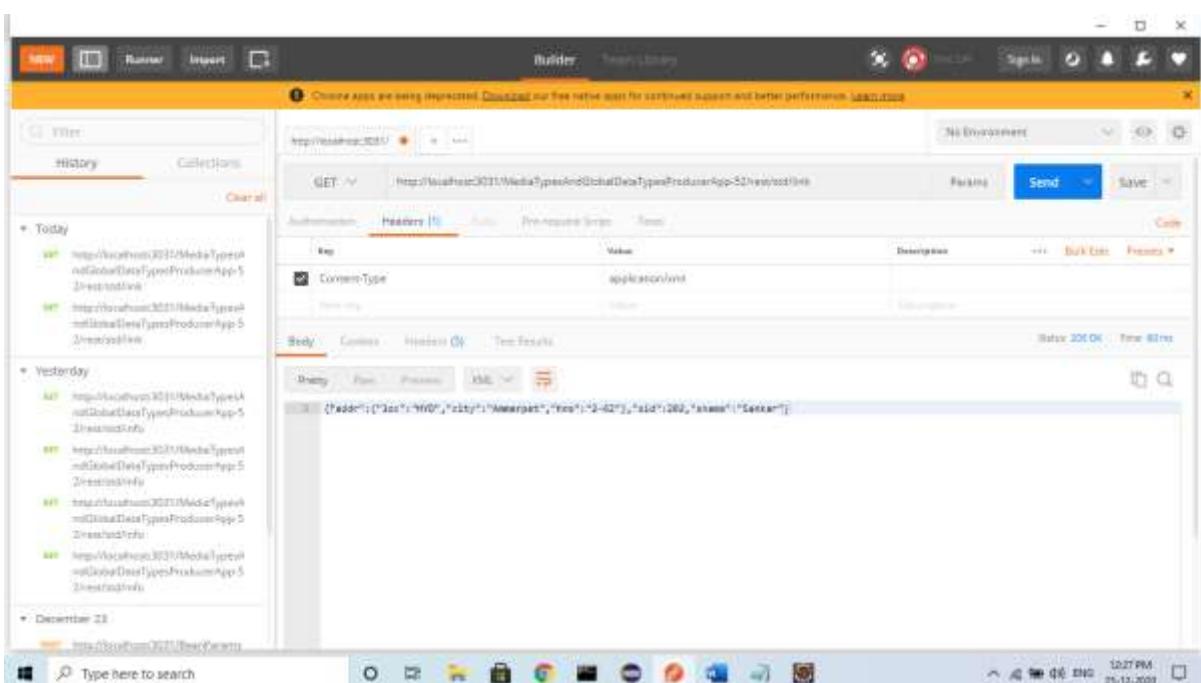
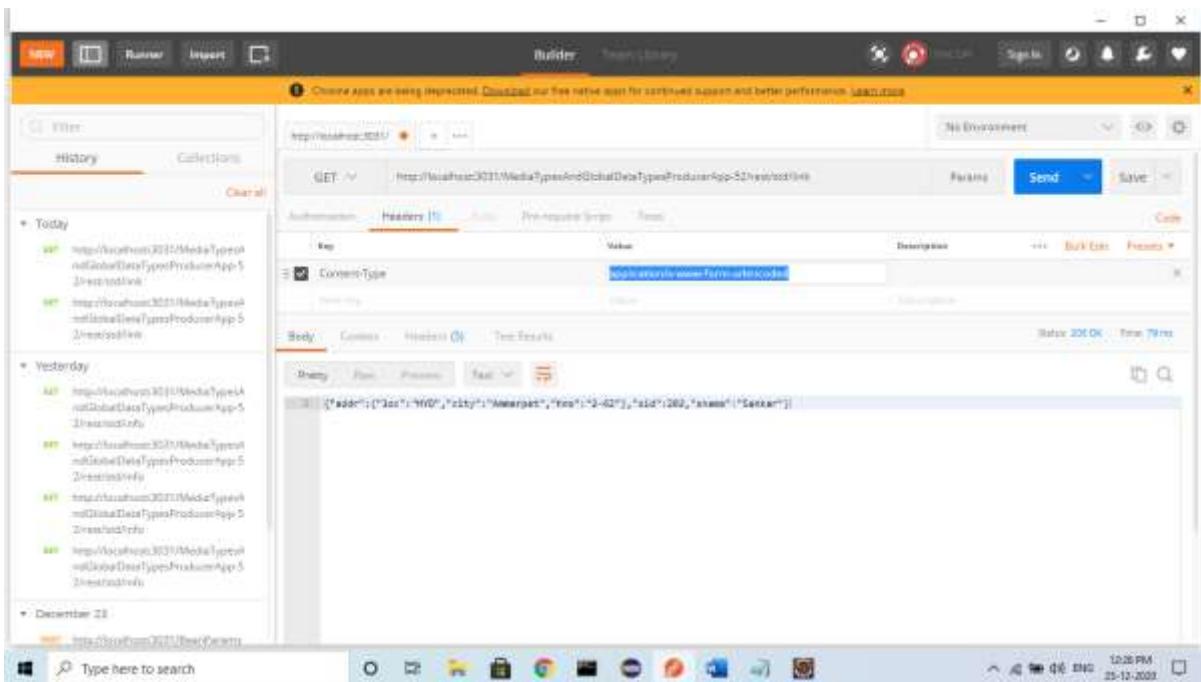
 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");

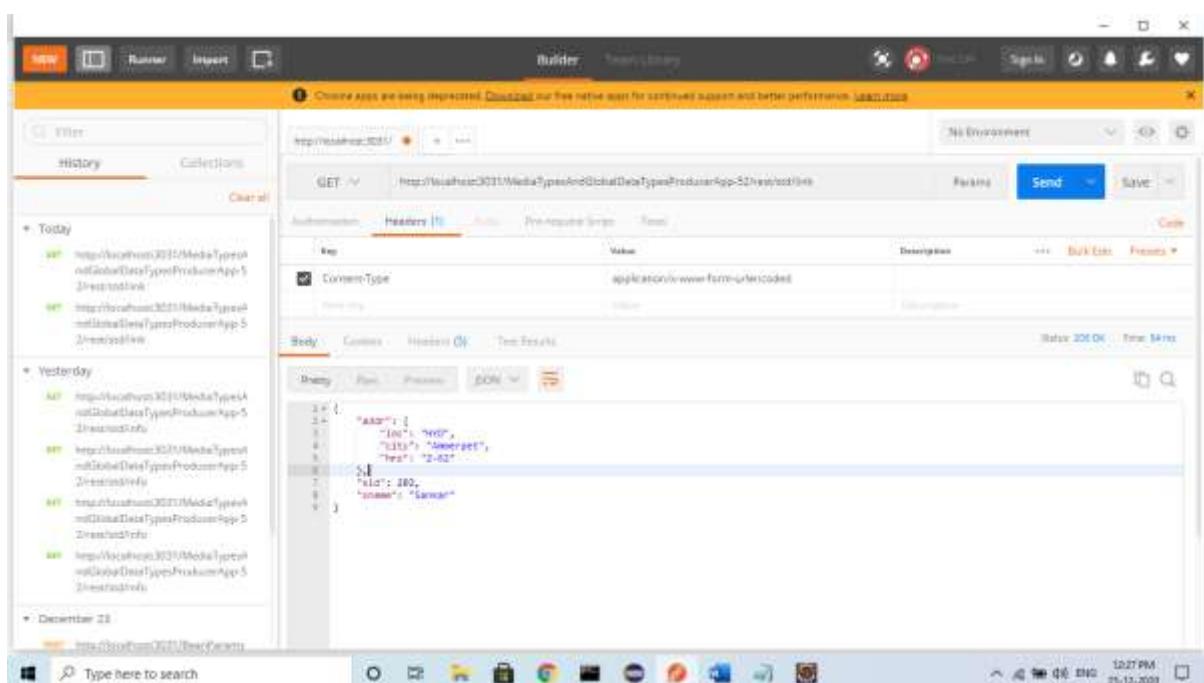
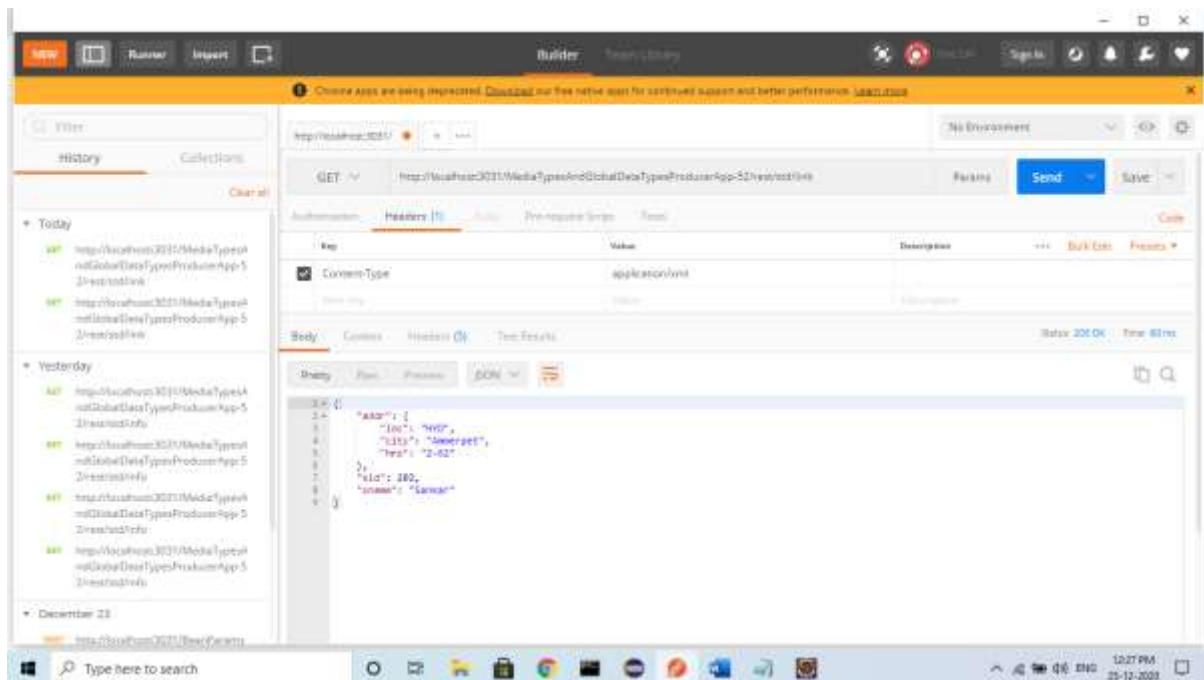
 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setAddr(addr);//Link between Objects.
 return s;
 }
}
```

-----output-----

In POSTMAN TOOL:

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>





## -----Example2 ProducerApp-----

Adding 3<sup>rd</sup> class i.e Country which is connected to Address class  
2. Student HAS-A Address HAS-A Country.

JSON out Format.

```
//Student Object
 "hasAVariable" :{//Address object
 "hasAVariable" :{//Country Object
 }
 }
 }
```

In Pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with JSON -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>jersey-media-json-jackson</artifactId>
<version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model Classes Student, Address, Country.

Country:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Country {
 private String code;
 private String capital;
}
```

Address:

```
package in.nit.model;
```

```
import lombok.AllArgsConstructor;
```

```
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Address {
private String hNo;
private String loc;
private String city;
//class as DataType->creating one variable.
private Country ctn;//HAS-A Relation.
}
```

In Student class:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@XmlRootElement
@AllArgsConstructor
@NoArgsConstructor
public class Student {
private int sld;
private String sName;
//one class has DataType -> Creating variable
private Address addr;//HAS-A Relation
}
```

In StudentRestController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
```

```
import in.nit.model.Address;
import in.nit.model.Country;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 //@Produces("application/json")
 //@Produces("application/xml")
 //Request (header Param: Accept = application/xml)
 @Produces({"application/json","application/xml"})
 public Student showData() {
 Country ctn=new Country();
 ctn.setCode("3vrgr");
 ctn.setCapital("Delhi");

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");
 addr.setCtn(ctn); //HAS-A Link between Objects.

 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setAddr(addr); //HAS-A Link between Objects.
 return s;
 }
}
```

-----output-----

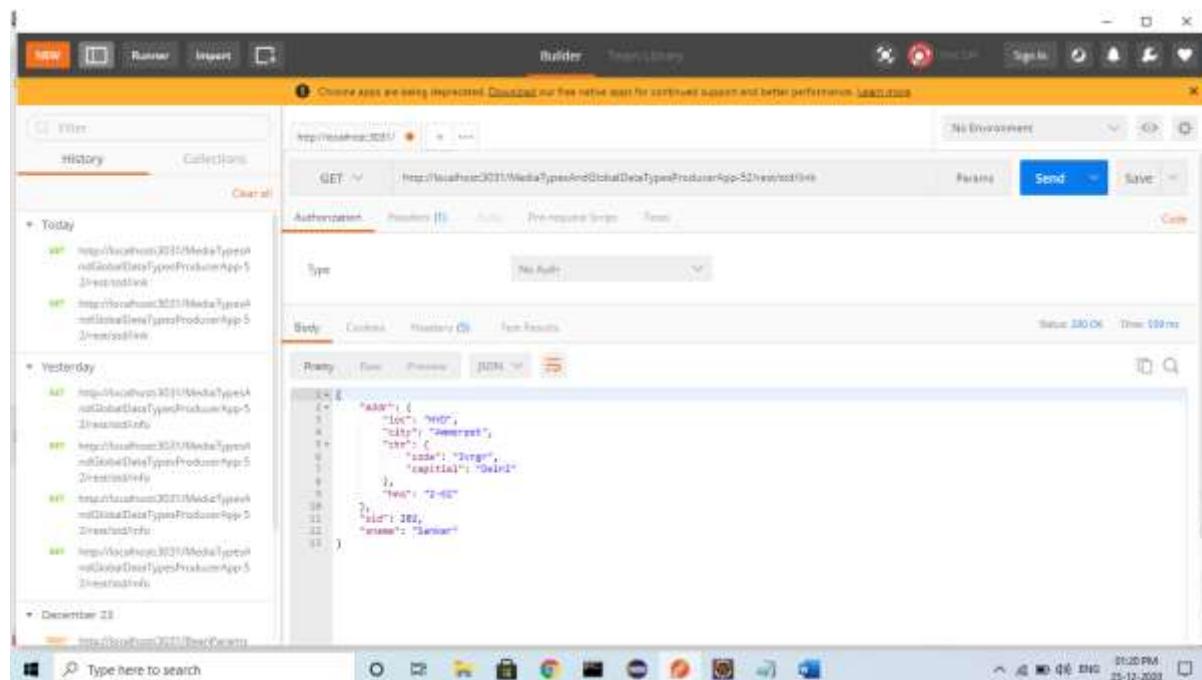
In chrome:

<http://localhost:3031/MediaTypesAndGlobalDataTypesProducerApp-52/rest/std/link>

```
<student>
<addr>
<city>Ammerpet</city>
<ctn>
<capital>Delhi</capital>
<code>3vrgr</code>
```

```
</ctn>
<HNo>2-62</HNo>
<loc>HYD</loc>
</addr>
<SIid>202</SIid>
<SName>Sankar</SName>
</student>
```

## -----POSTMAN TOOL-----



## -----Example3: MediaTypes List ProducerApp-----

- ➔ Working with List Collection.
- ➔ JSON: List of Objects=> [{} , {} , {} , {} , ..... ]

Consider Example: Student HAS-A List<Course>

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with JSON -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with XML -->
 <!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
 <dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
 <dependency>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<groupId>com.sun.xml.bind</groupId>
<artifactId>jaxb-core</artifactId>
<version>2.3.0</version>
</dependency>

<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.16</version>
<scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In Model classes:

### Country:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Country {
 private String code;
 private String capital;
}
```

### Address:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Address {
 private String hNo;
 private String loc;
 private String city;
 //class as DataType->creating one variable.
 private Country ctn;//HAS-A Relation.
}
```

### Course:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Course {
 private String cname;
 private Double cfee;
 private int duration;
}
```

Student:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@XmlRootElement
@AllArgsConstructor
@NoArgsConstructor
public class Student {
 private int sld;
 private String sName;
 //one class has DataType -> Creating variable
 private Address addr; //HAS-A Relation
 private List<Course> cos; //HAS-A Relation
}
```

In StudentProducerController:

```
package in.nit.controller;

import java.util.ArrayList;
import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
```

```
import in.nit.model.Address;
import in.nit.model.Country;
import in.nit.model.Course;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 //@Produces("application/json")
 //@Produces("application/xml")
 //Request (header Param: Accept = application/xml)
 @Produces({"application/json","application/xml"})
 public Student showData() {
 Country ctn=new Country();
 ctn.setCode("3vrgr");
 ctn.setCapital("Delhi");

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");
 addr.setCtn(ctn); //HAS-A link between Objects.

 Student s=new Student();
 s.setSId(202);
 s.setSName("Sankar");
 s.setAddr(addr); //Link between Objects.

 Course c1=new Course("webservices",234.2,6);
 Course c2=new Course("hibernate",231.2,4);
 Course c3=new Course("Spring",201.2,8);
 //JDK7 Collection-Type Inference (empty <> operator)

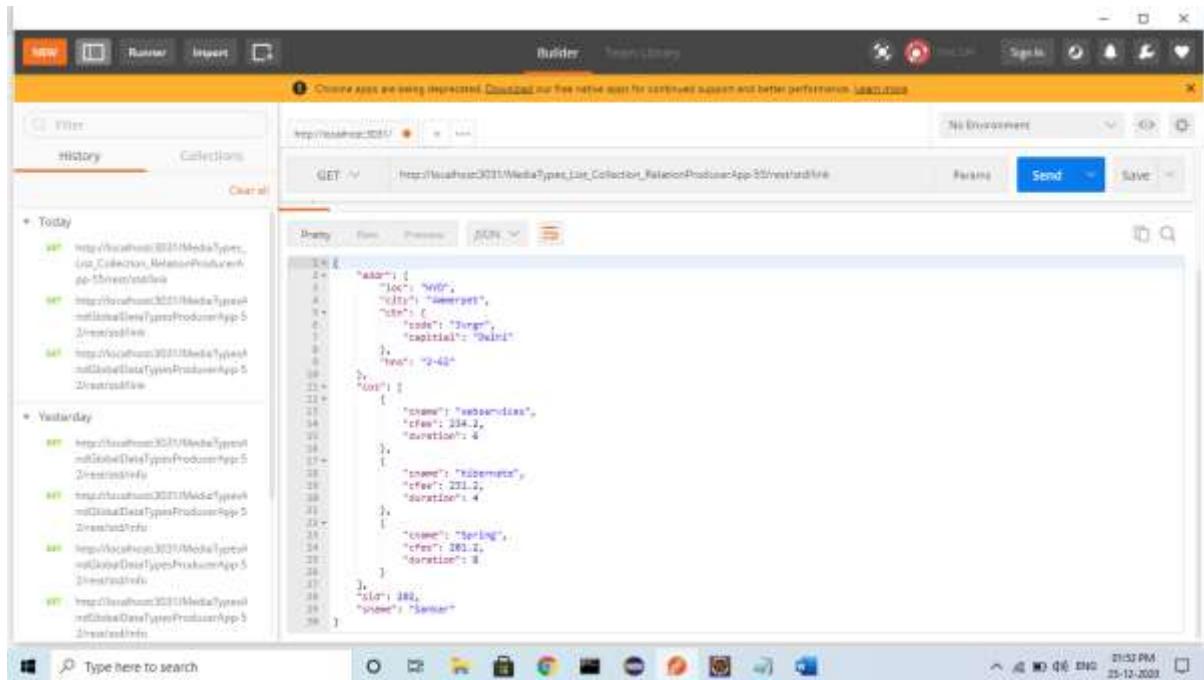
 List<Course> list=new ArrayList<>();
 list.add(c1);
 list.add(c2);
 list.add(c3);
 s.setCos(list);
 return s;
 }
}
```

-----Output-----

[http://localhost:3031/MediaTypes\\_List\\_Collection\\_RelationProducerApp-55/rest/std/link](http://localhost:3031/MediaTypes_List_Collection_RelationProducerApp-55/rest/std/link)

```
<student>
<addr>
<city>Ammerpet</city>
<ctn>
<capital>Delhi</capital>
<code>3vrgr</code>
</ctn>
<HNo>2-62</HNo>
<loc>HYD</loc>
</addr>
<cos>
<cfee>234.2</cfee>
<cname>webservices</cname>
<duration>6</duration>
</cos>
<cos>
<cfee>231.2</cfee>
<cname>hibernate</cname>
<duration>4</duration>
</cos>
<cos>
<cfee>201.2</cfee>
<cname>Spring</cname>
<duration>8</duration>
</cos>
<SId>202</SId>
<SName>Sankar</SName>
</student>
```

-----POSTMAN TOOL-----



Note: If one Model class is connected with another Model class using HAS-A Relation then, in this case connected Model class Object is not given (not exist) then, Output for reference link (HAS-A variable) is null.

package in.nit.controller;

```

import java.util.ArrayList;
import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import in.nit.model.Address;
import in.nit.model.Country;
import in.nit.model.Course;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 //@Produces("application/json")
 //@Produces("application/xml")

```

```
//Request (header Param: Accept = application/xml)
@Produces({"application/json","application/xml"})
public Student showData() {
 Country ctn=new Country();
 ctn.setCode("3vrgr");
 ctn.setCapital("Delhi");

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");
 addr.setCtn(ctn);//HAS-A link between Objects.

 Student s=new Student();
 s.setId(202);
 s.setName("Sankar");
 s.setAddr(addr);//Link between Objects.

 Course c1=new Course("webservices",234.2,6);
 Course c2=new Course("hibernate",231.2,4);
 Course c3=new Course("Spring",201.2,8);
//JDK7 Collection-Type Inference (empty <> operator)

 List<Course> list=new ArrayList<>();
 list.add(c1);
 list.add(c2);
 list.add(c3);
 s.setCourses(list);
 return s;
}

@Path("/link1")
@GET
//@Produces("application/json")
//@Produces("application/xml")
//Request (header Param: Accept = application/xml)
@Produces({"application/json","application/xml"})
public Student showData1() {
 /*Country ctn=new Country();
 ctn.setCode("3vrgr");
 ctn.setCapital("Delhi");

 Address addr=new Address();
 addr.setHNo("2-62");
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
addr.setLoc("HYD");
addr.setCity("Ammerpet");
addr.setCtn(ctn); //HAS-A link between Objects.*/

Student s=new Student();
s.setSId(202);
s.setSName("Sankar");
//s.setAddr(addr); //Link between Objects.

/*Course c1=new Course("webservices",234.2,6);
Course c2=new Course("hibernate",231.2,4);
Course c3=new Course("Spring",201.2,8);
//JDK7 Collection-Type Inference (empty <> operator)

List<Course> list=new ArrayList<>();
list.add(c1);
list.add(c2);
list.add(c3);
s.setCos(list);*/
return s;
}
}
```

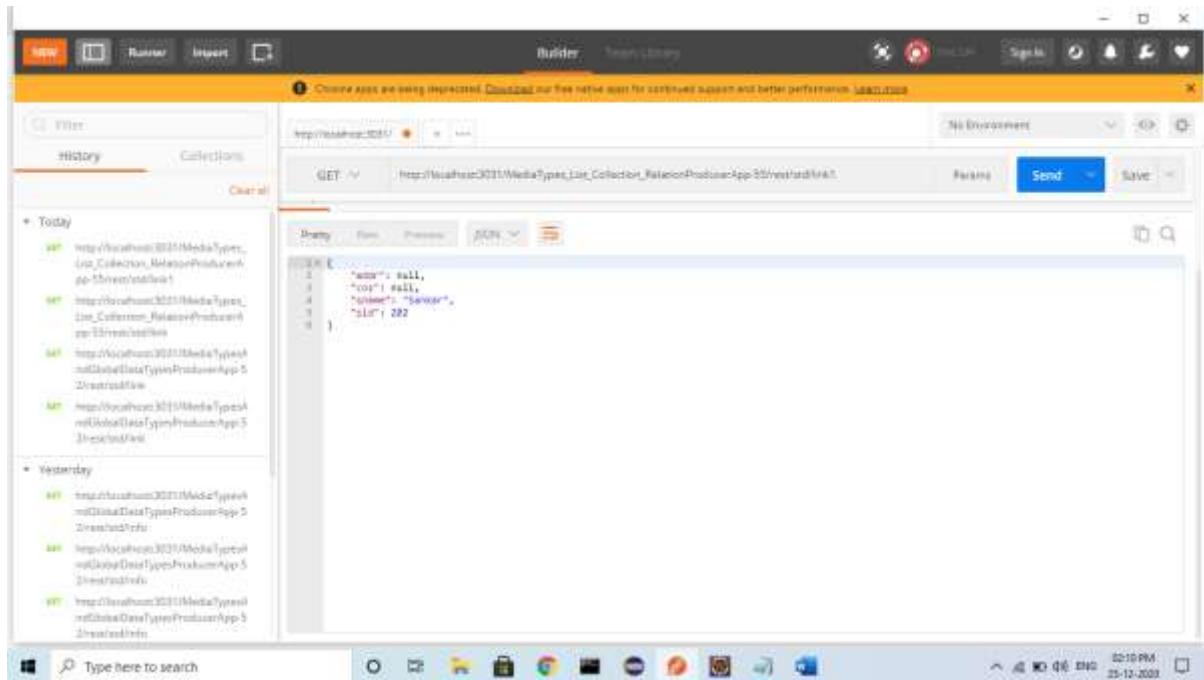
-----Output-----

In Chrome:

```
<student>
<SId>202</SId>
<SName>Sankar</SName>
</student>
```

---

-----POSTMAN TOOL-----



### -----Example MediaTypes ProducerApp-----

- If return type of method is void or returns null, in this case NO OUTPUT is given to ConsumerApp.
- Then HTTP Status is: 204 NO CONTENT.
- Note: We can even apply `@XmlElementWrapper` over collection type

In pom.xml:

```

<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

```

```

<dependencies>
 <!--

```

<https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet> -->

```

 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>

```

```
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson --
>
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime --
>
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!--
https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-impl --
>
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!--
https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-core --
>
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model classes:

Address:

```
package in.nit.model;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
```

```
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Address {
 private String hNo;
 private String loc;
 private String city;
 //class as DataType->creating one variable.
 private Country ctn; //HAS-A Relation.
}
```

Country:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Country {
 private String code;
 private String capital;
}
```

Course:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Course {
 private String cname;
 private Double cfee;
 private int duration;
}
```

}

Student:

```
package in.nit.model;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Student {
 @XmlTransient
 private int sld;
 private String sName;
 //one class has DataType -> Creating variable
 private Address addr;//HAS-A Relation
 @XmlElementWrapper(name = "courses")
 @XmlElement(name = "course")
 private List<Course> cos;//HAS-A Relation
}
```

ProducerController:

```
package in.nit.controller;
```

```
import java.util.ArrayList;
import java.util.List;
```

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
```

```
import javax.ws.rs.Produces;

import in.nit.model.Address;
import in.nit.model.Country;
import in.nit.model.Course;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 //@Produces("application/json")
 //@Produces("application/xml")
 //Request (header Param: Accept = application/xml)
 @Produces({"application/json","application/xml"})
 public Student showData() {
 Country ctn=new Country();
 ctn.setCode("3vrgr");
 ctn.setCapital("Delhi");

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
 addr.setCity("Ammerpet");
 addr.setCtn(ctn);//HAS-A link between Objects.

 Student s=new Student();
 s.setId(202);
 s.setName("Sankar");
 s.setAddr(addr);//Link between Objects.

 Course c1=new Course("webservices",234.2,6);
 Course c2=new Course("hibernate",231.2,4);
 Course c3=new Course("Spring",201.2,8);
 //JDK7 Collection-Type Inference (empty <> operator)

 List<Course> list=new ArrayList<>();
 list.add(c1);
 list.add(c2);
 list.add(c3);
 s.setCourses(list);
 return s;
 }
}
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

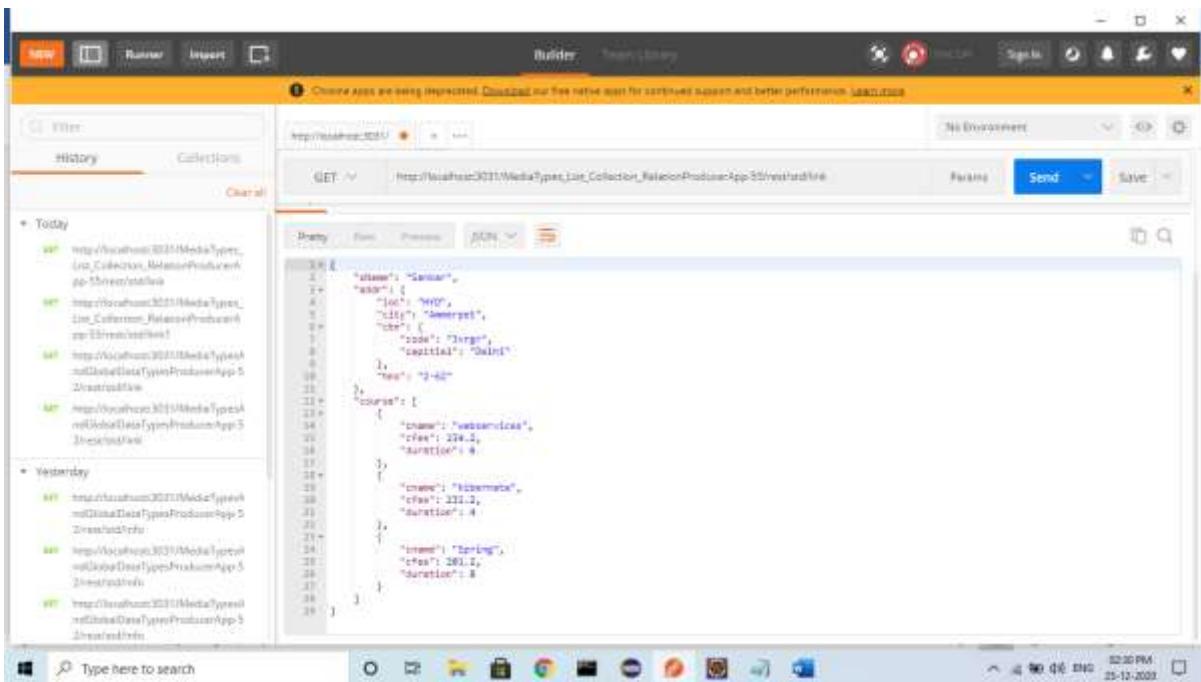
}

-----output-----

[http://localhost:3031/MediaTypes\\_List\\_Collection\\_RelationProducerApp-55/rest/std/link](http://localhost:3031/MediaTypes_List_Collection_RelationProducerApp-55/rest/std/link)

```
<student>
<sName>Sankar</sName>
<addr>
<city>Ammerpet</city>
<ctn>
<capital>Delhi</capital>
<code>3vrgr</code>
</ctn>
<HNo>2-62</HNo>
<loc>HYD</loc>
</addr>
<courses>
<course>
<cfee>234.2</cfee>
<cname>webservices</cname>
<duration>6</duration>
</course>
<course>
<cfee>231.2</cfee>
<cname>hibernate</cname>
<duration>4</duration>
</course>
<course>
<cfee>201.2</cfee>
<cname>Spring</cname>
<duration>8</duration>
</course>
</courses>
</student>
```

[http://localhost:3031/MediaTypes\\_List\\_Collection\\_RelationProducerApp-55/rest/std/link](http://localhost:3031/MediaTypes_List_Collection_RelationProducerApp-55/rest/std/link)




---

Student:

**package** in.nit.model;

**import** java.util.List;

**import** javax.xml.bind.annotation.XmlAccessType;  
**import** javax.xml.bind.annotation.XmlAccessorType;  
**import** javax.xml.bind.annotation.XmlElement;  
**import** javax.xml.bind.annotation.XmlElementWrapper;  
**import** javax.xml.bind.annotation.XmlRootElement;  
**import** javax.xml.bind.annotation.XmlTransient;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

**import** lombok.AllArgsConstructorConstructor;  
**import** lombok.Data;  
**import** lombok.NoArgsConstructor;

```

@Data
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
@AllArgsConstructorConstructor
@NoArgsConstructor
public class Student {
```

```
//@XmlTransient
@JsonIgnore
private int std;
private String sName;
//one class has DataType -> Creating variable
private Address addr;//HAS-A Relation
@XmlElementWrapper(name = "courses")
@XmlElement(name = "course")
private List<Course> cos;//HAS-A Relation
}
```

ProducerController:

```
package in.nit.controller;

import java.util.ArrayList;
import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import in.nit.model.Address;
import in.nit.model.Country;
import in.nit.model.Course;
import in.nit.model.Student;

@Path("/std")
public class StudentProducerController {
 @Path("/link")
 @GET
 // @Produces("application/json")
 // @Produces("application/xml")
 //Request (header Param: Accept = application/xml)
 @Produces({"application/json", "application/xml"})
 public Student showData() {
 Country ctn=new Country();
 ctn.setCode("3vrgr");
 ctn.setCapitil("Delhi");

 Address addr=new Address();
 addr.setHNo("2-62");
 addr.setLoc("HYD");
```

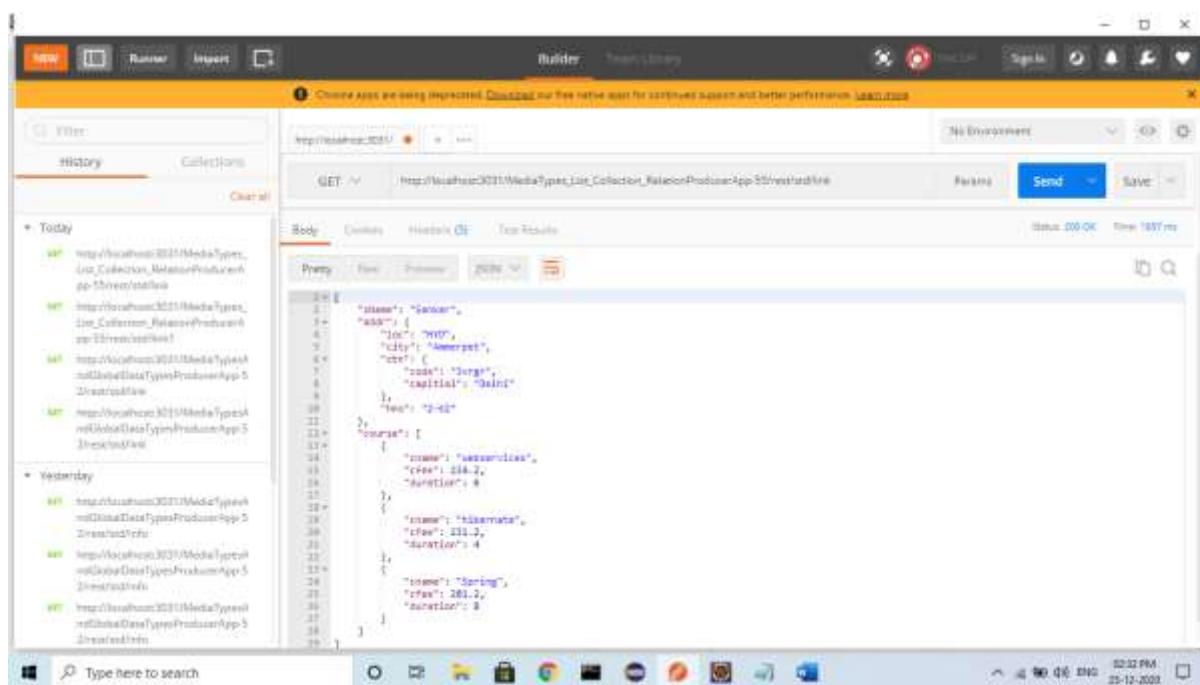
```
addr.setCity("Ammerpet");
addr.setCtn(ctn); //HAS-A link between Objects.
```

```
Student s=new Student();
s.setSId(202);
s.setSName("Sankar");
s.setAddr(addr); //Link between Objects.
```

```
Course c1=new Course("webservices",234.2,6);
Course c2=new Course("hibernate",231.2,4);
Course c3=new Course("Spring",201.2,8);
//JDK7 Collection-Type Inference (empty <> operator)
```

```
List<Course> list=new ArrayList<>();
list.add(c1);
list.add(c2);
list.add(c3);
s.setCos(list);
return s;
}
}
```

-----output-----



Note: `@XmlTransient`

`@JsonIgnore` those annotations are used for Ignore Value or data.

### Assignment(Token8):

Q) Define one JAX-RS application using  
Module Name : Company  
That has an operation : showServiceInfo()  
Which takes inputs like type, id, name.

- If Type is admin/ADMIN-then return Admin Model(aid, fname) class Object.
- Else if Type is dev/DEV-then return Developer Model (did, dname) class Object.
- Else return Error Object (id, message) [message=type+-NOT FOUND with name=+name].
- If Type is null then return Status-500 with Error Object (id, message) ["Invalid type specified"].
- If any Type is matched then return 200 else return 400,
- Should support both XML/JSON Output.

-----CompanyProducerApp-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>jersey-hk2</artifactId>
<version>2.30</version>
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson -->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime -->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
```

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model Classes:

Admin:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@NoArgsConstructor
@AllArgsConstructor
@XmlRootElement
```

```
public class Admin {
 private int aid;
 private String aname;
 private String type;
}
```

Developer:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.AllArgsConstructorConstructor;
import lombok.Data;
import lombok.NoArgsConstructorConstructor;
@Data
@NoArgsConstructorConstructor
@AllArgsConstructorConstructor
@XmlRootElement
public class Developer {
 private int did;
 private String dname;
 private String type;
}
```

ErrorType:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.AllArgsConstructorConstructor;
import lombok.Data;
import lombok.NoArgsConstructorConstructor;
@Data
@NoArgsConstructorConstructor
@AllArgsConstructorConstructor
@XmlRootElement
public class ErrorType {
 private int id;
 private String message;
}
```

### CompanyProducerController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Admin;
import in.nit.model.Developer;
import in.nit.model.ErrorType;
@Path("/cmp")
public class CompanyRestController {
 //...rest/cpm/info?id=202&.....
 @Path("/info")
 @GET
 //@Produces(MediaType.APPLICATION_JSON)
 @Produces({MediaType.APPLICATION_JSON,MediaType.
APPLICATION_XML})
 public Response showServiceInfo(
 @QueryParam("type")String type,
 @QueryParam("id")int id,
 @QueryParam("name")String name
) {
 Response resp=null;
 try {
 if(id<=0 || name==null || "".equals(name.trim())) {
 resp=Response.status(400)
 .entity(new ErrorType(-
1,"Invalid Id, name are given"))
 .build();
 }
 else if(type.equalsIgnoreCase("ADMIN")) {
 resp=Response.status(200)
 .entity(new
Admin(id,name,type))
 .build();
 }
 else if(type.equalsIgnoreCase("DEV")) {
```

```
 resp=Response.status(Status.OK)
 .entity(new
Developer(id,name,type))
 .build());
 }
 else {//Type as no matched value.

 resp=Response.status(Status.BAD_REQUEST)
 .entity(new
ErrorType(id,type+"NOT FOUND, with name"+name))
 .build());
 }
 } catch (Exception e) {//type as null

 resp=Response.status(Status.INTERNAL_SERVER_ERRO
R)
 .entity(new ErrorType(-1,"Invalid Type
Specified....."))
 .build();
 }

 return resp;
 }
}
```

-----Output-----

In Chrome:

<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info>

```
<errorType>
<id>-1</id>
<message>Invalid Id, name are given</message>
</errorType>
```

<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info?type=ADMIN>

```
<errorType>
<id>-1</id>
<message>Invalid Id, name are given</message>
</errorType>
```

<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info?type=ADMIN&id=6&name=Sankar>

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
<admin>
<aid>6</aid>
<aname>Sankar</aname>
<type>ADMIN</type>
</admin>
```

<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info?type=DEV&id=6&name=Sankar>

```
<developer>
<did>6</did>
<dname>Sankar</dname>
<type>DEV</type>
</developer>
```

<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info?type=QA>

```
<errorType>
<id>-1</id>
<message>Invalid Id, name are given</message>
</errorType>
```

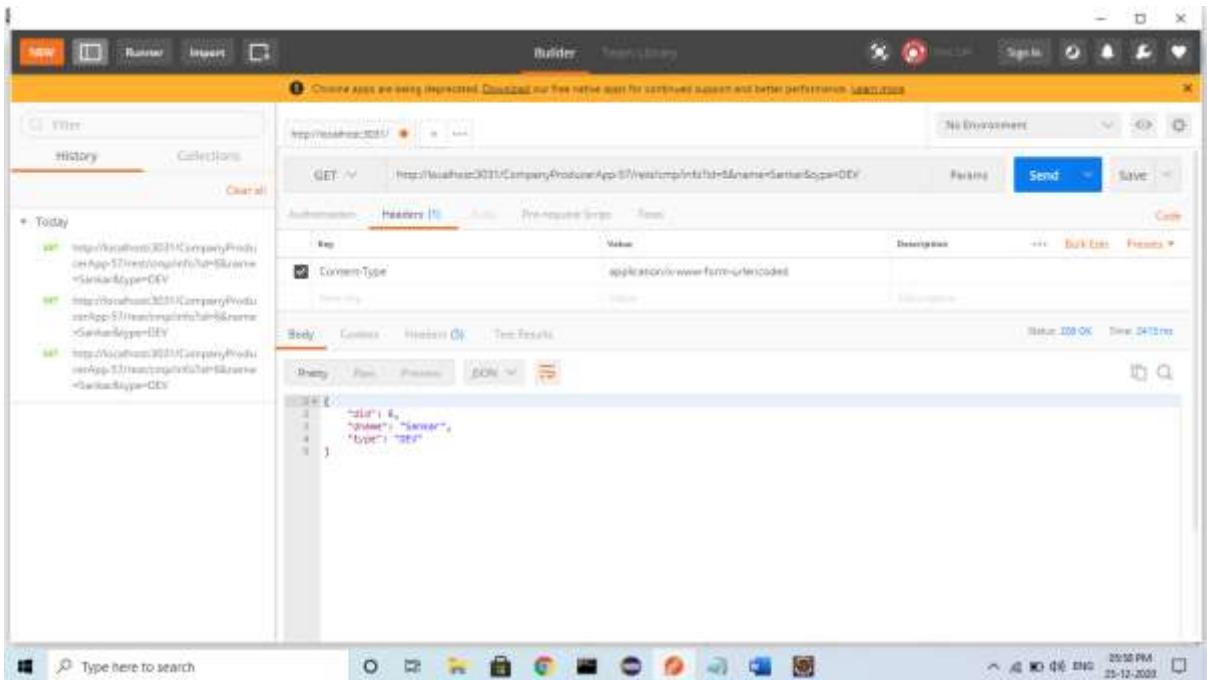
<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info?id=6&name=Sankar>

```
<errorType>
<id>-1</id>
<message>Invalid Type Specified.....</message>
</errorType>
```

<http://localhost:3031/CompanyProducerApp-57/rest/cmp/info?id=6&name=Sankar&type=QA>

```
<errorType>
<id>6</id>
<message>QANOT FOUND, with nameSankar</message>
</errorType>
```

-----POSTMAN SCREEN-----



- @Consumes:
- This MediaType annotation works on HTTP Request.
- Header Param should contain
- Example: Content-Type: application/json
- Body should contain data
  - {
  - Key : value
  - }
- Here, @Consumes read data from HTTP Request Body and converts into Object format which is given as Input to RestController method.

-----StudentProducerApp-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with JSON -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with XML -->
 <!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
 <dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
 <dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
 <dependency>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<groupId>com.sun.xml.bind</groupId>
<artifactId>jaxb-core</artifactId>
<version>2.3.0</version>
</dependency>

<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.16</version>
<scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In Model class:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;
import lombok.Data;
@Data
@XmlRootElement
public class Student {
private int sid;
private String sname;
private String sfees;
}
```

StudentRestController:

```
package in.nit.controller;

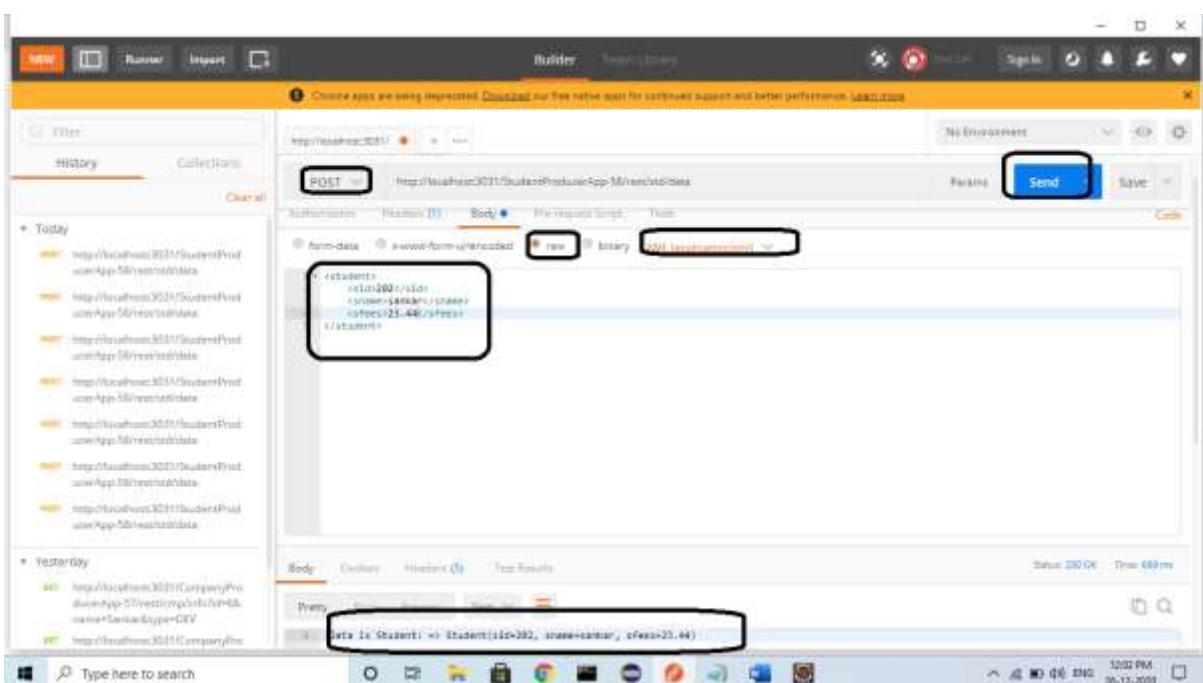
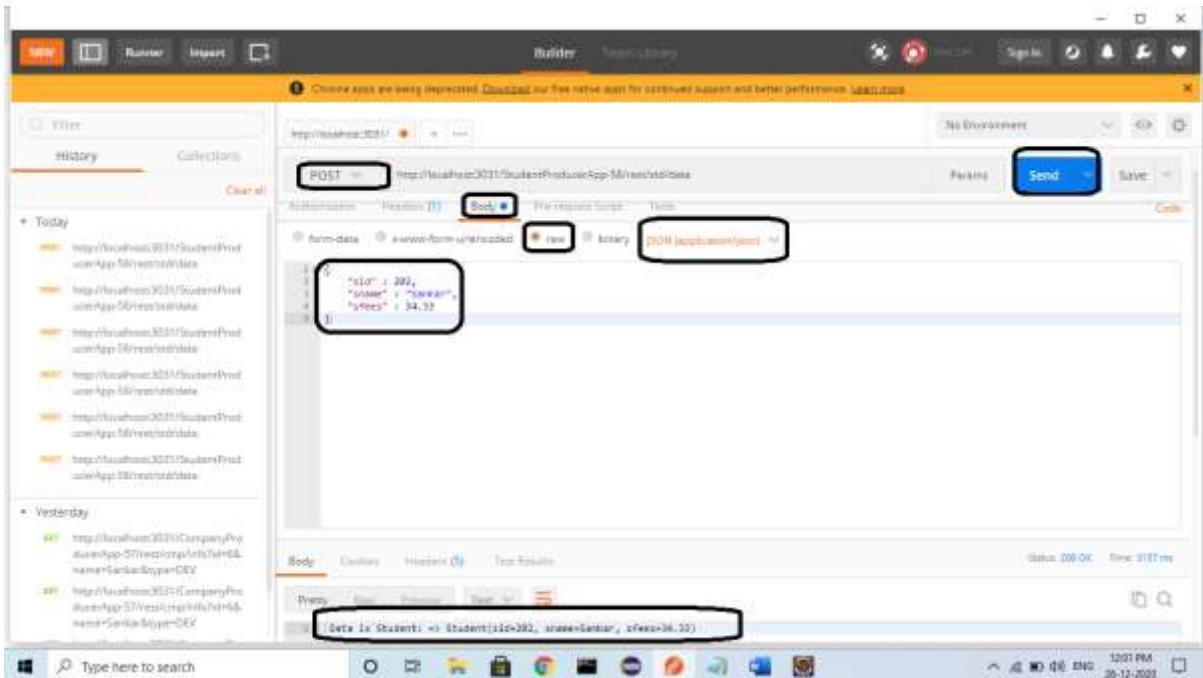
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

import in.nit.model.Student;
@Path("/std")
public class StudentRestController {
 // @GET // it is not supported Request Body
 @Path("/data")
 @POST
 @Consumes("application/xml,application/json")
 public String readInfo(Student student) {
 return " Data is Student: => "+student;
 }
}
```

-----Output-----

In chrome:  
Not possible to see output:

-----In POSTMAN SCREEN-----  
<http://localhost:3031/StudentProducerApp-58/rest/std/data>



### Assignment(Token9):

Using JAX-RS Jersey API

Module Name: Student

Operation : printResult()

Input :

Student(sid,name,marks:Map<String,Integer>,faculty>List<String>).

Example: {

```
 "sid" : 1001,
 "name" : "Sankar",
 "marks" : {"ENG":85,"MAT":65,"SCI":95},
 "faculty" : ["ABC","MNO","PQR"]
}
```

Output: String Output:

ID:\_\_\_\_\_

Name:\_\_\_\_\_

Total Marks:\_\_\_\_\_, AVG:\_\_\_\_\_

Result:\_\_\_\_\_, GRADE:\_\_\_\_\_

Not faculties:\_\_\_\_\_

Conditions:

Total Marks = s1+s2+s3;

Avg =total/3;

Result : PASS If each sub marks>40, else FAIL.

Grade : If PASS Display one grade else—NA----

```
avg: >75 A+
avg: 60 to 75 A
avg: 45 to 59 B+
avg: 40 to 44 B
```

-----Producer App-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <!--
```

[https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-  
container-servlet -->](https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet)

```
 <dependency>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<groupId>org.glassfish.jersey.containers</groupId>
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson --
>
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>
```

```
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model class:

Student class:

```
package in.nit.model;
```

```
import java.util.List;
import java.util.Map;

import lombok.Data;
@Data
public class Student {
private int sid;
private String sname;
private Map<String, Integer> marks ;
private List<String> faculties;
}
```

Java Assignment Operator:

Adding both values +=

Example:

```
package in.test;
```

```
public class AssignmetOperator {

 public static void main(String[] args) {

 int a=10;
 a+=5;//here both values adding.
 System.out.println(a);//15

 }
}
```

In StudentProducerController:

```
package in.nit.controller;

import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
```

```
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

import in.nit.model.Student;
@Path("/std")
public class StudentRestController {
@Path("/result")
@POST
@Consumes("application/xml,application/json")
public String printResult(Student student) {
 int total=0;
 double avg=0.0;
 String result="PASS";
 String grade=null;
 int count=0;

 List<String> facultyList=student.getFaculties();
 if(facultyList!=null) {
 count=facultyList.size();
 }
 Map<String,Integer> marksMap=student.getMarks();
 Collection<Integer> values=marksMap.values();
 Iterator<Integer> itr=values.iterator();
 while(itr.hasNext()) {
 int v=itr.next();
 total+=v;
 if(v<40) {
 result="FAILD";
 }
 avg=total/3.0f;

 if(result.equals("PASS")) {
 if(avg>=75.0) {
 grade="A+";
 }
 else if(avg>=60.0 && avg<=75.0) {
 grade="A";
 }
 else if(avg>=45.0 && avg<=60.0) {
 grade="B+";
 }
 else if(avg>40.0 && avg<=45.0) {
```

```
 grade="B";
 }
}
else {
 grade="---NA---";
}
}

StringBuffer sb=new StringBuffer();
sb.append(" ID= ").append(student.getId())
.append(" , NAME= ").append(student.getName())
.append(" , Total Marks= ").append(total)
.append(" , Average= ").append(avg)
.append(" , Result= ").append(result)
.append(" , Grade= ").append(grade)
.append(" , No Faculties= ").append(count);

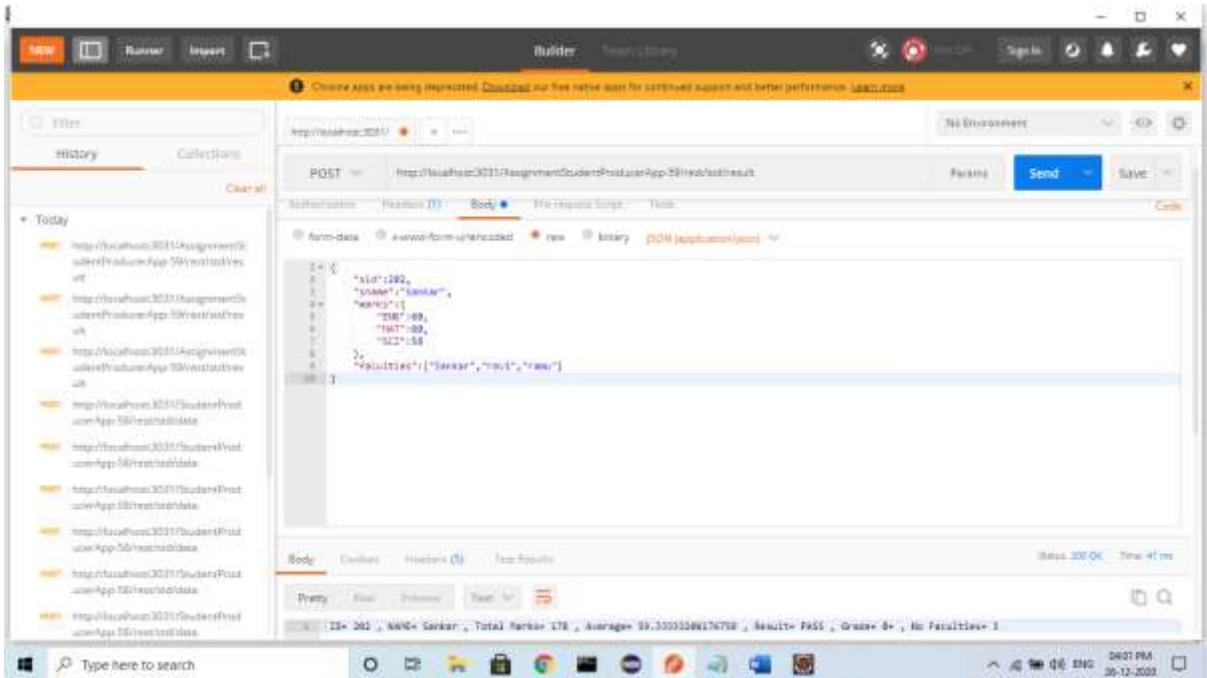
return sb.toString();
}
}
```

-----Output-----

In chrome:

Not possible to see output in Chrome:

-----POSTMAN SCREEN-----



- DataType is Selected by Java Compiler.  
→ That reduces burden to Programmer.

-----Example Modified Controller-----

```
package in.nit.controller;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

import in.nit.model.Student;
@Path("/std")
public class StudentRestController {
@Path("/result")
@POST
@Consumes("application/xml,application/json")
public String printResult(Student student) {
//JDK10 Local Variable type reference.
var total=0;
var avg=0.0;
var result="PASS";
var grade="";
var count=0;
```

```
var facultyList=student.getFaculties();
if(facultyList!=null) {
 count=facultyList.size();
}
var marksMap=student.getMarks();
var values=marksMap.values();
var itr=values.iterator();
while(itr.hasNext()) {
 var v=itr.next();
 total+=v;
 if(v<40) {
 result="FAILD";
 }
 avg=total/3.0f;

 if(result.equals("PASS")) {
 if(avg>=75.0) {
 grade="A+";
 }
 else if(avg>=60.0 && avg<=75.0) {
 grade="A";
 }
 else if(avg>=45.0 && avg<=60.0) {
 grade="B+";
 }
 else if(avg>40.0 && avg<=45.0) {
 grade="B";
 }
 }
 else {
 grade="---NA---";
 }
}
```

```
var sb=new StringBuffer();
sb.append(" ID= ").append(student.getId())
.append(" , NAME= ").append(student.getName())
.append(" , Total Marks= ").append(total)
.append(" , Average= ").append(avg)
.append(" , Result= ").append(result)
.append(" , Grade= ").append(grade)
.append(" , No Faculties= ").append(count);
```

```
 return sb.toString();
 }
}
```

### -----POSTMAN SCREEN-----

The screenshot shows the Postman interface. The URL is set to `http://localhost:8081/AssignmentStudentsProcessorApp-501/webapi/result`. The request method is `POST`. The body is defined as `form-data` with the key `student` and value `{"custId":1002, "custName": "Karrasankar", "marks": [90, 85, 88]}`. The response status is `200 OK` and the result is `PASS`.

### Assignment(Token10):

→ Using JAX-RS implementing JERSEY2.30.

→ Module Name: Customer

→ Operation:calBill()

→ Inputs: custId is random(),

→ custName

Items: { "pen":10.0,"car":25.0,"btl":95.0} (MAP-ItemName, Price)

Cart: {"pen":4,"car":2,"btl":1} (MAP-Itemname,qty)

Discount: [12,15,10]

→ Output: (String)

Customer id:<custid> customer Name: <custName>

All Above details+

Item-pen-<price>,car-<price>,btl-<price>

totalCost (after discount)=<price>

gst=12% of totalCost

Billing Amount= totalCost+gst.

Sample Outputs in Console::

- 1.Customer id: 410781901 Customer Name: Sankar All Above Details+  
Items{pen=10, car=25, btl=95}, TotalCost( After Discount = 35.2 Gst= 1.6896 TotalBill= 36.8896
- 2.Customer id: 309134916 Customer Name: Sankar All Above Details+  
Items{pen=10, car=25, btl=95}, TotalCost( After Discount = 42.5 Gst= 3.1875 TotalBill= 45.6875
3. Customer id: 583499605 Customer Name: Sankar All Above Details+  
Items{pen=10, car=25, btl=95}, TotalCost( After Discount = 85.5 Gst= 8.1225 TotalBill= 93.6225

-----ProducerApp-----

In pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
```

```
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson-->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime-->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-impl-->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-core-->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok-->
<dependency>
 <groupId>org.projectlombok</groupId>
```

```
<artifactId>lombok</artifactId>
<version>1.18.16</version>
<scope>provided</scope>
</dependency>

</dependencies>
```

### In web.xml File:

```
<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

### In Model class:

```
package in.nit.model;

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

import lombok.Data;

@Data
public class Customer {
 private Integer custid;
```

```
private String custname;
private Map<String, Integer> items;
private Map<String, Integer> card;;
private List<Integer> discount;
}
```

In ProducerController:

```
/*
 * package in.nit.controller;
 *
 * import java.util.Collection; import java.util.Iterator; import
 * java.util.List; import java.util.Map;
 *
 * import javax.ws.rs.Consumes; import javax.ws.rs.POST; import
 * javax.ws.rs.Path;
 *
 * import in.nit.model.Student;
 *
 * @Path("/std") public class StudentRestController {
 *
 * @Path("/result")
 *
 * @POST
 *
 * @Consumes("application/xml,application/json") public String
 * printResult(Student student) { int total=0; double avg=0.0; String
```

```
* result="PASS"; String grade=null; int count=0;
*
* List<String> facultyList=student.getFaculties(); if(facultyList!=null) {
* count=facultyList.size(); } Map<String,Integer>
marksMap=student.getMarks();
* Collection<Integer> values=marksMap.values(); Iterator<Integer>
* itr=values.iterator(); while(itr.hasNext()) { int v=itr.next(); total+=v;
* if(v<40) { result="FAILD"; } avg=total/3.0f;
*
* if(result.equals("PASS")) { if(avg>=75.0) { grade="A+"; } else
if(avg>=60.0
* && avg<=75.0) { grade="A"; } else if(avg>=45.0 && avg<=60.0) {
grade="B+"; }
* else if(avg>40.0 && avg<=45.0) { grade="B"; } } else { grade="---NA----
"; } }
*
* StringBuffer sb=new StringBuffer();
* sb.append(" ID= ").append(student.getId())
* .append(" , NAME= ").append(student.getName())

```

package in.nit.controller;

```
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Random;

import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

import in.nit.model.Customer;
@Path("/cust")
public class CustomerRestController {
@Path("/bill")
@POST
@Consumes("application/xml,application/json")
public String calBill(Customer customer) {
 double totalCost=0;
 double gst=0;
 double totalBill=0;
 double totalQt=0;
 double pamt=0;
 String sb=null;
 Map<String,Integer> items=customer.getItems();
 Collection<Integer> prices=items.values();
 Iterator<Integer> itr=prices.iterator();
```

```
Map<String, Integer> card=customer.getCard();
```

```
Collection<Integer> qt=card.values();
```

```
Iterator<Integer> itr1=qt.iterator();
```

```
List<Integer> discount=customer.getDiscount();
```

```
Iterator<Integer> itr2=discount.iterator();
```

```
while(itr.hasNext()) {
```

```
 while(itr1.hasNext()) {
```

```
 while(itr2.hasNext()) {
```

```
 double dsAmt=itr2.next();
```

```
 totalCost=itr.next();
```

```
 totalQt=itr1.next();
```

```
 pamt=totalCost*totalQt;
```

```
 dsAmt=pamt*dsAmt/100;
```

```
 double amt=pamt-dsAmt;
```

```
 totalCost=amt;
```

```
 gst=totalCost*dsAmt/100.0;
```

```
 totalBill=totalCost+gst;
```

```
Random r=new Random();
```

```
Integer random=r.nextInt(1234567890);
```

```
customer.setCustid(random);
```

```
sb= new StringBuffer()
```

```
 .append(" Customer id:
").append(customer.getCustid())

 .append(" Customer Name:
").append(customer.getCustname())

 .append(" All Above Details+ ").append(" Items")
 .append(items).append(",")

 .append(" TotalCost(After Discount =
").append(totalCost)

 .append(" Gst= ").append(gst)

 .append(" TotalBill= ").append(totalBill)

 .toString();

 System.out.println(sb);

}
}
}

return sb;

}
}
```

-----Output-----

In chrome Not possible to see output.

In Console:

Customer id: 410781901 Customer Name: Sankar All Above Details+  
Items{pen=10, car=25, btl=95}, TotalCost( After Discount = 35.2 Gst=  
1.6896 TotalBill= 36.8896

Customer id: 309134916 Customer Name: Sankar All Above Details+  
Items{pen=10, car=25, btl=95}, TotalCost( After Discount = 42.5 Gst=  
3.1875 TotalBill= 45.6875

Customer id: 583499605 Customer Name: Sankar All Above Details+  
Items{pen=10, car=25, btl=95}, TotalCost( After Discount = 85.5 Gst=  
8.1225 TotalBill= 93.6225

-----POSTMAN SCREEN-----

The screenshot shows the Postman interface with a POST request to `http://localhost:3011/Assignment/CustomerProducerApp-01/rest/customer`. The request body is a JSON object:

```
{"Customer": "Sankar", "Items": [{"pen": 10, "car": 25, "btl": 95}, {"pen": 14, "car": 24, "btl": 98}], "Discount": [12, 15, 28]}
```

The response status is 200 OK, and the response body is:

```
Customer Id: 583499605 Customer Name: Sankar All Above Details+ Items{pen=10, car=25, btl=95}, TotalCost(After Discount = 85.5 Gst= 8.1225 TotalBill= 93.6225}
```

## Working with both: @Produces and @Consumes:

Assignment(Token11):

Define one JAX-RS Application for Service: Billing Process

Having method `calculateInvoiceData()`

That takes below inputs: Cart and returns Outputs: Invoice as given in JSON/XML format.

Input:

Cart ([cid:int,code:String,parts>List](#))

Part (`pid:int,pcode:String,pcost:double,qty:int,discPer:double`)

Output:

Invoice(SaleInfo:List,code:String,cid:int,totalPrice:double,gst:double,BillingAmount:double)

SaleInfo(pcode:String,pocst:double,lineAmt:double,discount:double,lineValue:double).

Calculations:

lineAmt =pcost\*qty;

discount =lineAmt\*discPer/100.0;

linevalue =lineAmt-discount;

totalPrice=Sum of (lineValues);

gst=12% of totalPrice;

BillingAmount=totalPrice+gst;

---

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 </dependency>
```

```
<version>2.30</version>
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson-->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok-->
<dependency>
```

```
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.16</version>
<scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model classes:

```
package in.nit.model;
```

```
import lombok.Data;
```

```
@Data
```

```
//Input
```

```
public class Part {
```

```
 private Integer pid;
 private String pcode;
 private Double pcost;
 private Integer qty;
```

```
 private Double discPer;
}

package in.nit.model;

import java.util.List;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.Data;
@Data
@XmlRootElement
//Input
public class Cart {
private Integer cid;
private String code;
private List<Part> parts;
}

package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
//Output
public class SaleInfo {
private String pcode;
private Double pcost;
private Double lineAmt;
private Double discount;
private Double lineValue;
}

package in.nit.model;

import java.util.List;
```

```
import javax.xml.bind.annotation.XmlRootElement;
```

```
import lombok.Data;
@Data
@XmlRootElement
//Output
public class Invoice {
 private Integer cid;
 private String code;
 private List<SaleInfo> sales;
 private Double totalPrice;
 private Double gst;
 private Double billingAmount;
}
```

```
package in.nit.controller;
import java.util.ArrayList;
import java.util.List;
```

```
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
```

```
import in.nit.model.Cart;
import in.nit.model.Invoice;
import in.nit.model.Part;
import in.nit.model.SaleInfo;
@Path("/bill")
public class BillingProcessRestController {
 @Path("/call")
```

@POST

```
@Consumes({"application/json","application/xml"})
@Produces({"application/json","application/xml"})
public Invoice calculateInvoiceData(
 Cart cart
) {
 //Get All Parts Of Data From Cart.
 List<Part> parts=cart.getParts();
 Double totalprice=0.0;
 List<SaleInfo> sales=new ArrayList<>();
 for(Part p: parts) {
 //Calculate Line Values:
 Double lineAmt=p.getPcost()*p.getQty();
 Double discount=lineAmt*p.getDiscPer()/100.0;
 Double lineValue=lineAmt-discount;

 //Convert to salesInfo Object.
 SaleInfo sale=new SaleInfo(p.getPcode(),p.getPcost(),
lineAmt, discount, lineValue);
 //save in List<SaleInfo>
 sales.add(sale);
 totalprice+=lineValue;
 }

 //Calculate GST
 Double gst=totalprice*12/100.0;
 Double billingAmount=totalprice+gst;
```

//Final Output:Invoice Object.

```
Invoice inv=new Invoice();
inv.setCid(cart.getCid());
inv.setCode(cart.getCode());
inv.setSales(sales);
inv.setTotalPrice(totalprice);
inv.setGst(gst);
inv.setBillingAmount(billingAmount);
return inv;
}
```

-----output-----

In chrome:

Not possible to see output.

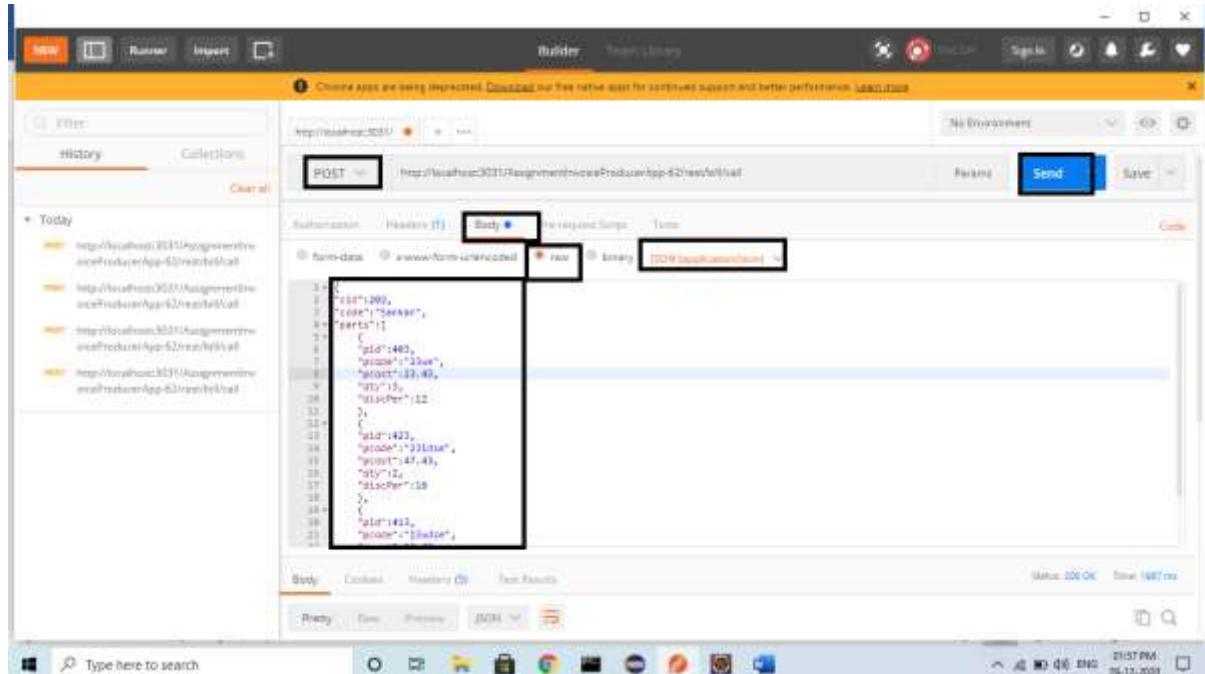
-----POSTMAN SCREEN-----

Input[JSON]:

```
{
 "cid":202,
 "code":"Sankar",
 "parts":[
 {
 "pid":403,
 "PCODE":"23we",
 "pcost":23.43,
 "qty":3,
```

```
"discPer":12
},
{
"pid":423,
"PCODE":"23idse",
"pcost":47.43,
"qty":2,
"discPer":16
},
{
"pid":413,
"PCODE":"23wdse",
"pcost":13.43,
"qty":2,
"discPer":2
}
]]
```

}

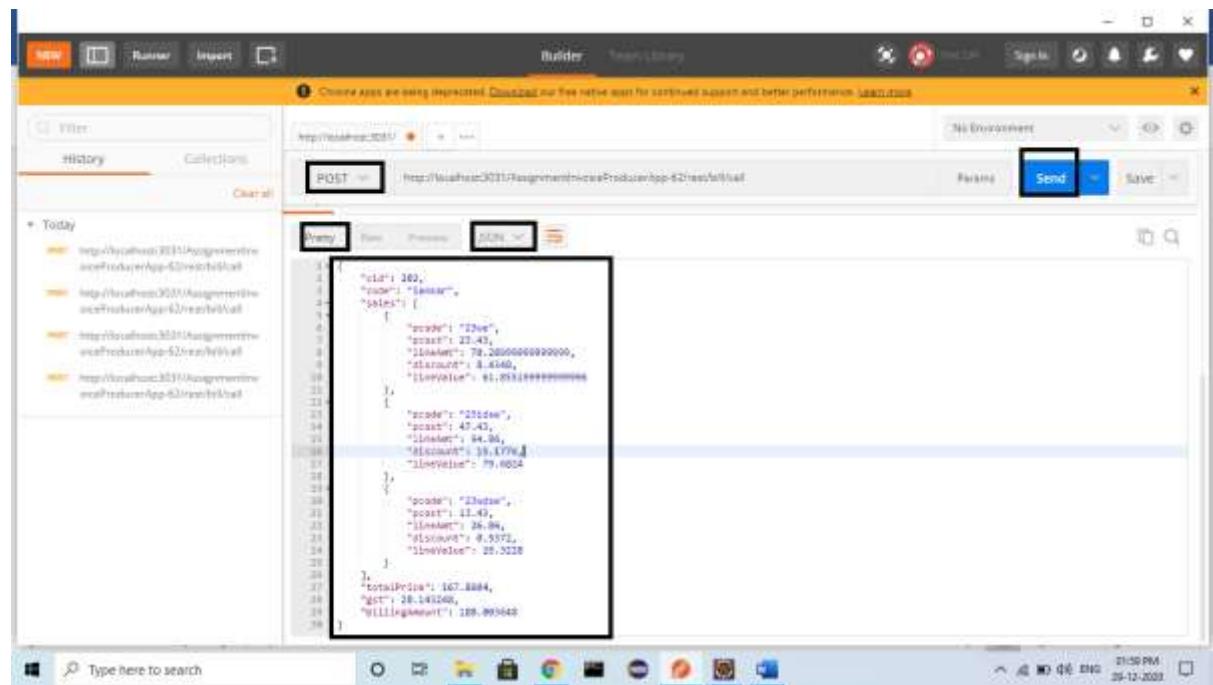


Output[JSON]:

```
{
 "cid": 202,
 "code": "Sankar",
 "sales": [
 {
 "PCODE": "23we",
 "pcost": 23.43,
 "lineAmt": 70.28999999999999,
 "discount": 8.4348,
 "lineValue": 61.85519999999999
 },
 {
 "PCODE": "23idse",
 "pcost": 47.43,
 "lineAmt": 94.86,
 "discount": 15.1776,
 "lineValue": 79.6824
 },
 {
 "PCODE": "23wdse",
 "pcost": 13.43,
 "lineAmt": 26.86,
 "discount": 0.5372,
 "lineValue": 26.3228
 }

```

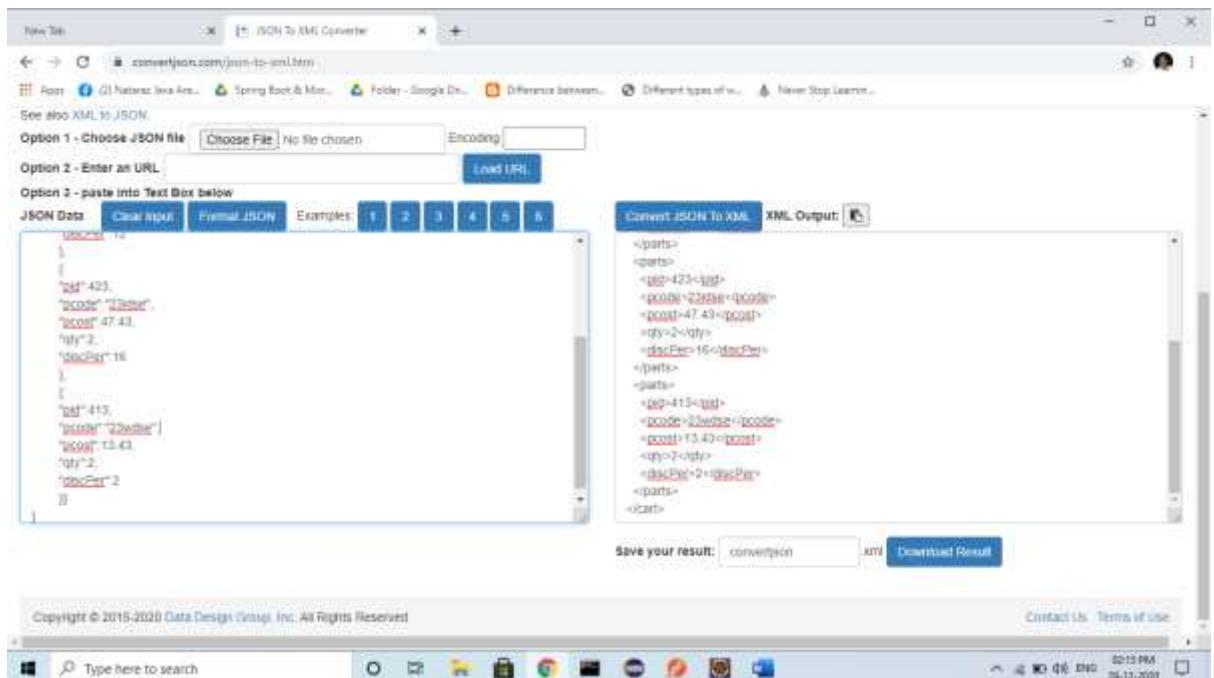
```
"totalPrice": 167.8604,
"gst": 20.143248,
"billingAmount": 188.003648
}
```



@Consumes({"application/xml,application/json"})  
@Produces({"application/xml,application/json"})

To Convert JSON to XML:

- You can write manually or use online tools like:
- <https://www.convertjson.com/json-to-xml.htm>

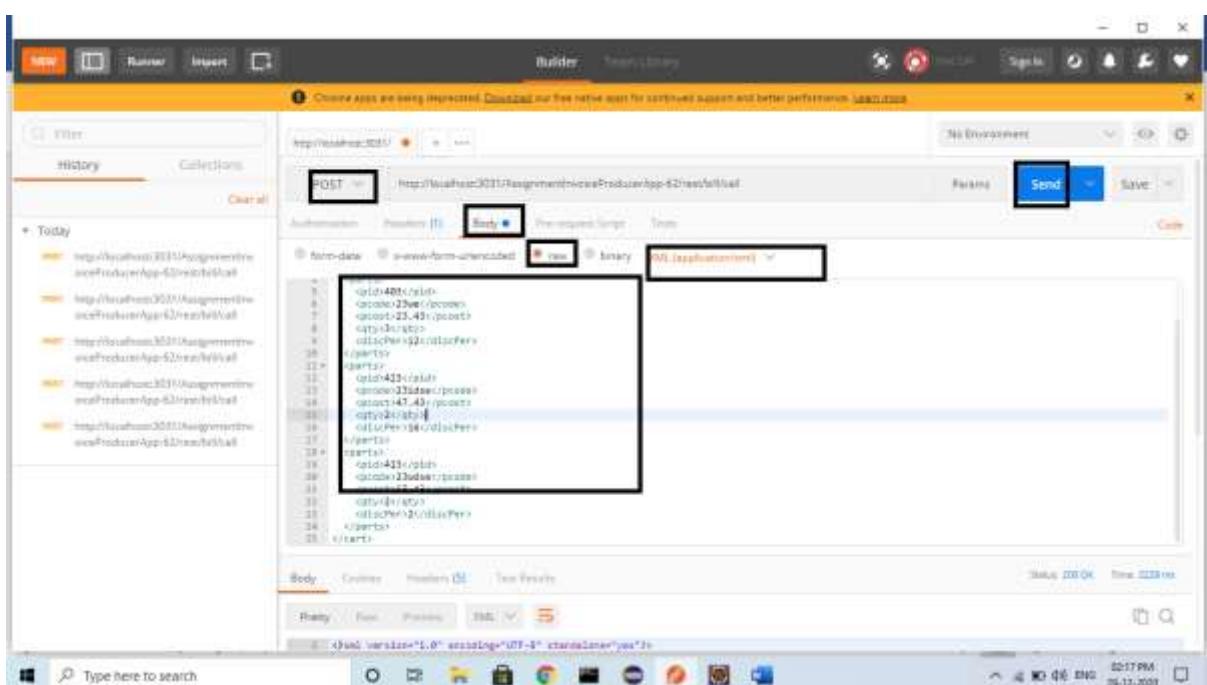
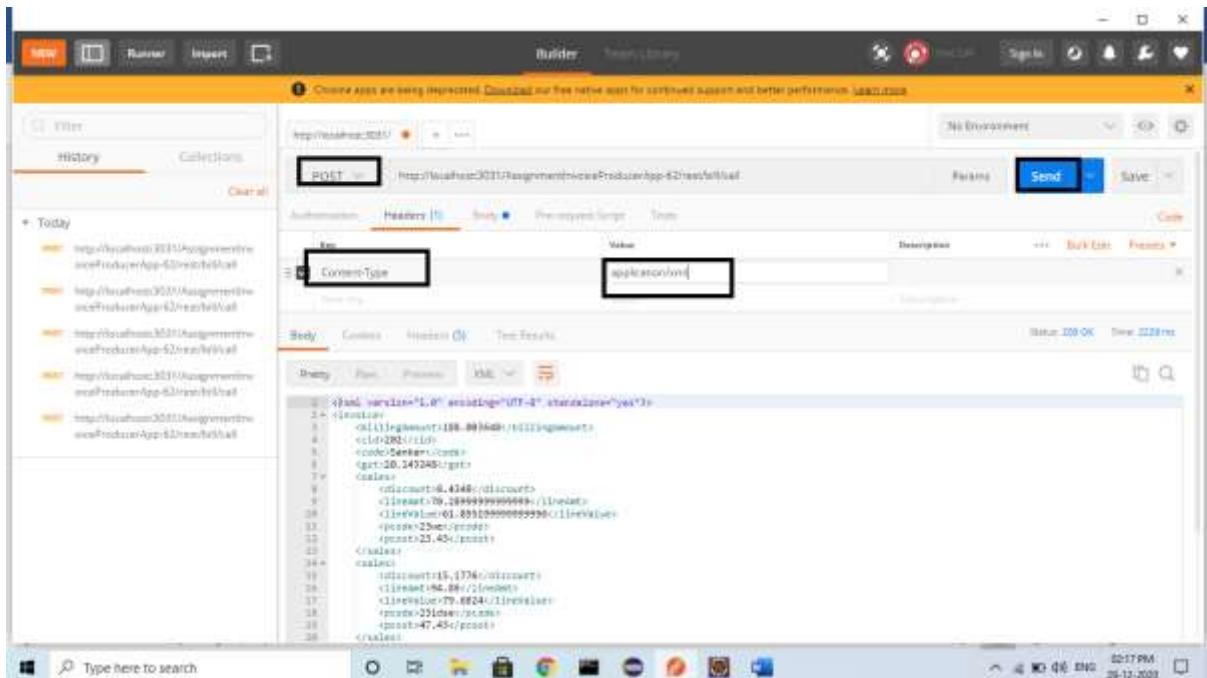


Input:[xml]

```
<cart>
 <cid>202</cid>
 <code>Sankar</code>
 <parts>
 <pid>403</pid>
 <pcode>23we</pcode>
 <pcost>23.43</pcost>
 <qty>3</qty>
 <discPer>12</discPer>
 </parts>
 <parts>
 <pid>423</pid>
 <pcode>23idse</pcode>
 <pcost>47.43</pcost>
 <qty>2</qty>
 <discPer>16</discPer>
 </parts>
 <parts>
 <pid>413</pid>
 <pcode>23wdse</pcode>
 <pcost>13.43</pcost>
 <qty>2</qty>
 </parts>
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

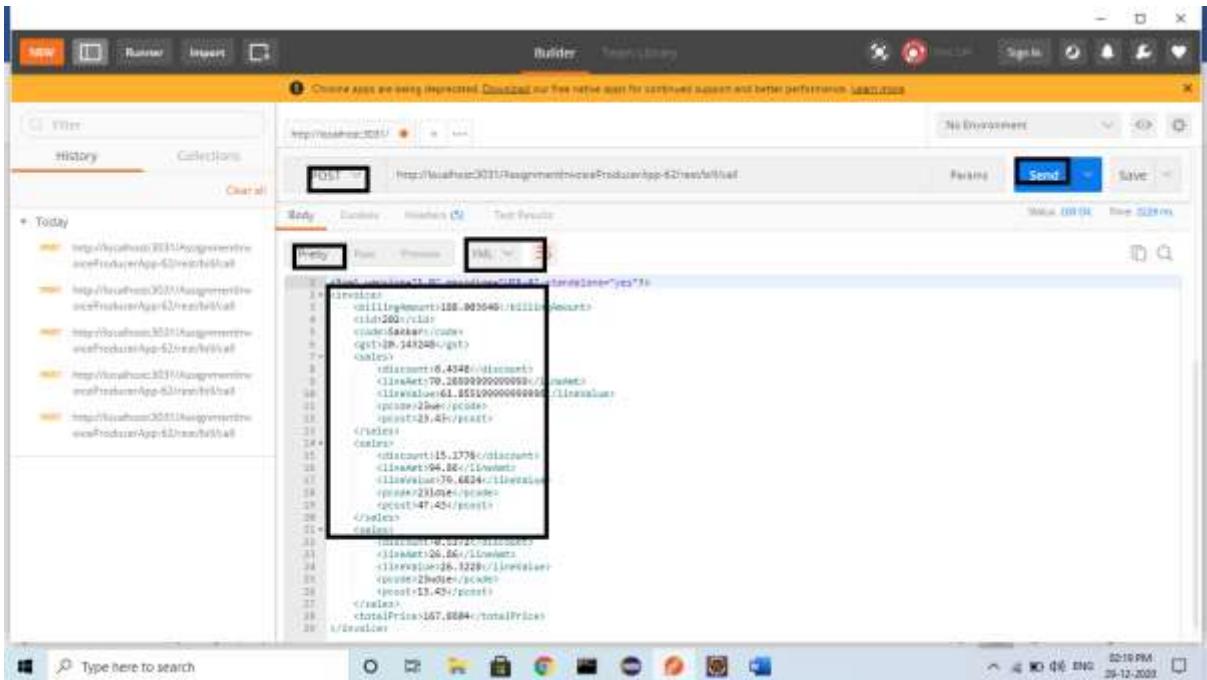
```
<discPer>2</discPer>
</parts>
</cart>
```



## Output:[xml]

```
<invoice>
 <billingAmount>188.003648</billingAmount>
```

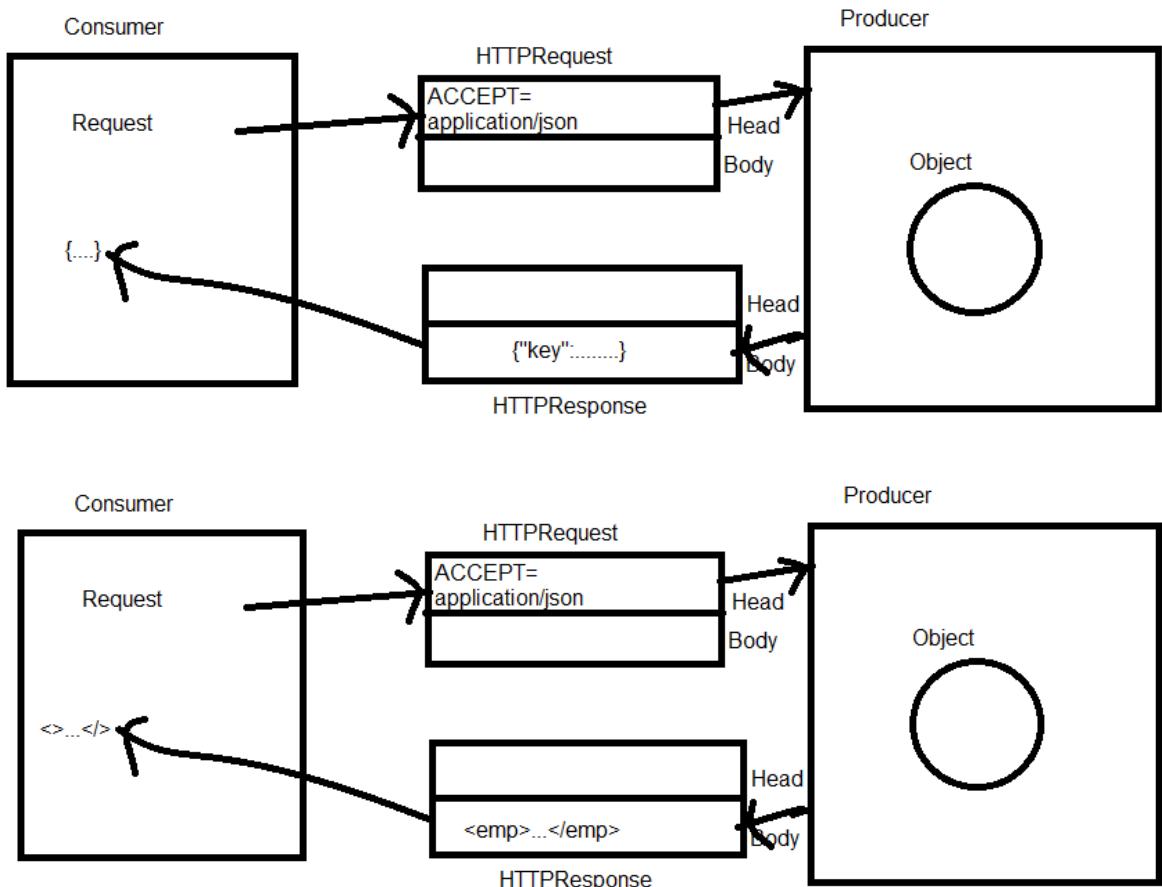
```
<cid>202</cid>
<code>Sankar</code>
<gst>20.143248</gst>
<sales>
 <discount>8.4348</discount>
 <lineAmt>70.28999999999999</lineAmt>
 <lineValue>61.85519999999996</lineValue>
 <pcode>23we</pcode>
 <pcost>23.43</pcost>
</sales>
<sales>
 <discount>15.1776</discount>
 <lineAmt>94.86</lineAmt>
 <lineValue>79.6824</lineValue>
 <pcode>23idse</pcode>
 <pcost>47.43</pcost>
</sales>
<sales>
 <discount>0.5372</discount>
 <lineAmt>26.86</lineAmt>
 <lineValue>26.3228</lineValue>
 <pcode>23wdse</pcode>
 <pcost>13.43</pcost>
</sales>
<totalPrice>167.8604</totalPrice>
</invoice>
```



## Content Negotiation in ReST:

- Jersey supports Content Negotiation using Media Types and “ACCEPT” header param.
- It is a process of returning data using MediaType based on client request in multiple types.
- Request should contain HeaderParam “ACCEPT” with one value  
Example: application/json
- If producer supports that MediaType, then data is given back to client, else HTTP-406 (Not Acceptable).

Design:



### -----Content Negotiation Producer App-----

In pom.xml File:

```

<dependencies>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
 >
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>

```

```
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson-->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime-->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-impl-->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-core-->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok-->
<dependency>
 <groupId>org.projectlombok</groupId>
```

```
<artifactId>lombok</artifactId>
<version>1.18.16</version>
<scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
<display-name>Archetype Created Web Application</display-name>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>in.nit.controller</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/rest/* </url-pattern>
</servlet-mapping>
</web-app>
```

In Model class:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import com.fasterxml.jackson.annotation.JsonIgnore;

import lombok.AllArgsConstructorConstructor;
import lombok.Data;

@Data
@XmlRootElement
@NoArgsConstructor
public class Employee {
```

```
private int empld;
private String empName;
private double empSal;
@JsonIgnore
private String secrete;
}
```

In ProducerRestController:

```
package in.nit.controller;

import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import in.nit.model.Employee;
@Path("/emp")
public class EmployeeRestController {
 @POST
 @Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
 public Employee showData() {
 Employee e=new Employee();
 e.setEmpld(678);
 e.setEmpName("Sankar");
 e.setEmpSal(66.88);
 e.setSecrete("8uhhu7");
 return e;
 }
}
```

-----output-----

In chrome:  
Not possible to see output.

-----POSTMAN SCREEN-----

The screenshot shows the Postman interface with a successful POST request to `/api/resource/101`. The request headers include an `Accept` header set to `application/xml`. The response body displays XML data:

```
XML [4.4 KB] - Accept: application/xml
<Resource>
<id>101</id>
<name>Sample Resource</name>
<version>1.0.0</version>
<created>2023-12-30T10:00:00Z</created>
```

The screenshot shows the Postman interface with a successful POST request to `/api/resource/101`. The request headers include an `Accept` header set to `Content-Type remove and search Accept`. The response body displays JSON data:

```
JSON [1.1 KB] - Content-Type: application/json
[{"result": 470, "message": "Success", "status": 470}]
```

Note:

For the above application if request has no “ACCEPT” headerParam provided then first MediaType given in order(XML) is chosen by default.

## Auto Detection of MediaType for Multi-Consumes:

- @Consumes supports different MediaTypes as input request which can be converted to Object/Collection.
- Type will be auto-detected by @Consumes by reading HTTPRequest header Param “Content-Type”.

-----Producer App-----

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- To work with JSON -->
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
```

```
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-
core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies></pre>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/* </url-pattern>
 </servlet-mapping>
</web-app>
```

In Model class:

```
package in.nit.model;

import javax.xml.bind.annotation.XmlRootElement;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@XmlRootElement
@NoArgsConstructor
public class Employee {
 private int empld;
```

```
private String empName;
private double empSal;
@JsonIgnore
private String secrete;
}
```

In ProducerRestController:

```
package in.nit.controller;
```

```
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.MediaType;

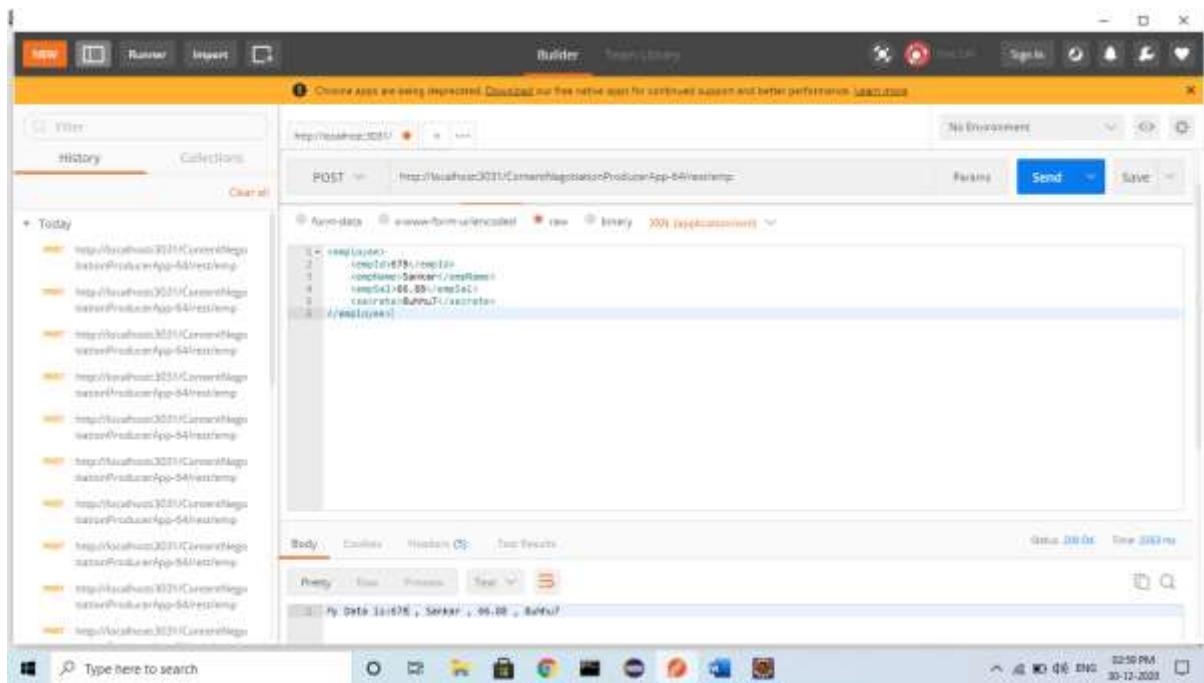
import in.nit.model.Employee;
@Path("/emp")
public class EmployeeRestController {
 @POST
 @Consumes({MediaType.APPLICATION_JSON,MediaType.APPLICATION_XML})
 public String showData(Employee e) {
 return "My Data is:"
 +e.getEmpId()+" , "
 +e.getEmpName()+" , "
 +e.getEmpSal()+" , "
 +e.getSecrete();
 }
}
```

-----output-----

In chrome:

Not possible to see.

-----POSTMAN SCREEN-----



```
<employee>
 <emplId>678</emplId>
 <empName>Sankar</empName>
 <empSal>66.88</empSal>
 <secrete>8uhhu7</secrete>
</employee>
```

To convert xml to json::

<https://codebeautify.org/xmltojson>

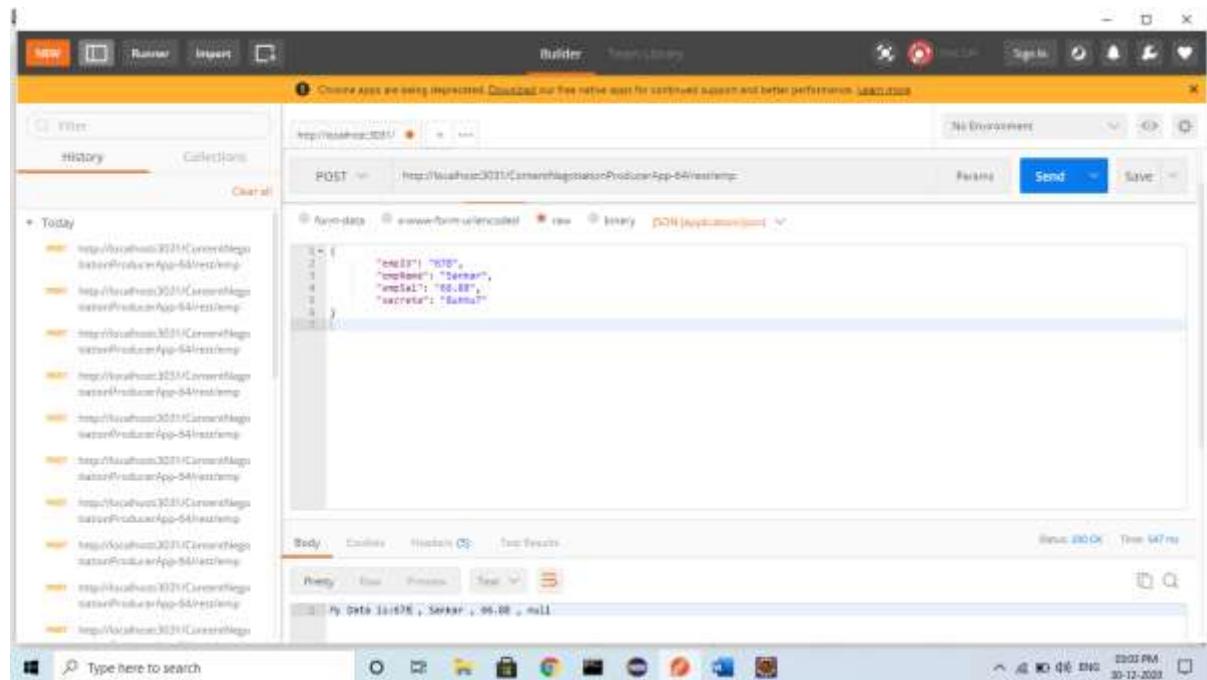
```
{
 "employee": {
 "emplId": "678",
 "empName": "Sankar",
 "empSal": "66.88",
 "secrete": "8uhhu7"
 }
}
```

```
}
```

```
}
```

Remove some data:

```
{
 "empId": "678",
 "empName": "Sankar",
 "empSal": "66.88",
 "secrete": "8uhhu7"
}
```



| @Produces |                  | @Consumes       |                  |
|-----------|------------------|-----------------|------------------|
| Headers:  |                  | Body:           |                  |
| Accept    | application/json | raw             | application/json |
| Accept    | application/xml  | application/xml | application/xml  |

---

\*\*\*Hint: Default output type for above code is: application/json

To get XML output: use Header param:

Accept: application/xml

---

Assignment(Token12):

Define one JAX-RS application for ExamResult Module

Having operation: findExamResultInfo();

That taken Input: (JSON/XML)

Student(sid,sname,marksList>List<mark>)

Mark(subName,theoryMarks,labmarks)

Given Output: (JSON/XML)

Result(sid,sname,totalMarks,avg,grade,marksInfoList>List<MarksInfo>,finalResult)

MarksInfo(subName,theoryMarks,labMarks,theoryMarksPer,labMarksPer,examInfo)

\*\*\*Conditions:

→ examInfo-PASS/FAIL condition: theoryMarks>40 and labMark>12 or

70

→ If all subj PASS ->finalResult=PASS

(if any one SUBJ Fail then finalResult=FAIL)

→ totalMarks=sum of all MarksInfo(theoryMarks+labMarks)

→ avg=totalMarks/no. of Subject [marksList.length()]

→ If (finalResult = PASS)

    Apply grade  
    Avg>=75-A+  
    Avg-60-75 A  
    Avg -45-59 B+  
    Avg -40-44 B

→ Coding Steps

1. Define Model classes with HAS-A
  2. Design Sample JSON/XML Inputs
  3. Design Sample JSON/XML Output
  4. Start writing RestController
  5. Calculate one by one values
  6. Link Objects
  7. Final Output return.
- 

-----ExamResultProducerApp-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>jersey-hk2</artifactId>
<version>2.30</version>
</dependency>
<!-- To work with JSON -->
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson -->
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- To work with XML -->
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime -->
<dependency>
 <groupId>org.glassfish.jaxb</groupId>
 <artifactId>jaxb-runtime</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-impl -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-impl</artifactId>
 <version>2.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jAXB-core -->
<dependency>
 <groupId>com.sun.xml.bind</groupId>
 <artifactId>jaxb-core</artifactId>
 <version>2.3.0</version>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
```

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In Model classes:

Mark class:

```
package in.nit.model;

import lombok.Data;

@Data
//Input
public class Mark {

 private String subName;
```

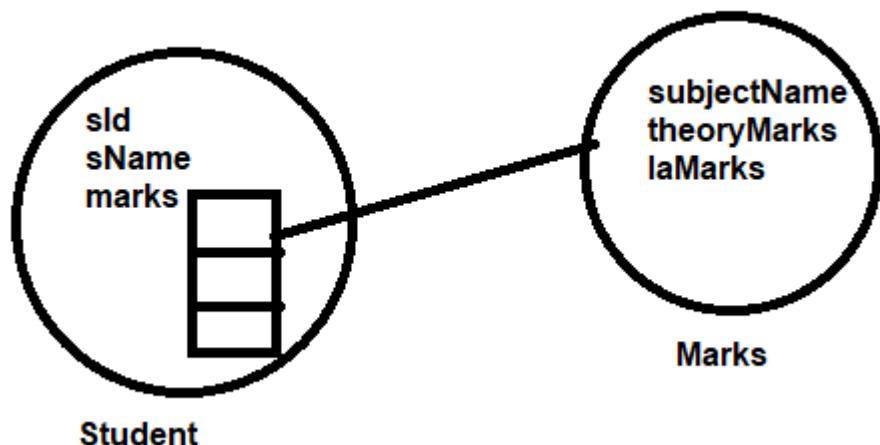
```
private Double theoryMarks;
private Double labmarks;
}
```

Input: (JSON/XML)

Student(sid,sname,marksList>List<mark>)

```
Mark(subName,theoryMarks,labMarks)
@Data
Student{
 private Integer sId;
 private String sName;
 private List<Mark> marks;//HAS-A
}
@Data
Mark{
 private String subjectName;
 private Integer theoryMarks;
 private Integer labMarks;
```

Diagram Representation:



Student class:

package in.nit.model;

```
import java.util.List;
```

```
import javax.xml.bind.annotation.XmlRootElement;
```

```
import lombok.Data;
@Data
@XmlRootElement
//Input
public class Student {
```

```
private Integer sid;
private String sname;
private List<Mark> markList;//HAS-A
}
MarksInfo class:
package in.nit.model;

import lombok.AllArgsConstructorConstructor;
import lombok.Data;
import lombok.NoArgsConstructorConstructor;

@Data
@NoArgsConstructorConstructor
@AllArgsConstructorConstructor
//Output
public class MarksInfo {
private String subName;
private Double thoeryMarks;
private Double labMarks;
private Double theoryMarksPer;
private Double labMarksPer;
private String examInfo;
}
Result class:
package in.nit.model;

import java.util.List;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.Data;
@Data
@XmlRootElement
//Output
public class Result {
private Integer sid;
private String sName;
private Double totalMarks;
private Double avg;
private String grade;
```

```
private List<MarksInfo> marksInfoList;
private String finalResult;
}
```

In ExamResultRestController:

```
package in.nit.controller;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import in.nit.model.Mark;
import in.nit.model.MarksInfo;
import in.nit.model.Result;
import in.nit.model.Student;
@Path("/exam")
public class ExamResultRestController {

 @POST
 @Consumes({"application/xml,application/json"})
 @Produces({"application/xml,application/json"})

 public Result findExamResultInfo(
 Student student
) {

 Double totalMarks=0.0;
 Double labTotalMarks=0.0;
 String examInfo="";
 Double allMarks=0.0;
 Double avg=0.0;
 List<MarksInfo> marks=null;
 String grade=null;
 MarksInfo marksInfoList=null;
```

```
//total Theory Marks
List<Mark> marksInfo=student.getMarkList();
//looping
Iterator<Mark> itr=marksInfo.iterator();
while(itr.hasNext()) {
 Mark add=itr.next();
 totalMarks+=add.getTheoryMarks();
}

//total Lab Marks.....looping
Iterator<Mark> itr1=marksInfo.iterator();
while(itr1.hasNext()) {
 Mark add=itr1.next();
 labTotalMarks+=add.getLabmarks();
}

//Calculate Percentage
double
marksPercentage=totalMarks/marksInfo.size()/100;
double
labMarksPercentage=labTotalMarks/marksInfo.size()/100;
marks=new ArrayList<>();

//checking pass or failed
for(Mark m:marksInfo) {
 if(m.getTheoryMarks()>40 &&
m.getLabmarks()>60) {
 examInfo="PASS";

 if(avg>=75.0) {
 grade="A+";
 }
 else if(avg>=60.0 && avg<=75.0) {
 grade="A";
 }
 else if(avg>=45.0 && avg<=60.0) {
 grade="B+";
 }
 else if(avg>40.0 && avg<=45.0) {
```

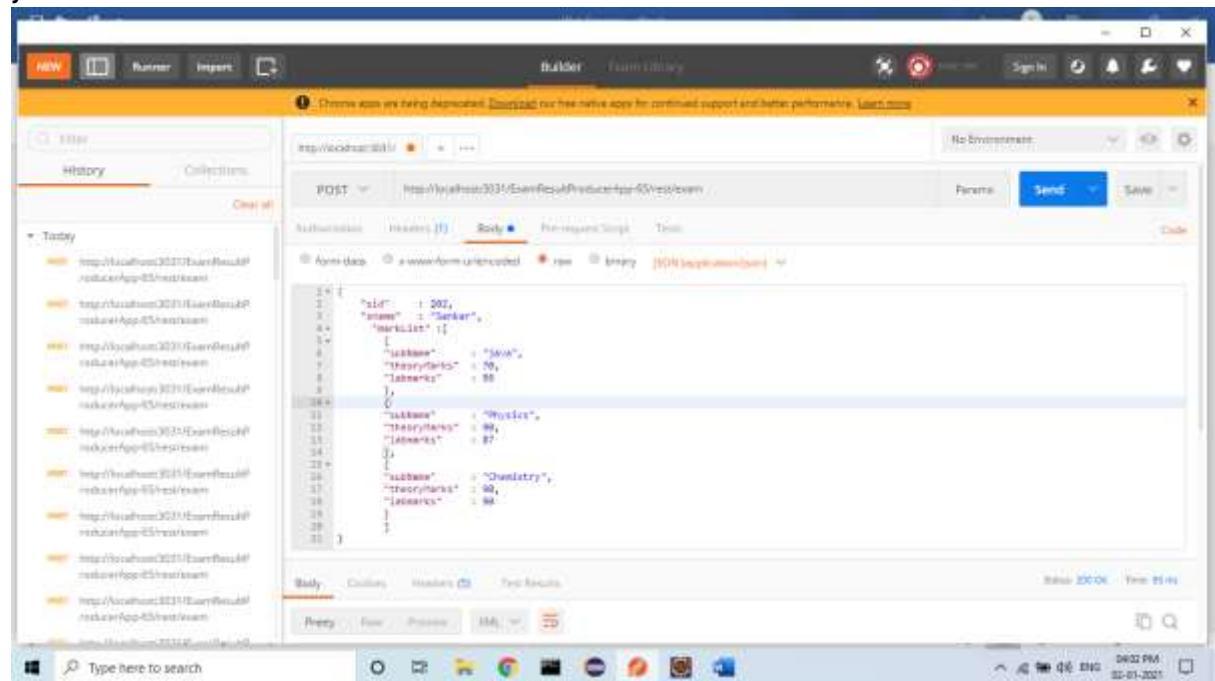
```
 grade="B";
 }
 //Convert to MarksInfo Object
 marksInfoList=new
MarksInfo(m.getSubName(),totalMarks,labTotalMarks,marksPer
centage,labMarksPercentage, examInfo);
}
else {
 examInfo="FAILED";
 //Convert to MarksInfo Object
 marksInfoList=new
MarksInfo(m.getSubName(),totalMarks,labTotalMarks,marksPer
centage,labMarksPercentage, examInfo);
 grade="---NA---";
 marks.clear();
 break;
}
//calculating
allMarks=totalMarks+labTotalMarks;
avg=allMarks/m.getSubName().length();
marks.add(marksInfoList);
}
//Result set to Object
Result re=new Result();
re.setSid(student.getSid());
re.setSName(student.getSname());
re.setTotalMarks(totalMarks);
re.setAvg(avg);
re.setMarksInfoList(marks);
re.setGrade("Grade is: "+grade);
re.setFinalResult("Final Result is: "+examInfo);
return re;

}
}
```

-----Output-----

Input:

```
{
 "sid" : 202,
 "sname" : "Sankar",
 "markList": [
 {
 "subName" : "java",
 "theoryMarks" : 70,
 "labmarks" : 96
 },
 {
 "subName" : "Physics",
 "theoryMarks" : 90,
 "labmarks" : 87
 },
 {
 "subName" : "Chemistry",
 "theoryMarks" : 90,
 "labmarks" : 90
 }
]
}
```



## Output:

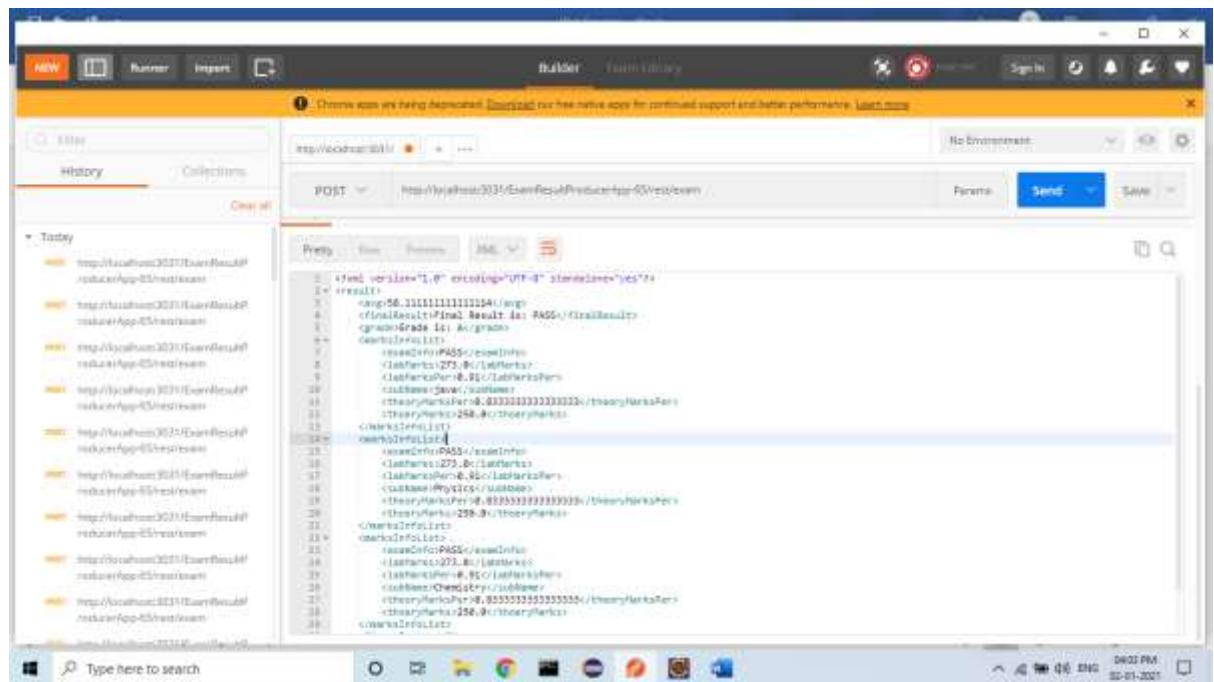
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<result>
 <avg>58.1111111111114</avg>
 <finalResult>Final Result is: PASS</finalResult>
 <grade>Grade is: A</grade>
 <marksInfoList>
 <examInfo>PASS</examInfo>
 <labMarks>273.0</labMarks>
 <labMarksPer>0.91</labMarksPer>
 <subName>java</subName>

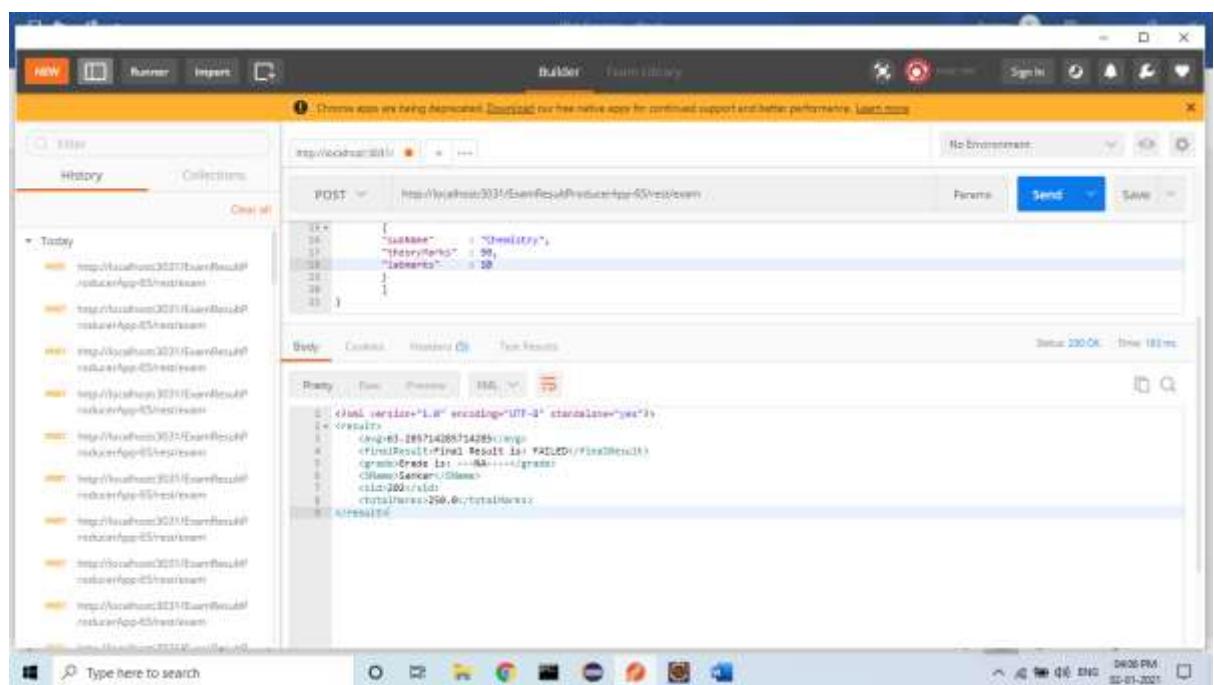
 <theoryMarksPer>0.8333333333333333</theoryMarksPer>
 <thoeryMarks>250.0</thoeryMarks>
 </marksInfoList>
 <marksInfoList>
 <examInfo>PASS</examInfo>
 <labMarks>273.0</labMarks>
 <labMarksPer>0.91</labMarksPer>
 <subName>Physics</subName>

 <theoryMarksPer>0.8333333333333333</theoryMarksPer>
 <thoeryMarks>250.0</thoeryMarks>
 </marksInfoList>
 <marksInfoList>
 <examInfo>PASS</examInfo>
 <labMarks>273.0</labMarks>
 <labMarksPer>0.91</labMarksPer>
 <subName>Chemistry</subName>

 <theoryMarksPer>0.8333333333333333</theoryMarksPer>
 <thoeryMarks>250.0</thoeryMarks>
 </marksInfoList>
 <SName>Sankar</SName>
 <sid>202</sid>
 <totalMarks>250.0</totalMarks>
</result>
```



If any subject Mark or lab Mark Failed:



Convert JSON to XML:  
<https://www.convertjson.com/json-to-xml.htm>

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

The screenshot shows a Windows application window titled "JSON To XML Converter". The main interface has three tabs: "JSON Data", "Convert JSON", and "XML Output". The "JSON Data" tab contains a JSON code editor with the following content:

```
[{"id": 202, "name": "Sankar", "marks": [{"subject": "java", "theoretical": 70, "internals": 95}, {"subject": "Physics", "theoretical": 90, "internals": 87}, {"subject": "Chemistry", "theoretical": 85, "internals": 90}]}
```

The "Convert JSON" tab shows the generated XML output:

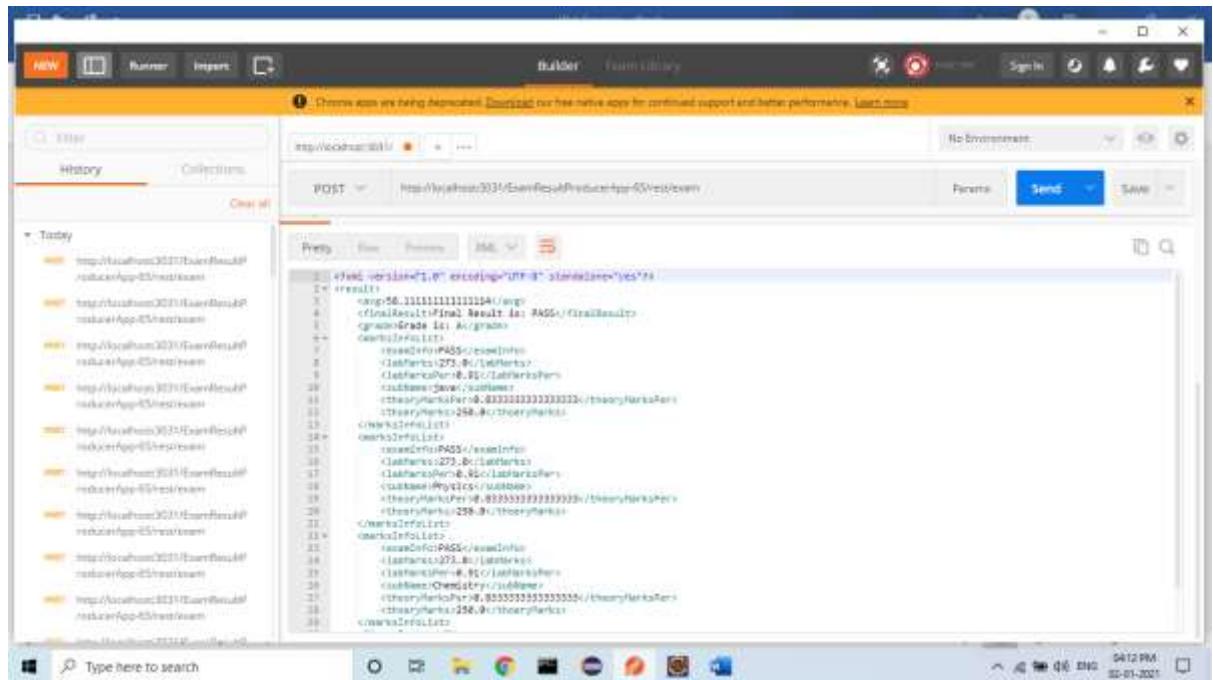
```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
<id>202</id>
<name>Sankar</name>
<marksList>
<subject>java</subject>
<theoretical>70</theoretical>
<internals>95</internals>
<marksList>
<subject>Physics</subject>
<theoretical>90</theoretical>
<internals>87</internals>
<marksList>
<subject>Chemistry</subject>
```

At the bottom right, there are buttons for "Save your result" and "Download Result".

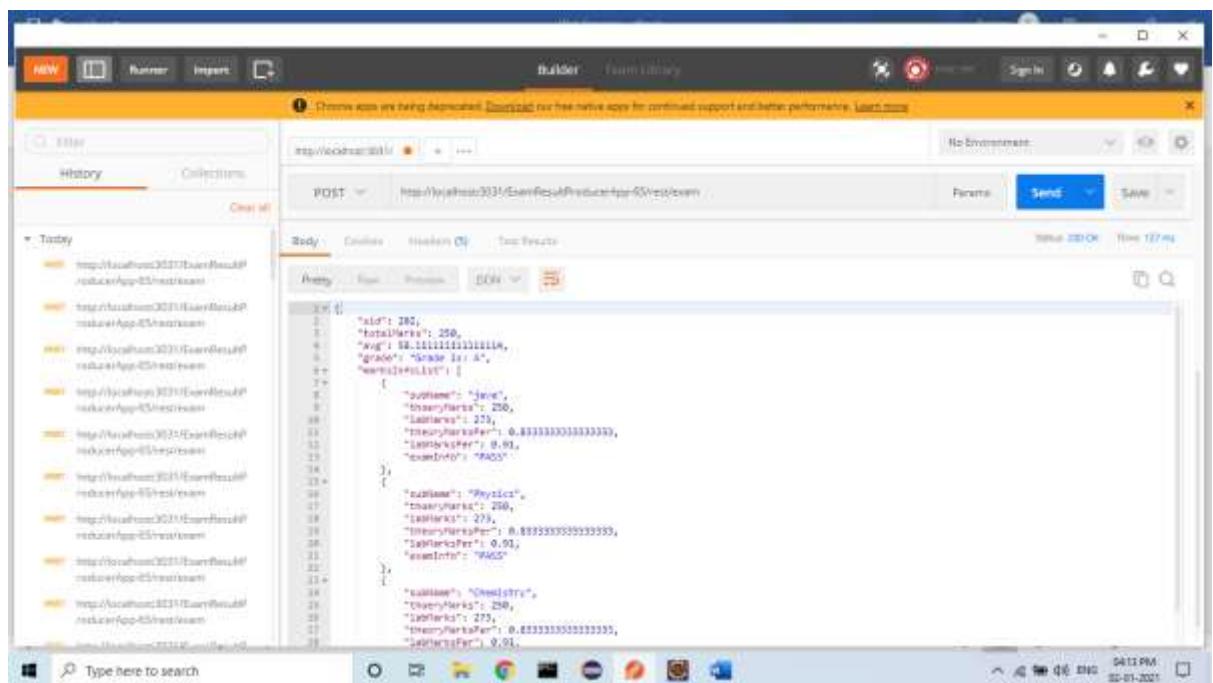
The screenshot shows the Postman application interface. The left sidebar lists recent collections and items. The main area shows a POST request to "http://localhost:3031/ExamResult/PostExam". The "Body" tab is selected, showing the JSON payload:

```
{
 "student": {
 "id": 202,
 "name": "Sankar",
 "marks": [
 {
 "subject": "java",
 "theoretical": 70,
 "internals": 95
 },
 {
 "subject": "Physics",
 "theoretical": 90,
 "internals": 87
 },
 {
 "subject": "Chemistry",
 "theoretical": 85,
 "internals": 90
 }
]
 }
}
```

Below the body, the "Headers" tab is visible. At the bottom, the status bar shows "Status: 200 OK Time: 103 ms".



Header-> accept->application/json



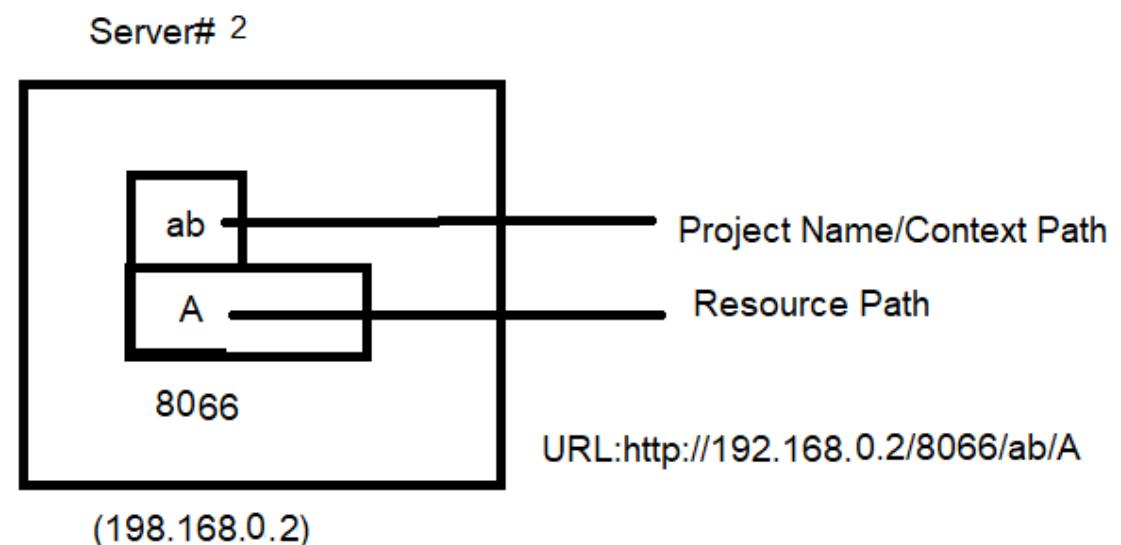
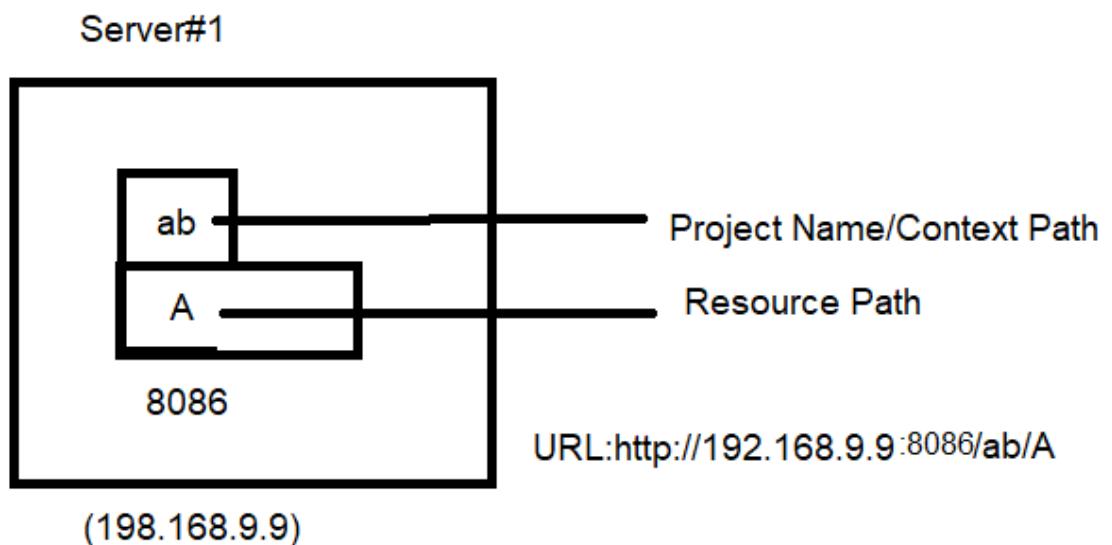
# URL&URI

- URL : Uniform Resource Locator
- URI : Uniform Resource Indicator
- Resource : Any file/code

- Uniform : Unique/ no duplicate type
- Ex URL: <http://198.2.3.5:8795/App/one>
- http : Protocol
- 198.2.3.5 : IP Address
- 8795 : Port
- App/one : URL
- App : ContextPath
- one : ResourcePath

Note : URL may change from Server to Server, but URI cannot change from Server to Server.

Example:



In the above two diagrams URLs are same i.e ab/A, even URLs different.

---

## JERSEY 2.X

- Setup of Jersey 2.x in Eclipse:
- Jersey 2.x has provided “jersey-quickstart” archetype which supports auto-configuration for:
  1. Application creation(web).
  2. Pom.xml configuration
  3. Web.xml
  4. One default producer class or Resource class.
- To add this in Eclipse, we need to follow Steps.  
File-> new -> Maven Project->do not choose any check box -> next-> click on button “Add Archetype” option.
- In pop-up enter details, like

Archetype GroupId : org.glassfish.jersey.archetypes  
Archetype ArtifactId : Jersey-quickstart-webapp  
Archetype Version : 3.0.0

- Click on Add button -> Finish.

## Creating Project Using Jersey2.x:

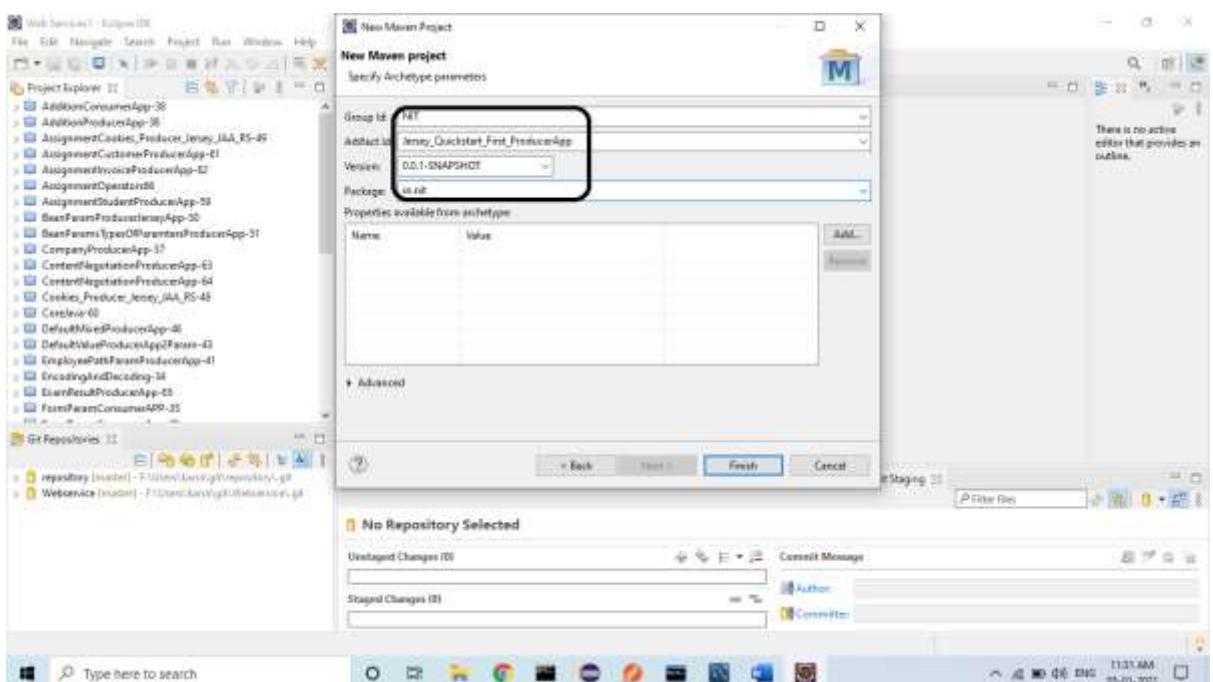
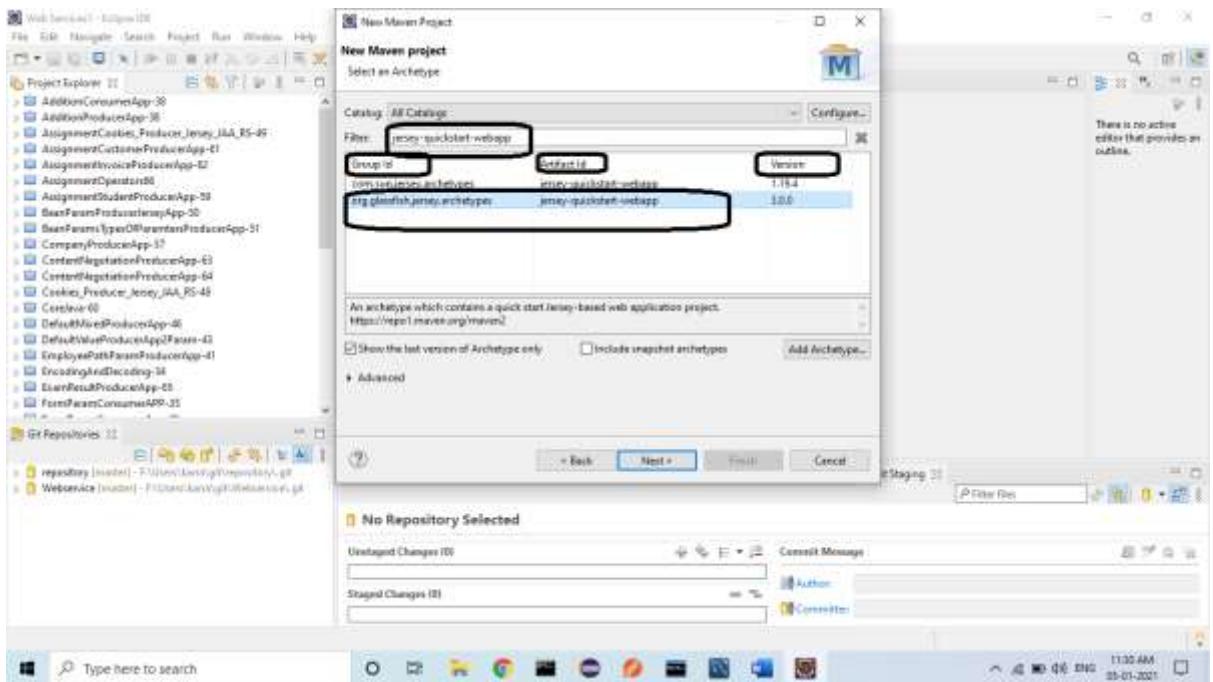
1. Create Jersey Project
- File->new->Maven Project
  - Next-> Search “jersey-quickstart-webapp” ->Choose jersey-quickstart-webapp
  - Next-> enter details like:

Note: choose version 2.x and tomcat9 .

Or

Note: If you choose version 3.0.0 then choose Tomcat10

Because of Jersey3.x package names changed servlet to jakarta



→ Finish.

2. Delete index.jsp from App

→ Go to “src/main/webapp”.

→ Click on index.jsp ->Delete

3. Update JDK details in pom.xml File or Project facets.

4. Configure Server to Project.

5. Delete default Resource class(producer class) given under package “in.nit”.
6. Update Maven Project. If you are changed Java Version Project Facets don’t do this step.

-----Jersey\_2.x\_Archetype\_Producer\_App-----

In pom.xml File: note: change like after created project in pom.xml

```
<build>
 <finalName>Jersey_Quickstart_First_ProducerApp</finalName>
 <plugins>
 <plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>2.5.1</version>
 <inherited>true</inherited>
 <configuration>
 <source>15</source>
 <target>15</target>
 </configuration>
 </plugin>
 </plugins>
</build>
<dependencies>
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet-core</artifactId>
 <!-- use the following artifactId if you don't need servlet 2.x
compatibility -->
 <!-- artifactId>jersey-container-servletjson-binding</artifactId>
```

```
</dependency>
-->
</dependencies>
<properties>
 <jersey.version>3.0.0</jersey.version>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
</properties>
```

In web.xml File:

```
<web-app>
<servlet>
 <servlet-name>Jersey Web Application</servlet-name>
 <servlet-
class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-
name>
 <param-value>in.nit</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
 <servlet-name>Jersey Web Application</servlet-name>
 <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
</web-app>
```

In RestController:

**package** in.nit;

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("p")
public class MyResource {
 @GET
 @Produces(MediaType.TEXT_PLAIN)
```

```
public String getIt() {
 return "Got it!";
}
}
```

-----output-----

In chrome:

[http://localhost:3031/Jersey\\_Quickstart\\_First\\_ProducerApp-68/rest/p](http://localhost:3031/Jersey_Quickstart_First_ProducerApp-68/rest/p)

Got it!

---

Note:

1. FrontController in Jersey2.x is designed by Glassfish Team, which is extension to old FrontController, in pack “org.glassfish.jersey.servlet”.
2. Here we must provide one init-param that reads package details of all our producer classes.
3. So that Jersey will create object to those classes only .
4. Code given as:

```
<init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit</param-value>
</init-param>
```

It means only classes written under “in.nit” package are considered as Producer classes others are Ignored.

**<Init-param> tag:**

Input Provided to init() method in Servlet.

**<Servlet> tag:**

It provides class and object name to container so that container can create object.

#### < servlet-mapping > tag:

It maps our java Object with URL, so client, like browser, can access this one.

- \*\*\*\* destroy() method will be called before destroying the object by container, which will do closing and shutdown works like close connection, close file, remove links. Etc.
- 

---TypesOfParametersInRestTestProjectProducer-70-----

---In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.32</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
```

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
</dependencies>
```

In web.xml File:

```
<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

In model class:

```
package in.nit.model;
```

```
import javax.ws.rs.QueryParam;

import lombok.Data;

@Data
public class Employee {
 @QueryParam("id")
 private int empld;
 @QueryParam("name")
 private String empName;
}
```

In Controller class:

```
/*
 * package in.nit.controller;
 *
 * import javax.ws.rs.GET; import javax.ws.rs.Path; import
 * javax.ws.rs.QueryParam; import javax.ws.rs.core.Response;
 *
 * @Path("/resp") public class ProducerQeuryParameter {
 *
 * @Path("/r")
 *
 * @GET public Response doOperation(
 *
 * @QueryParam("id")Integer id,
 *
 * @QueryParam("name")Integer name) { Object output=id+" "+name;
 *
 * return Response.status(200).entity(output).build(); } }
```

```
*/
//-----

/*
 * package in.nit.controller; import javax.ws.rs.GET; import
 * javax.ws.rs.MatrixParam; import javax.ws.rs.Path;
 *
 * @Path("/resp") public class ProducerQeuryParameter {
 *
 * @Path("/r")
 *
 * @GET public String doOperation(
 *
 * @MatrixParam("id")Integer id,
 *
 * @MatrixParam("name")Integer name) {
 *
 * return "id is :" +id+ " name is : "+name; } }
 */

//-----

/*
 * package in.nit.controller; import javax.ws.rs.GET; import
 * javax.ws.rs.HeaderParam; import javax.ws.rs.Path; import
```

```
* javax.ws.rs.core.Response; import javax.ws.rs.core.Response.Status;
*
* @Path("/resp") public class ProducerHeaderParamController {
*
* @Path("/r")
*
* @GET public Response validateUser(
*
* @HeaderParam("id")String id,
*
* @HeaderParam("name")String name) { Object output=id+" "+name;
*
* return Response.status(200).entity(output).build(); } }
*/
//-----

/*
* package in.nit.controller; import javax.ws.rs.FormParam; import
* javax.ws.rs.POST; import javax.ws.rs.Path;
*
* @Path("/resp") public class ProducerHeaderParamController {
*
* @Path("/r")
*
* @POST public String validateUser(
*
* @FormParam("id")String id,
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
/*
 * @FormParam("name")String name) { return "id : "+id+ " name:
 "+name; } }

*/
//-----
```

---

```
/*
 * package in.nit.controller; import javax.ws.rs.FormParam; import
 * javax.ws.rs.POST; import javax.ws.rs.Path;
 *
 * @Path("/resp") public class ProducerHeaderParamController {
 *
 * @Path("/r")
 *
 * @POST public String validateUser(
 *
 * @FormParam("id")String id,
 *
 * @FormParam("name")String name) { return "id : "+id+ " name:
 "+name; } }

*/
//-----
```

---

```
/*
 * package in.nit.controller; import javax.ws.rs.DefaultValue; import
 * javax.ws.rs.GET; import javax.ws.rs.Path; import
 * javax.ws.rsPathParam;
```

```
* @Path("/resp") public class ProducerHeaderParamController {
*
* @Path("/r")
*
* @GET public String validateUser(
*
* @DefaultValue("12")
*
* @PathParam("id")int id,
*
* @DefaultValue("Sankar")
*
* @PathParam("name")String name) { return "id : "+id+ " name:
"+name; } }
*/
package in.nit.controller;

import javax.ws.rs.BeanParam;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

import in.nit.model.Employee;

@Path("/resp")
public class BeanParamProcuderController {
//.../rest/emp/data?eid=202&ename=Sankar&esal=23.44

 @GET
```

```

@Path("/r")
public String readInputs(
 @BeanParam Employee employee
) {
 return "Data is => "+employee;
}

```

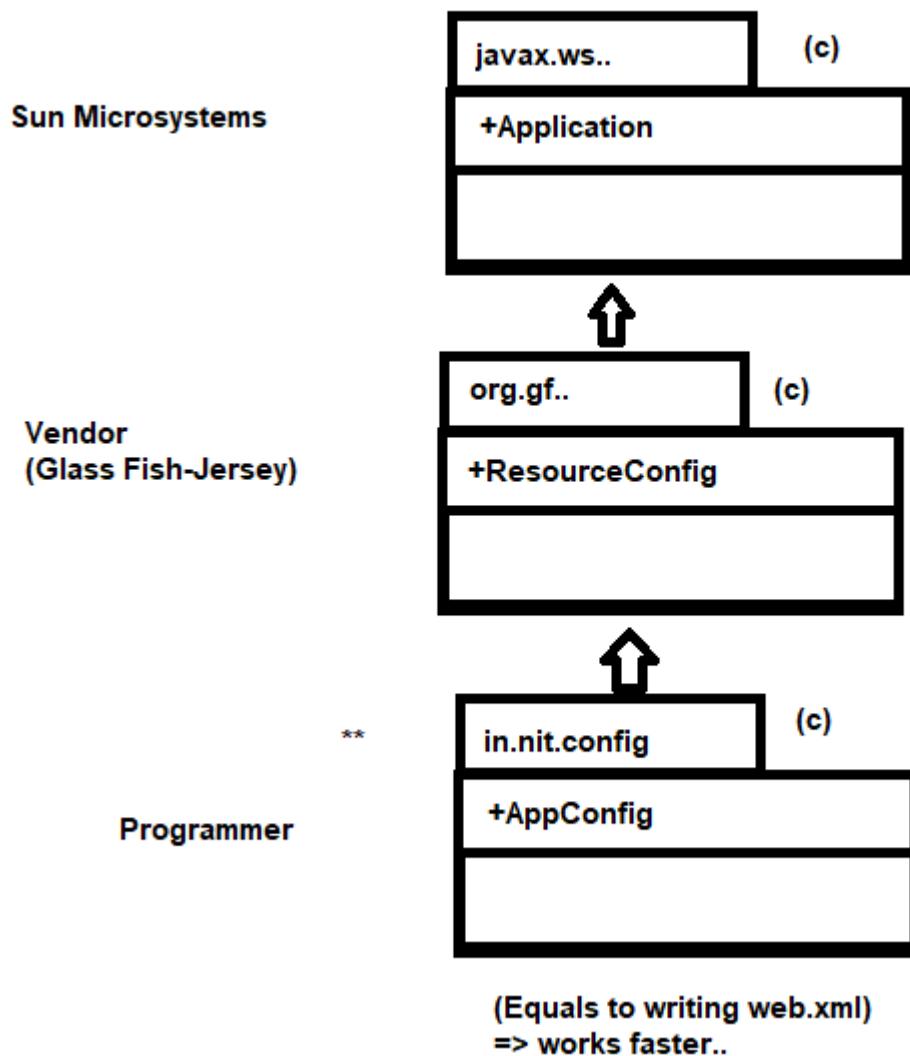
-----Outputs I am not printing-----

-----Comparison table-----

| ReSTful WebServices Parameters(Inputs given by Consumer): Note Jersey2.x Implementation                                                                                                                                       |                                 |                                                                                                                          |                              |                                                                                                                                                   |                                              |                                                                                                                                                                                                                 |             |                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|---------------------------------------------------|
|                                                                                                                                                                                                                               | Key Is Not Present              | Duplicate keys                                                                                                           | Data Type Mismatch           | Not Found(key values)                                                                                                                             | Order                                        | URL and Symbols                                                                                                                                                                                                 | Performance | Used For                                          |
| 1. <code>@Query Parameters</code>                                                                                                                                                                                             | getting default values          | first communion values                                                                                                   | 404 Not Found                | Ignored and getting values                                                                                                                        | any order                                    | (URL?key&key=value) ? &                                                                                                                                                                                         | slow        | Using input and output URL                        |
| 2. <code>@Matrix Parameters</code>                                                                                                                                                                                            | getting certain values          | first communion values                                                                                                   | 404 Not Found                | Ignored and getting default values                                                                                                                | any order                                    | (URL,key=val&key=val)                                                                                                                                                                                           | fast        | Retrieval of data                                 |
| 3. <code>@Header Parameters</code>                                                                                                                                                                                            | Getting default values          | getting result                                                                                                           | getting result               | Ignored until getting default values                                                                                                              | any order                                    | (HTTP Header Section) key=value                                                                                                                                                                                 | --          | Security Headers in position                      |
| 4. <code>@Form Parameters</code>                                                                                                                                                                                              | Getting default values          | first communion values                                                                                                   | getting result               | Ignored if key not defined                                                                                                                        | any order                                    | (HTML Form Object) action=...                                                                                                                                                                                   | --          | To Send Http Request Form Data / Response Headers |
| 5. <code>@Path Parameters/Path Variable (Template)</code>                                                                                                                                                                     | static: default dynamic: 404    | static: default dynamic: default                                                                                         | static: default: dynamic:404 | Ignored and getting default values                                                                                                                | first                                        | (URL/val/val/val) &/A                                                                                                                                                                                           | fast        | without key sending data                          |
| 6. <code>@Cookie Parameters</code>                                                                                                                                                                                            | --                              | --                                                                                                                       | --                           | --                                                                                                                                                | <i>/ here pass path name</i>                 |                                                                                                                                                                                                                 | --          | Value is automatically set based on browser       |
| 7. <code>@Bean Parameters</code>                                                                                                                                                                                              | Getting null Value              | first communion values                                                                                                   | 404-not Found:               | Ignored And getting default values                                                                                                                | any order                                    | Above All based on parameter                                                                                                                                                                                    | --          | reading different patterns                        |
| 8. <code>@DefaultValue</code>                                                                                                                                                                                                 | not work in Path Dynamic values | not work in Path Dynamic values                                                                                          | Dynamic, Path=404            | Not                                                                                                                                               | used for default values<br>10.Senkar,A,23,33 |                                                                                                                                                                                                                 | --          | Setting default values                            |
| - means not applicable<br>Note: Testing done through browser and Postman Tool                                                                                                                                                 |                                 |                                                                                                                          |                              |                                                                                                                                                   |                                              |                                                                                                                                                                                                                 |             |                                                   |
| Dependencies:<br><a href="http://jersey.java.net/">jersey-container-servlet</a><br><a href="http://jersey.java.net/jersey-1.12">jersey-1.12</a>                                                                               |                                 |  Browser not support Use PostMan Tool |                              |  Ignoring: Sending in two ways<br>1. using GET<br>2. Form Data |                                              |  Preferences are Tending in two ways<br>1. Matrix Path --> Higher Priority<br>2. Ignored Path (if exists) --> lower priority |             |                                                   |
|  <b>GET--&gt;4kb</b><br> <b>POST--&gt;Unlimited</b> |                                 |                                                                                                                          |                              |                                                                                                                                                   |                                              |                                                                                                                                                                                                                 |             |                                                   |

# Weservices-part#2

## (Advanced Programming)



- No web.xml is used.
- It is faster compared to normal Configuration.
- It Supports Filters, Layer Coding, Scopes.
- It supports Bean validation (Data Invalidations).
- It supports Tools Config: Log4j, Junit Config.
- It supports Security Basic, Token based Auth(JWT) in webservice,  
In Springboot and microservices CSRF token used Auth.  
Etc.....

-----Setup for Coding-----

#1. Maven web project creation

- File-> new->maven project->next->Search with web-app

→ Choose maven-archetype-webapp->next -> Enter Details

GroupId : in.nit  
ArtifactId : Jersey2AdvancedFirstApp  
Version : 1.0

→ Finish

#2 Update Build path

- Right click on project -> Build path-> Configure build path
- Library option-> Choose JRE->Edit->Update to workspace JRE/JDK.
- Apply
- Click on Add Library -> Server runtime -> Apache tomcat
- Apply and Close.

#3. Expand Project and delete index.jsp,web.xml file.

#4. Add below details in pom.xml File.

-----Jersey\_No\_Web\_xml\_AdvancedFirstAopp-71-----

In pom.xml File:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
```

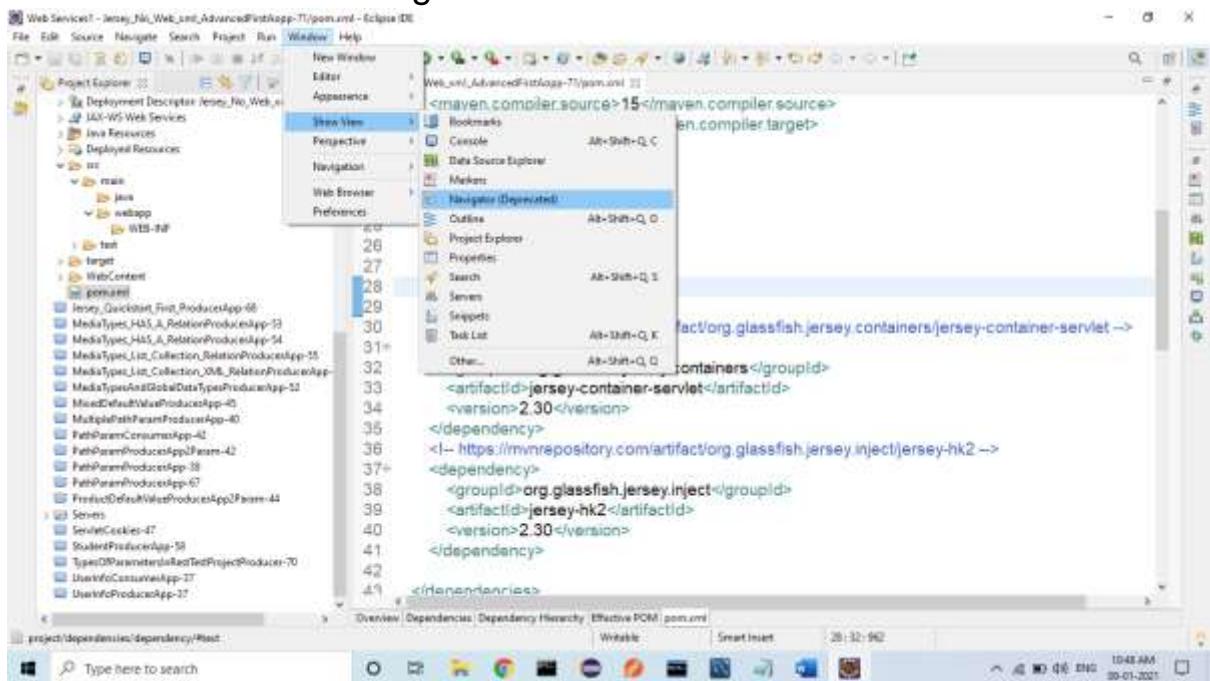
```
<groupId>org.glassfish.jersey.containers</groupId>
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.30</version>
</dependency>

</dependencies>
```

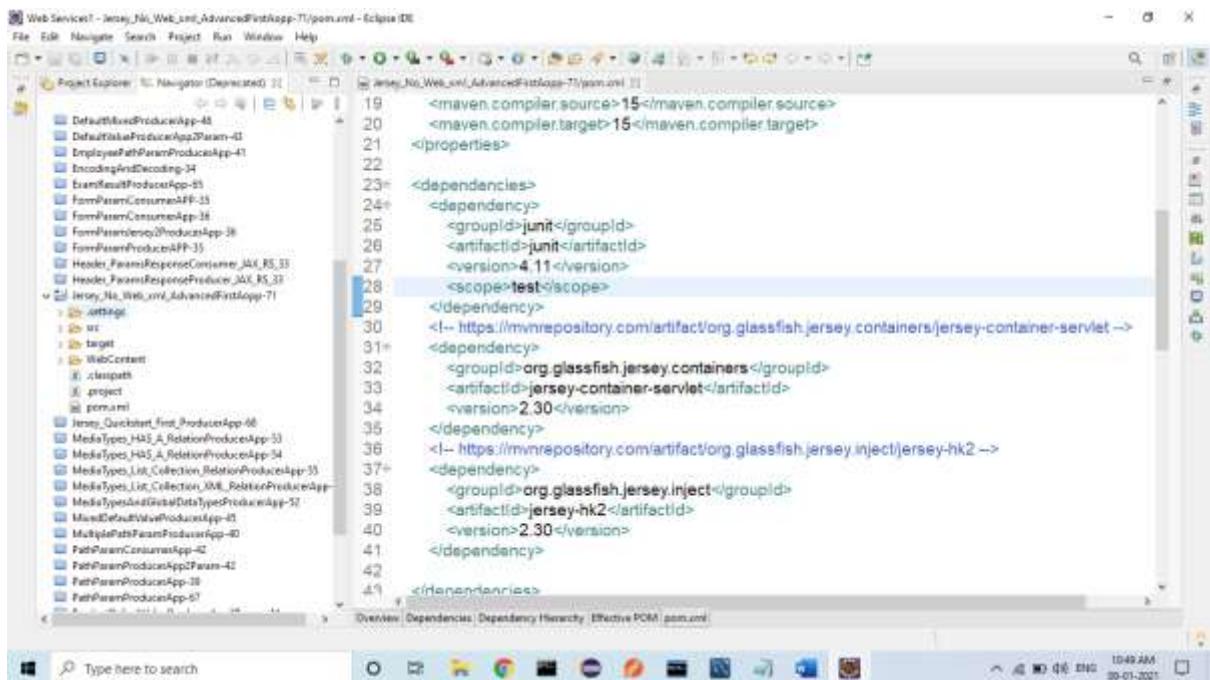
- \*\* Here `<failOnMissingWebXml>false</failOnMissingWebXml>` indicates App works without web.xml (Java based Config Process)
- #5. Update Maven Project(Alt+F5)->Force Update->Finish

## #6.\*\*\* Facets Updates

- Windows>show view >Navigator



- Expand our Project  
→ Click on .setting folder



```

<maven.compiler.source>15</maven.compiler.source>
<maven.compiler.target>15</maven.compiler.target>
<properties>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->
<dependency>
<groupId>org.glassfish.jersey.containers</groupId>
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.30</version>
</dependency>
</dependencies>

```

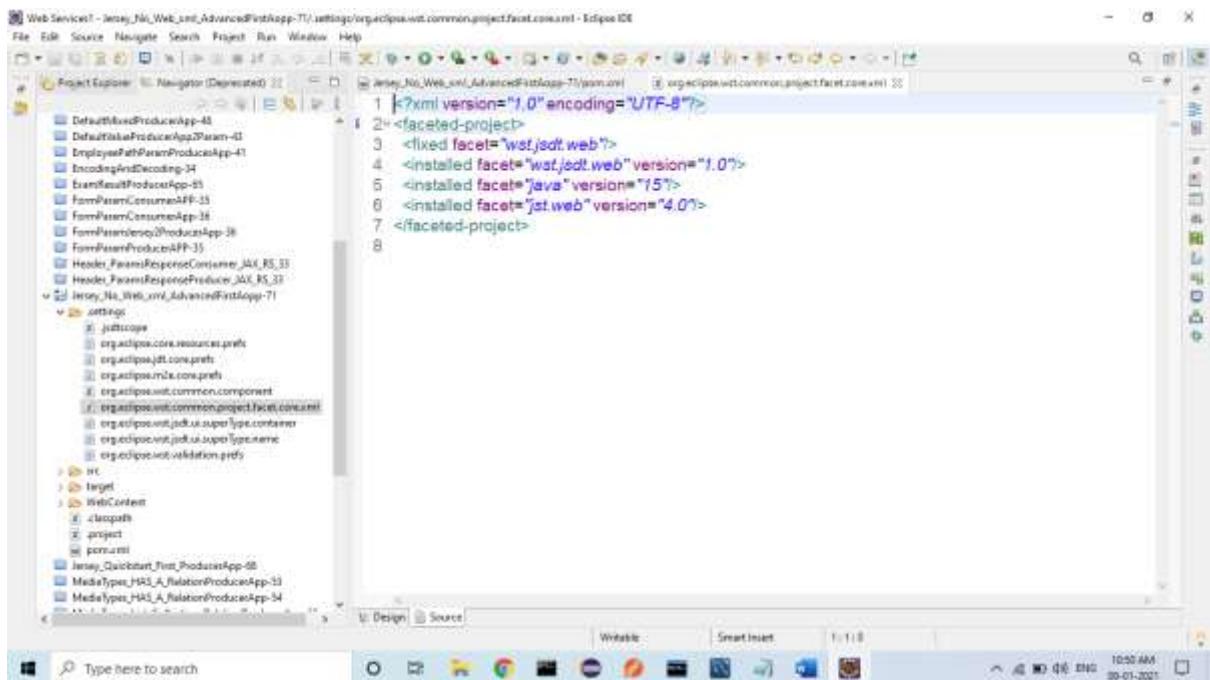
→ Open file 'org.eclipse.wst.common.project.facet.core.xml'

→ Modify details

jst.web =4.0

java =15

Note: If not above versions then change else keep it



```

<?xml version="1.0" encoding="UTF-8"?>
<faceted-project>
<fixed facet="wst,jdt.web">
<installed facet="wst,jdt.web" version="1.0"/>
<installed facet="java" version="15"/>
<installed facet="jst.web" version="4.0"/>
</faceted-project>

```

#7 Back to Project Explorer and  
 Update Maven Project (Alt+F5) -> Force Update -> Finish

Note: I am Using Dynamic Module4.0

And java 15 so no need above process just add tag under properties no suggestions are coming

```
<failOnMissingWebXml>false</failOnMissingWebXml>
```

#8. Define one Class Ex: AppConfig that behaves like FC(web.xml code)  
(src/main/java)

In AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.server.ResourceConfig;
@Path("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC:
ServletContainer
//Constructor
public AppConfig() {
 //packages("in.nit");//init-param
 //or
 this.packages("in.nit");//init-param
}
}
```

---

→ Note:

1. method packages() is defined in ResourceConfig.  
Supper class method we can access in sub class  
With or without using “this” keyword.
2. Here, class: ResourceConfig is given by Jersey vendor that indicates FrontController(FC) class.
3. @ApplicationPath("") is used to provide URL pattern for our App.

#9 Define One RestController:

```
package in.nit.controller;
```

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/msg")
public class MessageRestController {
 @GET
 public String show() {

 return "Hello From Advanced Config!";
 }
}
```

#10 Run application and Enter URL:

[http://localhost:3031/Jersey\\_No\\_Web\\_xml\\_AdvancedFirstApp-71/rest/msg](http://localhost:3031/Jersey_No_Web_xml_AdvancedFirstApp-71/rest/msg)

---

Hello From Advanced Config!

---

Note:

If you are using this setup or higher then not require

<failOnMissingWebXml>false</failOnMissingWebXml>

Because by default false.

```
<plugin>
 <artifactId>maven-war-plugin</artifactId>
 <version>3.2.2</version>
</plugin>
```



Tomcat9, Servlet 4.0

Apache Tomcat version 9.0 implements the Servlet 4.0 and JavaServer Pages 2.3

**web.xml not Require if you are in same Setup:**

**maven 3.1.0 or later**

**Tomcat9 or later**

**Servlet3.0 or later      Dynamic Module4.0**

-----above Example with Modified code-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
```

```
</dependency>
<!--

<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

---

In AppConfig

```
package in.nit.config;
```

```
import javax.ws.rs.ApplicationPath;
```

```
import org.glassfish.jersey.server.ResourceConfig;
@ApplicationPath("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC: ServletContainer
 //Constructor
 public AppConfig() {
 packages("in.nit");//init-param
 }
}
```

In RestController:

```
package in.nit.controller;
```

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/msg")
public class MessageRestController {
 @GET
 public String show() {

 return "Hello From Advanced Config!";
 }
}
```

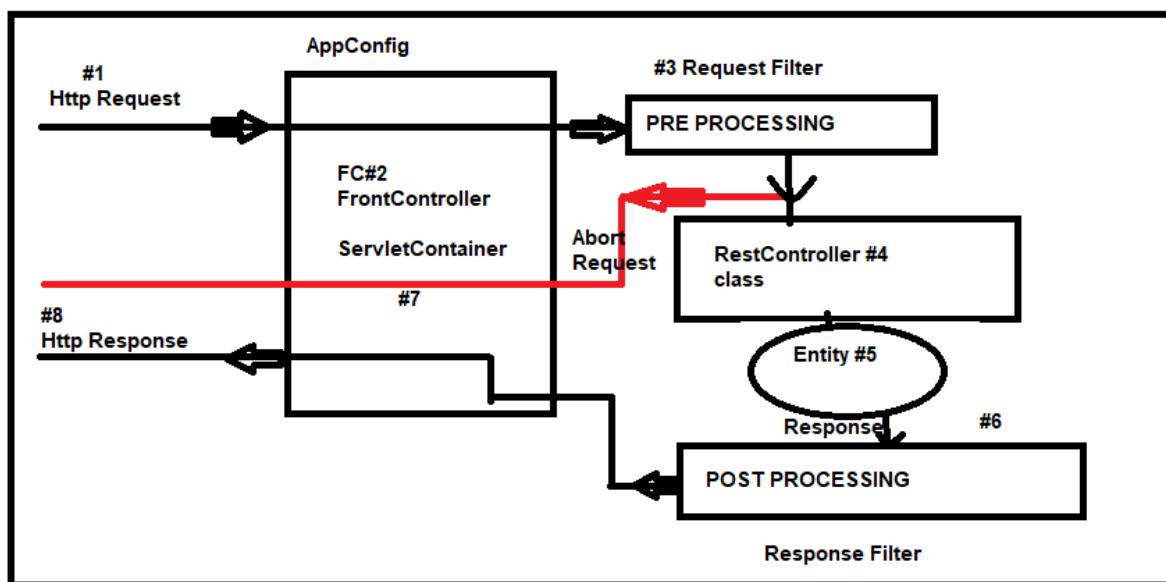
---

[http://localhost:3031/Jersey\\_No\\_Web\\_xml\\_AdvancedFirstApp-72/rest/msg](http://localhost:3031/Jersey_No_Web_xml_AdvancedFirstApp-72/rest/msg)

---

Hello From Advanced Config!

---



→ Filters: These are components added to Application which are used to execute

PRE PROCESSING over HTTP Request and  
POST PROCESSING over HTTP Response.

→ Here, PROCESSING means executing RestController.

→ In simple Filters are used to validate Request data Mainly like HeaderParams.

Example: Security details HeaderParams,  
MediaType HeaderParam, Data Length Header,  
Cookies exist or not in Head section.....etc.

→ To implement Filters concept at Producer Application JAX-RS 2.x (Rest Webservices 2.x) has provided two interfaces:

\*\*\*Note: In execution of Filter, while validation request, If it is Invalid(Has No valid inputs), then we can avoid executing of such request, i.e called as “Request Aborting with one Error Response.”

a) ContainerRequestFilter (I)

```
filter(ContainerRequestcontext req):void
throws IOException;
```

b) ContainerRequestFilter(I)

```
filter(ContainerRequestContext
req,ContainerResponseContext resp):void throws IOException;
```

→ Programmer has to define one sub class for above Filter interfaces and override method filter() there we can define PRE/POST processing logic.

→ It must be registered inside AppConfig, else these are not executed.

Example: register(MyRequestFilter.class);

→ If we do not register then Filters are not considered in execution order.

→ It will inform to FC, so that Container creates object and FC call filter() method on Req/Resp.

-----Coding Steps-----

1. Maven web Project Creation

Example: Jersey\_No\_Web\_xml\_AdvancedSecondApp-73

2. Update Build path

3. Expand Project and Delete index.jsp,web.xml Files

4. Add below details in pom.xml.

<properties>

```
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>

</dependencies>
```

## 5. Filters:

Request:

```
package in.nit.filters;

import java.io.IOException;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
```

```
public class MyRequestFilter implements ContainerRequestFilter{

 @Override
 public void filter(ContainerRequestContext requestContext)
throws IOException {
 System.out.println("FROM REQUEST FILTER....");
}

}
```

Response:

```
package in.nit.filters;

import java.io.IOException;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;

public class MyResponseFilter implements ContainerResponseFilter{

 @Override
 public void filter(ContainerRequestContext requestContext,
ContainerResponseContext responseContext)
throws IOException {
 System.out.println("FROM RESPONSE FILTER");
}

}
```

AppConfig:

6. Define one Class Ex: AppConfig that behaves like FC(web.xml code)  
(src/main/java)

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.server.ResourceConfig;

import in.nit.filters.MyRequestFilter;
import in.nit.filters.MyResponseFilter;
@ApplicationPath("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC:
ServletContainer
 //Constructor
 public AppConfig() {
 packages("in.nit");//init-param

 //Activate Filters
 register(MyRequestFilter.class);
 register(MyResponseFilter.class);
 }
}
```

#### 7. RestController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/msg")
public class MessageRestController {

 @GET
 public String show() {

 System.out.println(" From Rest Controller");
 return "Hello From Advanced Config!";
 }
}
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
}
```

```
}
```

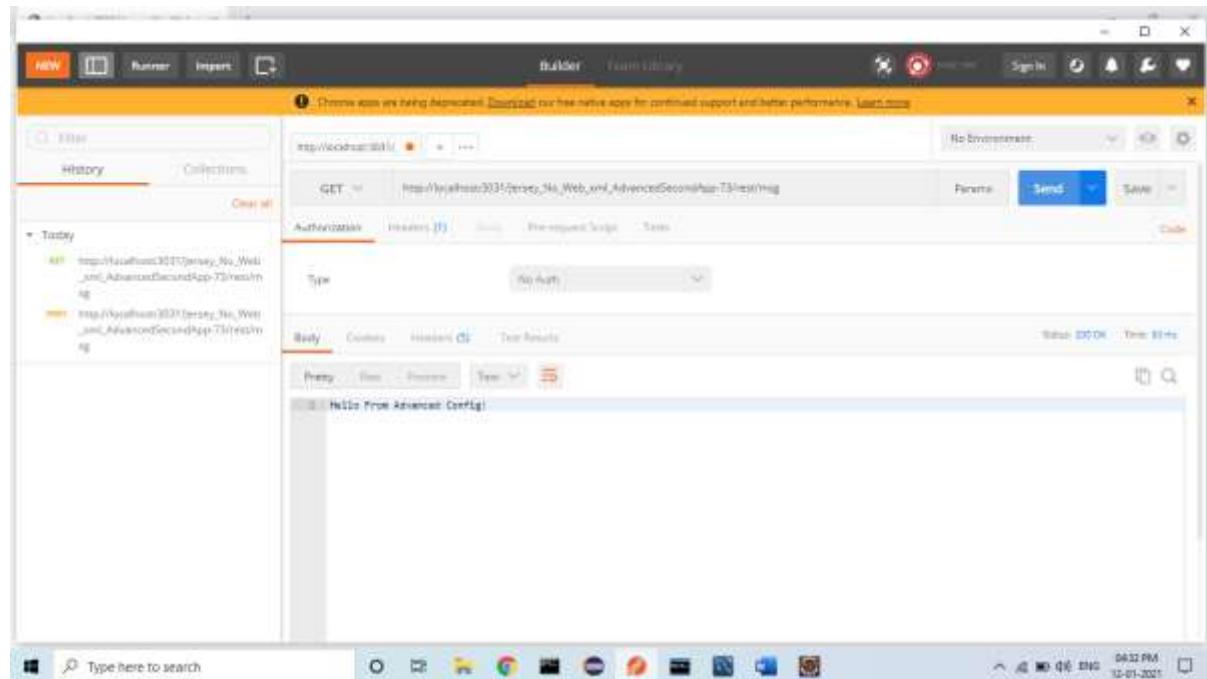
## -----Output-----

Console Output:

```
FROM REQUEST FILTER....
From Rest Controller
FROM RESPONSE FILTER
```

[http://localhost:3031/Jersey\\_No\\_Web\\_xml\\_AdvancedSecondApp-73/rest/msg](http://localhost:3031/Jersey_No_Web_xml_AdvancedSecondApp-73/rest/msg)

Hello From Advanced Config!



## -----Filters In Servlet-----

- Filter: Filters are auto-executable codes for Servlets. These are used to execute “Pre-processing logic on request” and “Post-processing logic on response”.
- To create one filter write one class and implement Filter(I) interface given by javax.servlet package.
- It has 3 abstract methods also called as life cycle methods.
- Those are init(), doFilter(), destroy().
- After writing this filter class, must configure in web.xml file using below code.

```
<filter>
 <filter-name></filter-name>
 <filter-class></filter-class>
</filter>
```

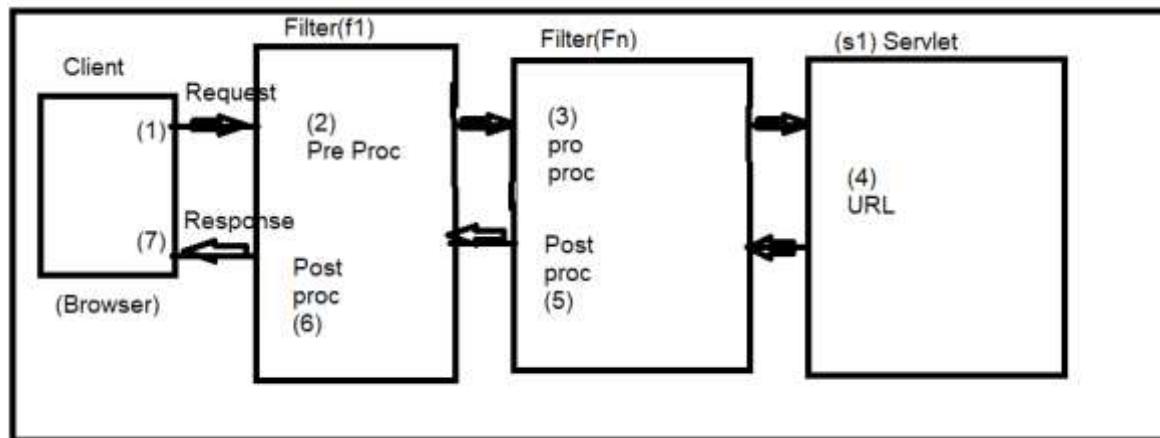
- It is used to create Filter class object by Servlet container(web container).

```
<filter-mapping>
 <filter-name></filter-name>
 <url-pattern></url-pattern>
</filter-mapping>
```
- This code is used to link one filter with Servlet, i.e \*\*\*\*\*Filter URL Pattern must match with Servlet URL Pattern.
- Filter class Objects are created by Servlet container on server startup (default: eager/early loading).
- <load-on-startup> is not Possible.
- For one Servlet we can write multiple Filter, execute in web.xml Configuration Order.
- Filter Accept the init() parameters like Servlet. Input to init() method.
- To unlink Filter with Servlet, just change filter URL Pattern i.e \*\*\*\*\*Servlet URL Pattern!=Filter URL Pattern.
- **It is also called as Filter are Pluggable.** i.e Easy link/unlink.
- Filter can be used only for either “Pre-Processing” or “Post-Processing” or Even Both.
- Browser makes request to Servlet only but filter process it then goes to Servlet.
- **\*\*\*\*\*doFilter() method is available in Filter interface as Life cycle method and in FilterChain to call next Filter or Servlet.**
- Filter are used in Security Programming, Session check, Encrypt and Decrypt, identity verification, request data validation, response conversions etc.

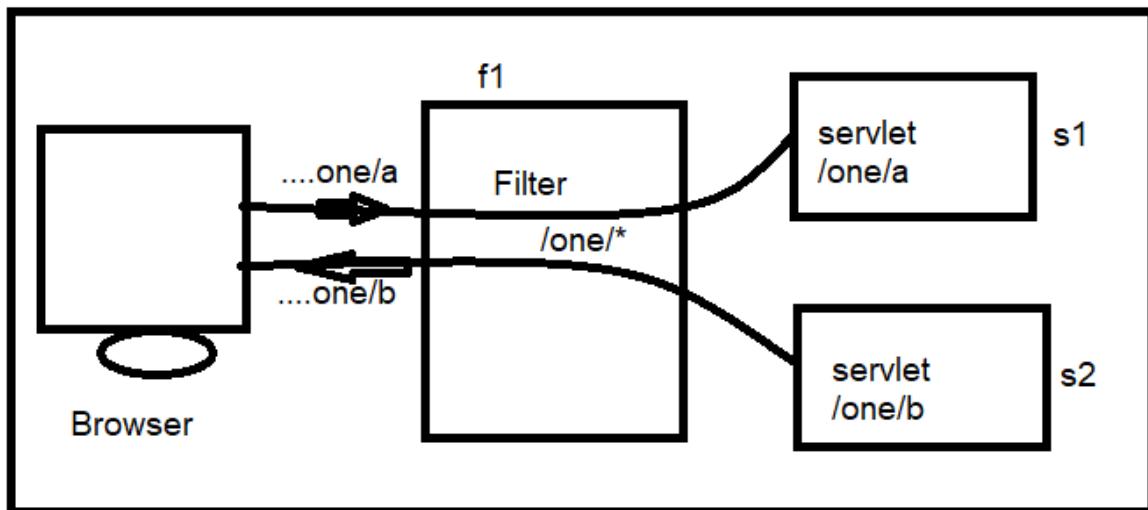
→ One Filter can be linked with multiple Servlets also. But Servlet URL Pattern Some part must be same, then Filter URL Pattern should follow either Extension match or directory Match.

Design1: Multiple filter for one Servlet.

- Request Execution order f1, f2, f3, ..... fn
- Response Execution order fn, fn-1,.....f2, f1.



Design2: One Filter linked with multiple Servlets:



Filter Coding Steps:

Step1: Write on public class and implements Filter(I).

Step2: Override 3 abstract methods init(), doFilter(), destroy().

Step3: Provide logic in three methods in Life Cycle doFilter() method call FilterChain doFilter() method, which calls next Filter in series or Servlet service method( or any doXXX() method).

Filter(I): doFilter(ServletRequest, ServletResponse, FilterChain)

→ It is a Life Cycle method. It is a 3 Parameters.

FilterChain(I): doFilter(ServletRequest,ServletResponse)

→ Calls next Filter or Servlet. It is a 2 Parameters.

Step4: Configured filter in web.xml make sure Filter URL Pattern must match with Servlet URL Pattern.

Format should be:

```
<filter>
 <filter-name></filter-name>
 <filter-class></filter-class>
</filter>
<filter-mapping>
 <filter-name></filter-name>
 <url-pattern></url-pattern>
</filter-mapping>
```

Step5: Run application in Server and make request to Servlet.

-----Example one Related to Design 1-----

In SampleServlet:

```
package in.nit.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SampleServlet extends HttpServlet {
 @Override
 public void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
 System.out.println("SampleServlet.doGet()");
 }
}
```

TestFilterOne:

```
package in.nit.filter;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class TestFilterOne implements Filter {
 public void destroy() {
 System.out.println("Destroy Filter one");
 }
 public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {
 System.out.println("filter one pre-processing");
 }
}
```

```
 chain.doFilter(request, response);
 System.out.println("filter one post-processing");
 }
 public void init(FilterConfig fConfig) throws ServletException {
 System.out.println("Init Filter one");
 }
}
```

In TestFilterTwo:

```
package in.nit.filter;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class TestFilterTwo implements Filter {
 public void destroy() {
 System.out.println("Destroy Filter Two");
 }
 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
 System.out.println("filter Two pre-processing");
 chain.doFilter(request, response);
 System.out.println("filter Two post-processing");
 }
 public void init(FilterConfig fConfig) throws ServletException {
```

```
String data=fConfig.getInitParameter("data");
System.out.println("Init Filter Two: "+data);
}

}
```

In web.xml File:

```
<web-app>
<display-name>ServletFilters-77</display-name>
<servlet>
<servlet-name>ss</servlet-name>
<servlet-class>in.nit.servlet.SampleServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ss</servlet-name>
<url-pattern>/sample</url-pattern>
</servlet-mapping>
<filter>
<filter-name>f1</filter-name>
<filter-class>in.nit.filter.TestFilterOne</filter-class>
</filter>
<filter-mapping>
<filter-name>f1</filter-name>
<url-pattern>/sample</url-pattern>
</filter-mapping>
<filter>
<filter-name>f2</filter-name>
<filter-class>in.nit.filter.TestFilterTwo</filter-class>
<init-param>
<param-name>data</param-name>
<param-value>Hello</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>f2</filter-name>
<url-pattern>/sample</url-pattern>
</filter-mapping>
</web-app>
```

→ Enter Below URL in Browser:

<http://localhost:3031/ServletFilters-77/sample>

Console Output:

Init Filter one  
Init Filter Two: Hello

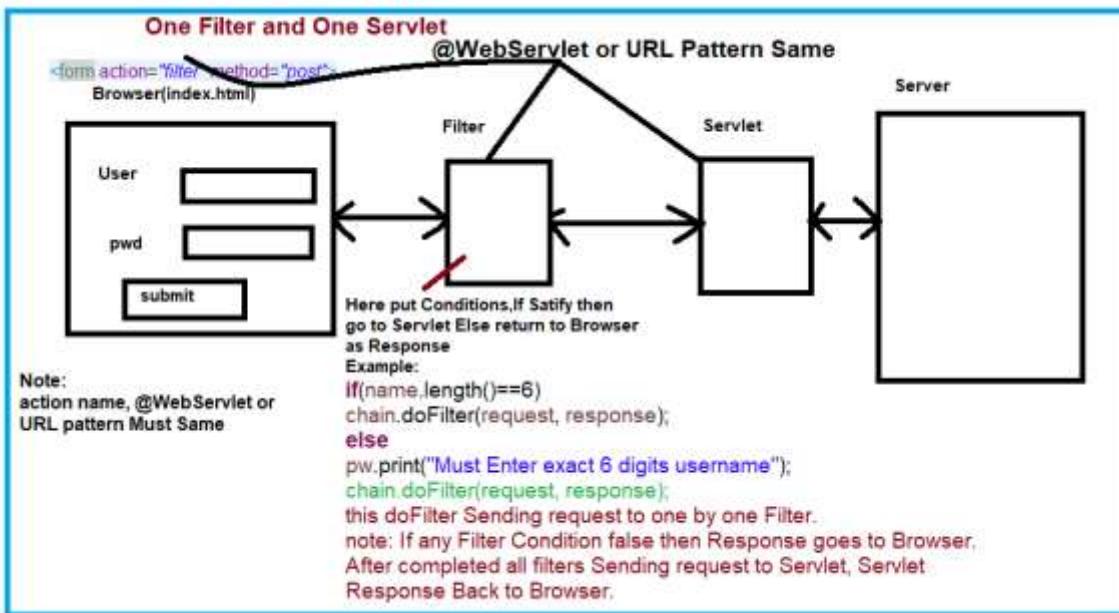
filter one pre-processing  
filter Two pre-processing  
SampleServlet doGet()  
filter Two post-processing  
filter one post-processing

<http://localhost:3031/ServletFilters-77/sample?data>

filter one pre-processing  
filter Two pre-processing  
SampleServlet doGet()  
filter Two post-processing  
filter one post-processing

---

-----Example2-----



-----index.html-----

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Filter</title>
</head>
<body>
 <form action="filter" method="post">
 <pre style="text-align: center">
UserName: <input type="text" name="user">

Password: <input type="password" name="pwd">

<input type="submit" value="Enter">
</pre>
 </form>
</body>
</html>

```

-----Welcome-Servlet-----

package in.nit.servlet;

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/filter")
public class WelcomeServlet extends HttpServlet {

 public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 PrintWriter pw=response.getWriter();
 response.setContentType("text/html");
 String user=request.getParameter("user");
 String pwd=request.getParameter("pwd");
 pw.print("Welcome to Servlet Filters!----->>> "+user+
"+pwd);
 }

 public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 doGet(request, response);
 }
}
```

}

-----FirstServletFilter-----

```
package in.nit.filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
@WebFilter("/filter")
public class FirstFilterServlet implements Filter {

 public void destroy() {

 }

 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
 String name=request.getParameter("user");
 PrintWriter pw=response.getWriter();
 if(name.length()==6) {
 chain.doFilter(request, response);
 }
 else {
```

```
 pw.print("Must Enter exact 6 digits username");
 }
}

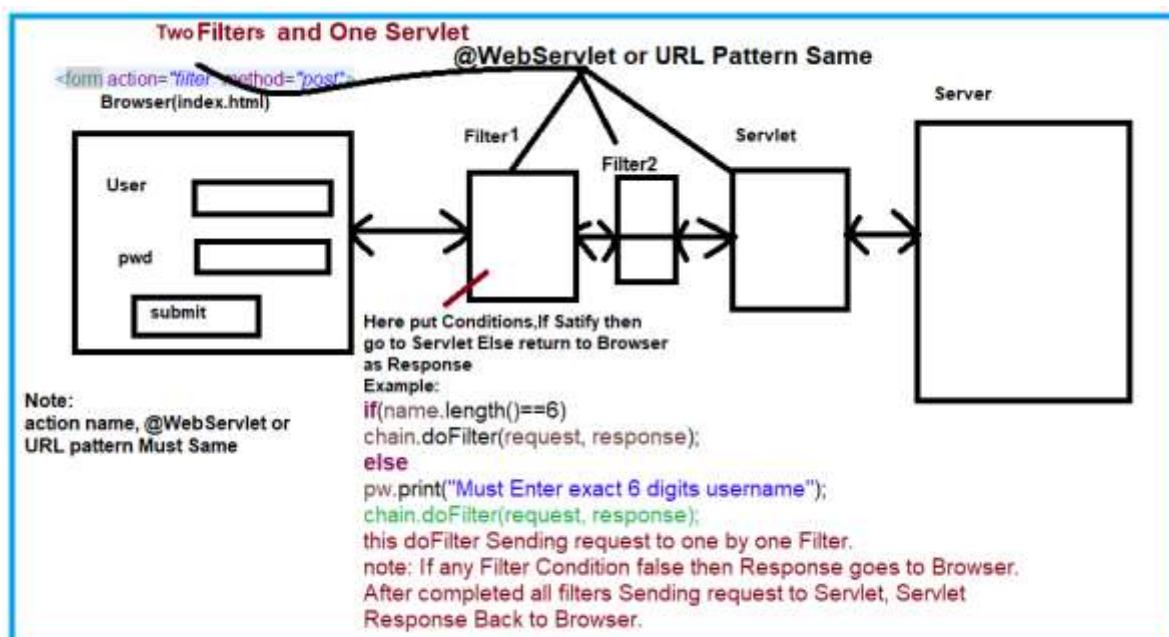
public void init(FilterConfig fConfig) throws ServletException {
}

}
-----Output-----
```

Enter username and pwd:

Welcome to Servlet Filters!---->>> Sankar 222

-----Example Three-----



Index.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Filter</title>
```

```
</head>
<body>
 <form action="filter" method="post">
 <pre style="text-align: center">
UserName: <input type="text" name="user">

Password: <input type="password" name="pwd">

<input type="submit" value="Enter">
</pre>
 </form>
</body>
</html>
```

-----WelcomeServlet-----

```
package in.nit.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/filter")
public class WelcomeServlet extends HttpServlet {

 public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 PrintWriter pw=response.getWriter();
 response.setContentType("text/html");
 String user=request.getParameter("user");
```

```
String pwd=request.getParameter("pwd");
pw.print("Welcome to Servlet Filters!---->>> "+user+
"+pwd);
}
```

```
public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 doGet(request, response);
}

-----FirstFilterServlet-----
```

```
package in.nit.filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
@WebFilter("/filter")
public class FirstFilterServlet implements Filter {
```

```
public void destroy() {

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
 String name=request.getParameter("user");
 PrintWriter pw=response.getWriter();
 if(name.length()==6) {
 chain.doFilter(request, response);
 }
 else {
 pw.print("First Filter.....Must Enter exact 6 digits
username");
 }
}

public void init(FilterConfig fConfig) throws ServletException {
}

}
```

-----SecondServletFilter-----

```
package in.nit.filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
```

```
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
@WebFilter("/filter")
public class SecondFilterServlet implements Filter {
 public void destroy() {
 }

 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
 String name=request.getParameter("user");
 String pwd=request.getParameter("pwd");
 PrintWriter pw=response.getWriter();
 if(name.equals("admin1")&& pwd.equals("admin1")) {
 chain.doFilter(request, response);
 }
 else {
 pw.print("Second Filter....username and pwd wrong");
 }
 }

 public void init(FilterConfig fConfig) throws ServletException {
 }
}
```

-----Output-----

Enter username and password in html page :

Welcome to Servlet Filters!---->>> admin1 admin1

-----Example Four-----

Index.html:

```
<form action="filter1" method="post">
```

Or

```
<form action="filter" method="post">
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Filter</title>
</head>
<body>
 <form action="filter1" method="post">
 <pre style="text-align: center">
UserName: <input type="text" name="user">

Password: <input type="password" name="pwd">

<input type="submit" value="Enter">
</pre>
 </form>
</body>
</html>
```

-----WelcomeServlet-----

```
package in.nit.servlet;
```

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/filter")
public class WelcomeServlet extends HttpServlet {

 public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 PrintWriter pw=response.getWriter();
 response.setContentType("text/html");
 String user=request.getParameter("user");
 String pwd=request.getParameter("pwd");
 pw.print("1--Welcome to Servlet Filters!---->>> "+user+
"+pwd);
 }

 public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 doGet(request, response);
 }
}
```

-----WelcomeServlet2-----

```
package in.nit.servlet;

import java.io.IOException;
```

```
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/filter1")
public class WelcomeServlet2 extends HttpServlet {

 public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 PrintWriter pw=response.getWriter();
 response.setContentType("text/html");
 String user=request.getParameter("user");
 String pwd=request.getParameter("pwd");
 pw.print("2--Welcome to Servlet Filters!----->>> "+user+
"+pwd);
 }

 public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
 doGet(request, response);
 }
}
```

-----FirstServletFilter-----

```
package in.nit.filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
@WebFilter("/filter")
public class FirstFilterServlet implements Filter {

 public void destroy() {

 }

 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
 String name=request.getParameter("user");
 PrintWriter pw=response.getWriter();
 if(name.length()==6) {
 chain.doFilter(request, response);
 }
 else {
 pw.print("First Filter.....Must Enter exact 6 digits
username");
 }
 }
}
```

```
}
```

```
public void init(FilterConfig fConfig) throws ServletException {
```

```
}
```

```
}
```

-----SecondFilterServlet-----

```
package in.nit.filter;
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import javax.servlet.Filter;
```

```
import javax.servlet.FilterChain;
```

```
import javax.servlet.FilterConfig;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.ServletRequest;
```

```
import javax.servlet.ServletResponse;
```

```
import javax.servlet.annotation.WebFilter;
```

```
@WebFilter({"filter","filter1"})
```

```
public class SecondFilterServlet implements Filter {
```

```
 public void destroy() {
```

```
}
```

```
 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
```

```
 String name=request.getParameter("user");
```

```
 String pwd=request.getParameter("pwd");
```

```
PrintWriter pw=response.getWriter();
if(name.equals("admin1")&& pwd.equals("admin1")) {
 chain.doFilter(request, response);
}
else {
 pw.print("Second Filter...username and pwd wrong");
}
}

public void init(FilterConfig fConfig) throws ServletException {
}

}
```

-----Output-----

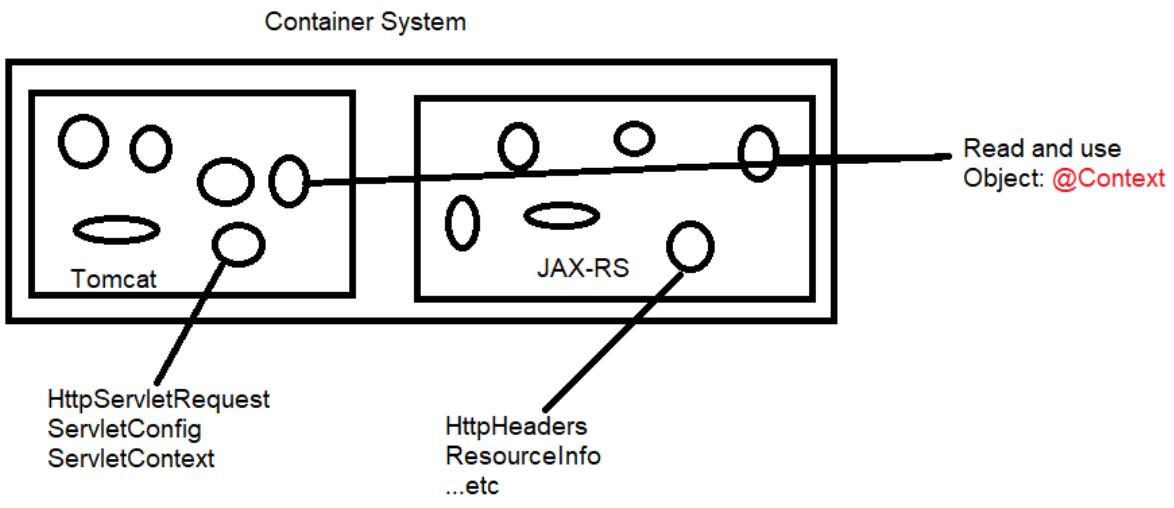
Enter username and pwd

```
<form action="filter" method="post">
1--Welcome to Servlet Filters!---->>> admin1 admin1
```

Enter username and pwd

```
<form action="filter1" method="post">
2--Welcome to Servlet Filters!---->>> admin1 admin1
```

---



## Reading Container Objects: `@Context`

- This annotation is used to read Objects which are created in Container.

Like `HttpServletRequest`, `ServletConfig`, `ServletContext`, `HttpHeaders`, `ResourceInfo`, `MediaType Info`....

- \*\*\* By using above annotation, we can only read Objects from Container. Programmer, neither creates nor destroy.

- Syntax to read container Objects in program

```
@Context
private Type objName;
```

### -----Example-----

```
@Context
private HttpServletRequest req;
@Context
private HttpHeaders headers;
```

### -----Code-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
```

```
<maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

In RestController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/home")
public class HomeRestController {
 @GET
 public String show(){
 System.out.println("From rest Controller");
 }
}
```

```
 return "Welcome To Home!";
 }
}
```

#### \*\*\*\*\* Filters with Context Objects

```
package in.nit.filter;

import java.io.IOException;

import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;

public class MyRequestFilter implements ContainerRequestFilter {
 @Context
 private HttpServletRequest req;//HAS-A
 @Context
 private HttpHeaders headers;//HAS-A

 @Override
 public void filter(ContainerRequestContext requestContext)
throws IOException {
 System.out.println("-----START #FROM FILTER WITH
CONTAINER OBJECTS-----");
 System.out.println(req.getRequestURL());
 System.out.println(req.getRequestURI());
 System.out.println(req.getContextPath());
 System.out.println(req.getMethod());
 System.out.println(req.getLocalAddr());

 System.out.println(headers.getRequestHeader("clientId"));
 System.out.println(headers.getRequestHeader("secret"));
 System.out.println("-----END#FROM FILTER WITH CONTAINER
OBJECTS-----");
}
}
```

### AppConfig(Equals to web.xml)

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.server.ResourceConfig;

import in.nit.filter.MyRequestFilter;

@Path("/rest")//url-pattern
public class AppConfig extends ResourceConfig{//FrontController
//Constructor
 public AppConfig() {
 packages("in.nit");//init-params
 register(MyRequestFilter.class);
 }
}
```

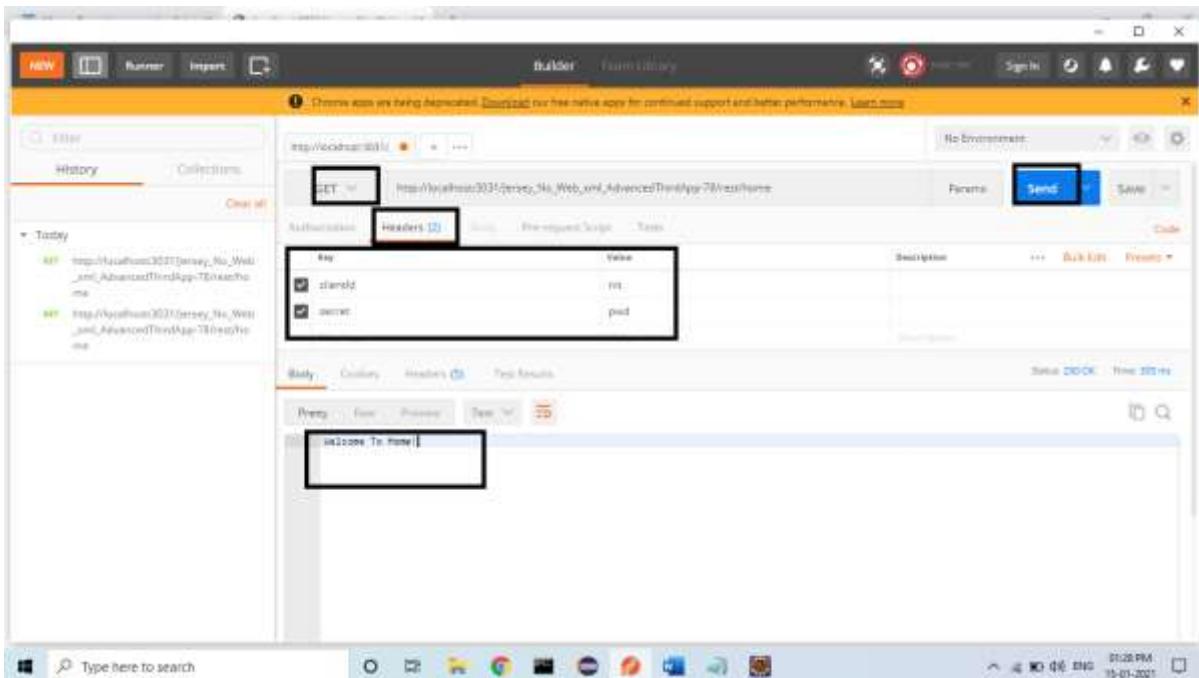
#### -----Output-----

In chrome

[http://localhost:3031/Jersey\\_No\\_Web\\_xml\\_AdvancedThirdApp-78/rest/home](http://localhost:3031/Jersey_No_Web_xml_AdvancedThirdApp-78/rest/home)

Welcome To Home!

#### -----POSTMAN SCREEN-----



-----In Console-----

-----START #FROM FILTER WITH CONTAINER OBJECTS-----  
`http://localhost:3031/Jersey_No_Web_xml_AdvancedThirdApp-78/rest/home`  
`/Jersey_No_Web_xml_AdvancedThirdApp-78/rest/home`  
`/Jersey_No_Web_xml_AdvancedThirdApp-78`  
GET  
0:0:0:0:0:0:1  
[nit]  
[pwd]  
-----END#FROM FILTER WITH CONTAINER OBJECTS-----  
From rest Controller

Q) When Object creates in Container?

- A) On App Startup/  
Only servlet Object while first request.

Advanced java web.xml :-- <load-on-startup>

- \*\*\*\*\* Scopes concept decides when Object creation is done in container.

→ Filters for Security Level-1 Programming: -

Example:

Define one Security Filter, that validates two Header Params

→ If those Header Params are null/empty

Then Http Status-400 BAD-REQUEST (Stop/Abort Execution)

Message -- Provide clientId/secret.

→ If those Header Params are nit/Sankar

Then Do not reject, just continue to RestController.

→ Else if any values for ClientId/secret

Then Http Status – 401 UN-AUTHORIZED(Stop/Abort Execution)

Message -- Invalid clientId/secret.

SQL:

```
mysql> use webservices;
Database changed
mysql> desc clienttab;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| cid | varchar(30) | YES | | NULL | |
| cser | varchar(30) | YES | | NULL | |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> insert into clienttab values('nit','raghu');
Query OK, 1 row affected (0.15 sec)

mysql> insert into clienttab values('sam','khan');
Query OK, 1 row affected (0.16 sec)

mysql> insert into clienttab values('RAM','SYED');
Query OK, 1 row affected (0.12 sec)

mysql> -
```

→ Create database webservices;

→ Use webservices;

→ create table clienttab(cid varchar(30), csecr varchar(30));

→ insert into clienttab values('nit','raghu');

→ insert into clienttab values('sam','khan');

→ insert into clienttab values('RAM','SYED');

-----code-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
 <dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.22</version>
 </dependency>
```

</dependencies>

In Rest Controller:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/home")
public class HomeRestController {
 @GET
 public String show() {

 return "WELCOME TO APP!!";
 }
}
```

In Database Connection:

```
package in.nit.conn;

import java.sql.Connection;
import java.sql.DriverManager;

public class DbConn {
 private static Connection con=null;
 static {
 try {
 Class.forName("com.mysql.cj.jdbc.Driver");

 con=DriverManager.getConnection("jdbc:mysql://localhost:3306/webservices","root","root");
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 }
 public static Connection getCon() {
```

```
 return con;
}
}
```

In Validator class:

```
package in.nit.validate;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

import in.nit.conn.DbConn;

public class ClientValidator {
 public static boolean exist(String clientId, String secret) {
 String query = "SELECT CID, CSER FROM CLIENTTAB WHERE CID=? AND CSER=?";
 boolean exist = false;
 try {
 PreparedStatement pt = DbConn.getCon().prepareStatement(query);
 pt.setString(1, clientId);
 pt.setString(2, secret);

 ResultSet rs = pt.executeQuery();
 if (rs.next()) {
 exist = true;
 }
 } catch (Exception e) {
 exist = false;
 e.printStackTrace();
 }
 return exist;
 }
}
```

In Security Filter:

```
package in.nit.filter;

import java.io.IOException;
import java.util.List;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.validate.ClientValidator;

public class SecurityFilter implements ContainerRequestFilter {
 @Context
 private static HttpHeaders headers;

 @Override
 public void filter(ContainerRequestContext requestContext)
 throws IOException {
 //Reading Two Header Params
 List<String> cid=headers.getRequestHeader("clientId");
 List<String> cser=headers.getRequestHeader("secret");

 //Check for Null or Empty
 if(cid==null || cid.isEmpty() || cser==null || cser.isEmpty()) {
 requestContext.abortWith(
 Response.status(Status.BAD_REQUEST)
 .entity("Provide clientId/secret")
 .build()
);
 return;
 }
 else {
 String clientId=cid.get(0);
 String secret=cser.get(0);
 //Check for Actual Data
 //if!("nit".equals(clientId)&&"raghu".equals(secret)) {
 if(ClientValidator.exist(clientId, secret)) {
 requestContext.abortWith(
 Response.status(Status.UNAUTHORIZED)
 }
 }
 }
}
```

```
 .entity("Invalid clientId/secret!!")
 .build()
);

}

return;
}

}
```

AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.server.ResourceConfig;

import in.nit.filter.SecurityFilter;
//It is equals to writing web.xml
@Path("/rest")//URL-Patterns
public class AppConfig extends ResourceConfig{//FC
 public AppConfig() {
 packages("in.nit");
 register(SecurityFilter.class);
 }
}
```

-----Output-----

In chrome:

[http://localhost:3031/Jersey\\_No\\_Web\\_xml MySql\\_AdvancedFourthApp-79/rest/home](http://localhost:3031/Jersey_No_Web_xml MySql_AdvancedFourthApp-79/rest/home)

Provide clientId/secret

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

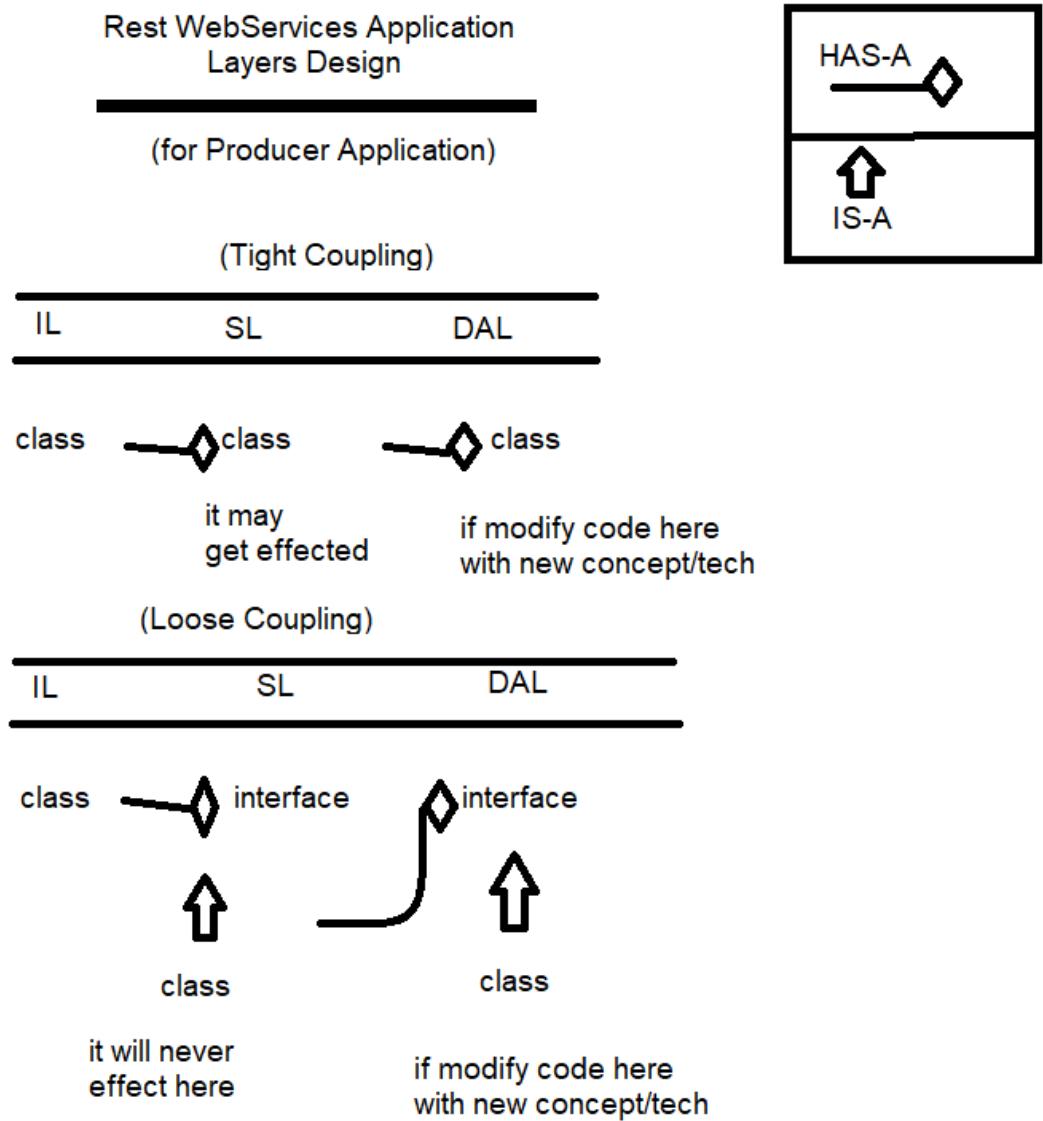
-----PostMan Screen-----

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'History', there are entries for 'Today' and 'Yesterday'. The 'Today' section contains four entries, each with a green success icon and a URL starting with 'http://localhost:3031/jersey\_hi\_web...'. The 'Yesterday' section also contains four entries with similar URLs. The main workspace shows a GET request to 'http://localhost:3031/jersey\_hi\_web...'. The 'Headers' tab is selected, showing two entries: 'clientid' with value '111' and 'secret' with value 'rghf'. Below the headers, the 'Body' tab is selected, showing a JSON object with 'clientid' and 'secret' fields. The 'Test' tab shows a passed test: 'WELCOME TO APP!'. The status bar at the bottom right indicates 'Status: 200 OK' and 'Time: 48 ms'.

This screenshot is identical to the one above, showing the same Postman interface. The history sidebar and the current request configuration are the same. However, the 'Test' tab now displays an error message: 'Unauthorised: clientId/secret!'. The status bar at the bottom right indicates 'Status: 401 Unauthorized' and 'Time: 48 ms'.

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

The screenshot shows the Postman application interface. On the left, there's a sidebar titled 'History' with sections for 'Today' and 'Yesterday'. Under 'Today', there are four items listed, each with a green icon and a URL like 'http://localhost:3031/jersey\_no\_web\_xml\_MySQL\_AdvancedAuth/app/TB\_enumeration'. Under 'Yesterday', there are three items with similar URLs. The main area is titled 'Builder' and shows a 'GET' request to 'http://localhost:3031/jersey\_no\_web\_xml\_MySQL\_AdvancedAuth/app/TB\_enumeration'. The 'Headers' tab is selected, showing two entries: 'clientId' with value '11' and 'secret' with value '123456'. Below the headers, the 'Body' tab is selected, showing a JSON payload: { "Provide": "clientId/secret" }. The bottom status bar indicates 'Status: 403 Bad Request' and 'Time: 54 ms'.



To develop application, in real time process, these are the 4 layers used.

1. Data Access Layer : DAL
2. Service Layer : SL
3. Presentation Layer : PL
4. Integration Layer : IL

→ Generally, one layer indicates one type of Operations  
 DAL- Used for Database Operations(Select, update, insert, delete).  
 SL - Used Calculations, logics, validations, Sorting...etc.  
 PL - Used to display final message/details at UI to consumer  
 IL - Link our App with other applications (Data Interchange).

→ We are writing RestController comes under IL.

- Here, we want to implement DAL and SL also link with IL which is implemented using HK2 Dependency.
- \*\*\*\* Coupling: Connecting Layer(classes) is called as Coupling  
This is Implemented using HAS-A(Association).

Types of Coupling: (2 types) [we can use any type to get final Output]

---

| a. Tight Coupling                                        |  | b. Loose Coupling****                                       |
|----------------------------------------------------------|--|-------------------------------------------------------------|
| It is implemented between classes only                   |  | It is implemented between class and interface               |
| If we modify one Layer class it may effect another class |  | If we modify one Layer class it never effects another class |

\*\*\* Note: Code to get modified or effected with new Designing/concepts/technologies comes, then. Otherwise no.

- \*\* One class Object creating in another class(Tight Coupling)
- One class Object is provided at runtime as Upcasted.  
To interface reference and provided to another class (Loose Coupling).

-----Example Tight Coupling-----

```
class A{}

class B{
//Direct Object
A a=new A();//Tight Couple
}
```

---

-----Example Loose Coupling-----

```
interface M{}
```

class A implements M{}

---

→ Container did like this:

M om=new A(); //Object Creation and Upcasting.

```
class B{
M om; //Runtime container Object is Provided.
}
```

---

Note: AbstractBinder is provided by HK2, this is going to create object to layer (child) class and upcast to (parent) interface at Runtime.

AbstractBinder (abstract class)  
void configure(); (abstract method).

-----Anonymous Inner class Syntax-----

```
new ClassName(){
//Override abstract method
}

new AbstractBinder(){
public void configure(){
 //IEmpService service=new EmpServiceImpl();
 bind(EmpServiceImpl.class).to(IEmpService.class);
}
}
```

---

#### Coding Steps

1. For Given module design and code for Layers
  - a. Service Layer Interface (I\_\_\_\_\_ Service)
  - b. Service Layer Impl class(\_\_\_\_\_  
ServiceImpl)
  - c. Create IS-A between Service Interface and Impl (implements keyword).
  
- d. Define Integration Layer RestController (class)
- e. Make HAS-A between RestController and Service interface

- f. Apply @Inject (Which looks for Object at runtime and provide to reference variable)
2. Specify Object creation steps at FrontController (AppConfig)
  - a. Configure Object details at FC class(AppConfig).  
Using abstractBinder
  - b. Override method configure() and specify code in below format  
bind(ClassName.class).to(InterfaceName.class);  
is equals to Interface ob=new ClassName();//Upcasting.

-----Example-----

-----In pom.xml File-----

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
```

```
<version>2.30</version>
</dependency>

</dependencies>
```

-----Service Layer Code-----

```
package in.nit.service;

public interface IEmpService {
 public String saveEmp();
}
```

Impl:

```
package in.nit.service.impl;

import in.nit.service.IEmpService;

public class EmpServiceimpl implements IEmpService {

 @Override
 public String saveEmp() {

 return ": FROM SERVICE SAVE EMP :";
 }

}
```

RestController Code:

```
package in.nit.controller;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

import in.nit.service.IEmpService;
```

```
@Path("/emp")
public class EmpRestController {
 @Inject//gets impl class Object at runtime
 private IEmpService service;//HAS-A

 @GET
 @Path("/show")
 public String show() {

 return " FROM RESTCONTROLLER=> "+service.saveEmp();
 }
}
```

AppConfig(FrontController):

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.service.IEmpService;
import in.nit.service.impl.EmpServiceImpl;
@Path("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 packages("in.nit");//nit-param
 register(new AbstractBinder() {

 @Override
 public void configure() {
 bind(EmpServiceImpl.class).to(IEmpService.class);
 }
 });
 }
}
```

Run Application and Enter URL:

-----Output-----

In chrome:

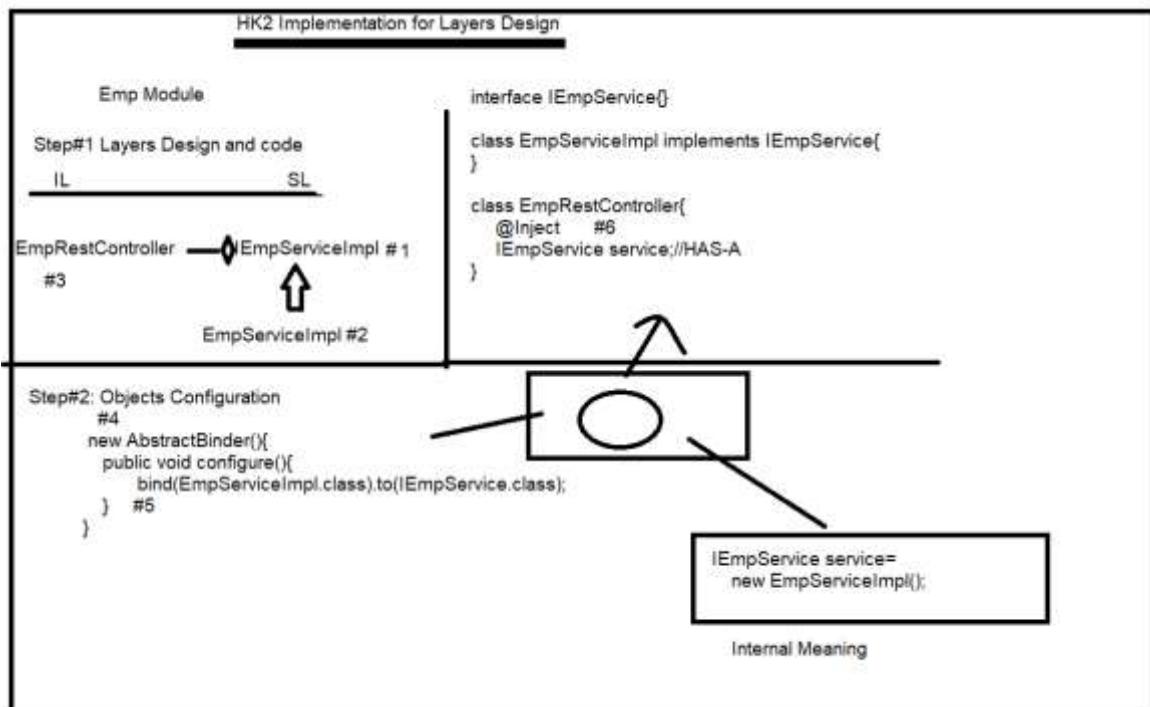
<http://localhost:3031/layeredProducerApp-80/rest/emp/show>

FROM RESTCONTROLLER=> : FROM SERVICE SAVE EMP :

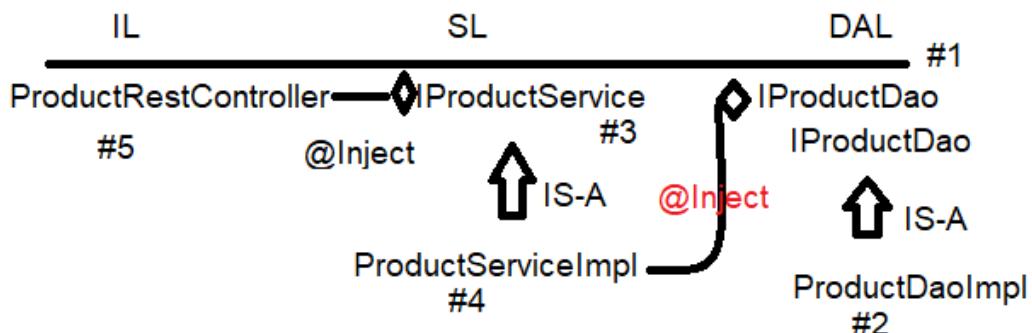
-----In POSTMAN SCREEN-----

The screenshot shows the Postman interface with a GET request to `http://localhost:3031/layeredProducerApp-80/rest/emp/show`. The response body displays the text `FROM RESTCONTROLLER=> : FROM SERVICE SAVE EMP .`

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>



## Design: Product Module



## AbstractBinder code:

```
bind(ProductDaoImpl.class).to(IProductDao.class);
bind(ProductServiceImpl.class).to(IProductService.class);
```

## Module: Product

## Step#1 Design and Code for Layers.

## Use @Inject on HAS-A Variable

## Step2: Configure in AbstractBinder

bind(ClassName.class).to(InterfaceName.class)

### -----Coding Steps-----

1. IProductDao.java
2. ProductDaoImpl.java
3. IProductService.java
4. ProductServiceImpl.java
5. ProductRestController.java
6. AppConfig.java

### -----Code-----

#### In pom.xml File

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

In Data Access Layer(DAL):

Interface:

```
package in.nit.dao;

public interface IProductDao {
 public String show();
}
```

Impl:

```
package in.nit.dao;

public class ProductDaoImpl implements IProductDao {

 @Override
 public String show() {

 return "from DAo";
 }

}
```

Service Layer(SL):

```
package in.nit.service;

public interface IProductService {
 public String showService();
}
```

Impl:

```
package in.nit.service.impl;

import javax.inject.Inject;

import in.nit.dao.IProductDao;
import in.nit.service.IProductService;

public class ProductServiceImpl implements IProductService {
 @Inject
 private IProductDao dao;//HAS-A
 @Override
 public String showService() {
 return ": Service :" + dao.show();
 }
}
```

RestController:

```
package in.nit.controller;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

import in.nit.service.IProductService;

@Path("/products")
public class ProductRestController {
 @Inject//gets impl class Object at runtime
}
```

```
private IProductService service;//HAS-A

@GET
public String findMsg() {

 return " FROM RESTCONTROLLER=> "+service.showService();
}
}
```

AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IProductDao;
import in.nit.dao.ProductDaoImpl;
import in.nit.service.IProductService;
import in.nit.service.impl.ProductServiceImpl;
@Override
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 packages("in.nit");//nit-param
 register(new AbstractBinder() {
 @Override
 public void configure() {
 bind(ProductDaoImpl.class).to(IProductDao.class);

 bind(ProductServiceImpl.class).to(IProductService.class);
 }
 });
 }
}
```

-----output-----

In chrome:

<http://localhost:3031/layeredProducerApp-81/rest/products>

FROM RESTCONTROLLER=> : Service :from Dao

-----In Postman screen-----

The screenshot shows the Postman interface. On the left, there's a sidebar with a history of requests. In the main area, a GET request is being viewed for the URL <http://localhost:3031/layeredProducerApp-81/rest/products>. The 'Authorization' tab is selected, showing 'No Auth'. The 'Body' tab is selected in the preview pane, which displays the response: 'FROM RESTCONTROLLER=> : Service :from Dao'. The status bar at the bottom right indicates 'Status 200 OK' and 'Time 31 ms'.

## Project#1 JAX-RS+JDBC Integration

(Producer)

-----Coding Files-----

1. Student.java
2. DbConn.java
3. IStudentDao.java
4. StudentDaoImpl.java
5. IStudentService.java
6. StudentServiceImpl.java
7. StudentRestController.java
8. AppConfig.java

-----MySQL Setup SQL's-----

```
mysql> use webservices
```

```
mysql> create table students(sid int,sname varchar(30),scourse
varchar(30),sfee double,sdiscount double);
```

```
mysql> insert into students
values(1,'Sankar','RESTFUL',3000.00,300.0);
```

```
mysql> insert into students
values(2,'raju','hibernate',4000.00,400.0);
```

```
mysql> insert into students
values(3,'sam','SPRING',5000.00,500.0);
```

```
mysql> commit;
```

---

-----Code-----

In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.22</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok --
>
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson --
>
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
<dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
</dependency>
</dependencies>
```

## Database Connection:

```
package in.nit.db;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConn {
private static Connection conn=null;

static {
 try {
 Class.forName("com.mysql.jdbc.Driver");

 conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/
webservices","root","root");
 } catch (Exception e) {
 e.printStackTrace();
 }
}
Public static Connection getConnection() {

 return conn;
}
}
```

Model class:

```
package in.nit.model;

import lombok.Data;

@Data
public class Student {
private Integer stdId;
private String stdName;
private String stdSource;
private Double stdFee;
private Double stdDiscount;
}
```

DAO CODE:

```
package in.nit.dao;

import in.nit.model.Student;

public interface IStudentDao {
 public Integer saveStudent(Student s);
}
```

IMPL:

```
package in.nit.dao;

import java.sql.PreparedStatement;

import in.nit.db.DBConn;
import in.nit.model.Student;

public class StudentImpl implements IStudentDao {

 @Override
 public Integer saveStudent(Student s) {
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?)";
 int count=0;
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,s.getStdId());
 ps.setString(2,s.getStdName());
 ps.setString(3,s.getStdCourse());
 ps.setDouble(4,s.getStdFee());
 ps.setDouble(5,s.getStdDiscount());
 count=ps.executeUpdate();
 } catch (Exception e) {
 e.printStackTrace();
 }
 return count;
 }
}
```

## ServiceLayer(SL):

### Interface:

```
package in.nit.service;

import in.nit.model.Student;

public interface IStudentService {
 public Integer saveStudent(Student s);
}
```

### Impl:

```
package in.nit.service.impl;

import javax.inject.Inject;

import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;

public class StudentServiceImpl implements IStudentService {
 @Inject
 private IStudentDao dao; //HAS-A

 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
 //Calling Data Access layer(DAL)

 return dao.saveStudent(s);
 }
}
```

## Controller:

```
package in.nit.controller;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject//gets impl class Object at runtime
 private IStudentService service;//HAS-A

 //Save Student
 @POST
 @Consumes("application/json")
 public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED)//201-
Success
 .entity("Student Saved!")
 .build();
 }
 else {
 resp=Response.status(Status.BAD_REQUEST)//400-
Wrong Input
 .entity("Student Not Saved!")
 .build();
 }
 } catch(Exception e) {
```

```
 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Student Not Saved!...Error")
 .build();
 e.printStackTrace();
 }
 return resp;
 }
}
```

AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IStudentDao;
import in.nit.dao.StudentDaolmpl;
import in.nit.service.IStudentService;
import in.nit.service.impl.StudentServicempl;
@Path("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 //this.packages("in.nit");//nit-param
 //this.register(new AbstractBinder() {
 //or
 packages("in.nit");//nit-param
 register(new AbstractBinder() {
 @Override
 public void configure() {
 bind(StudentDaolmpl.class).to(IStudentDao.class);

 bind(StudentServicempl.class).to(IStudentService.class);
 }
 });
 }
}
```

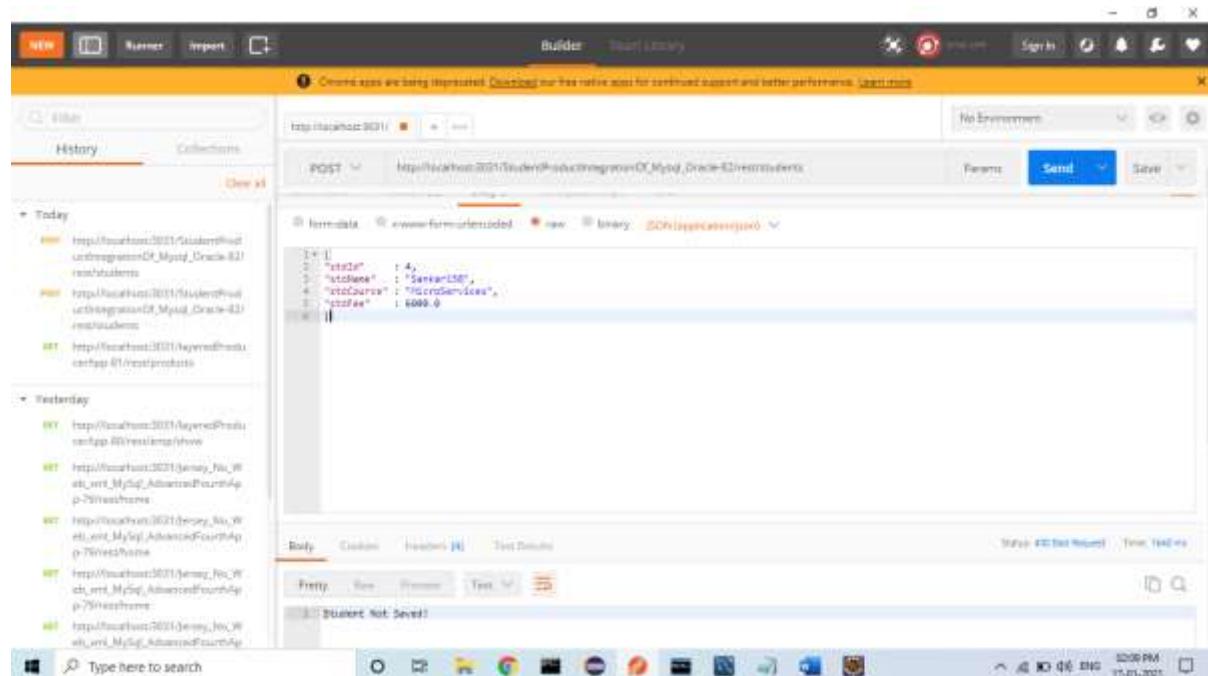
-----output-----

In Chrome:

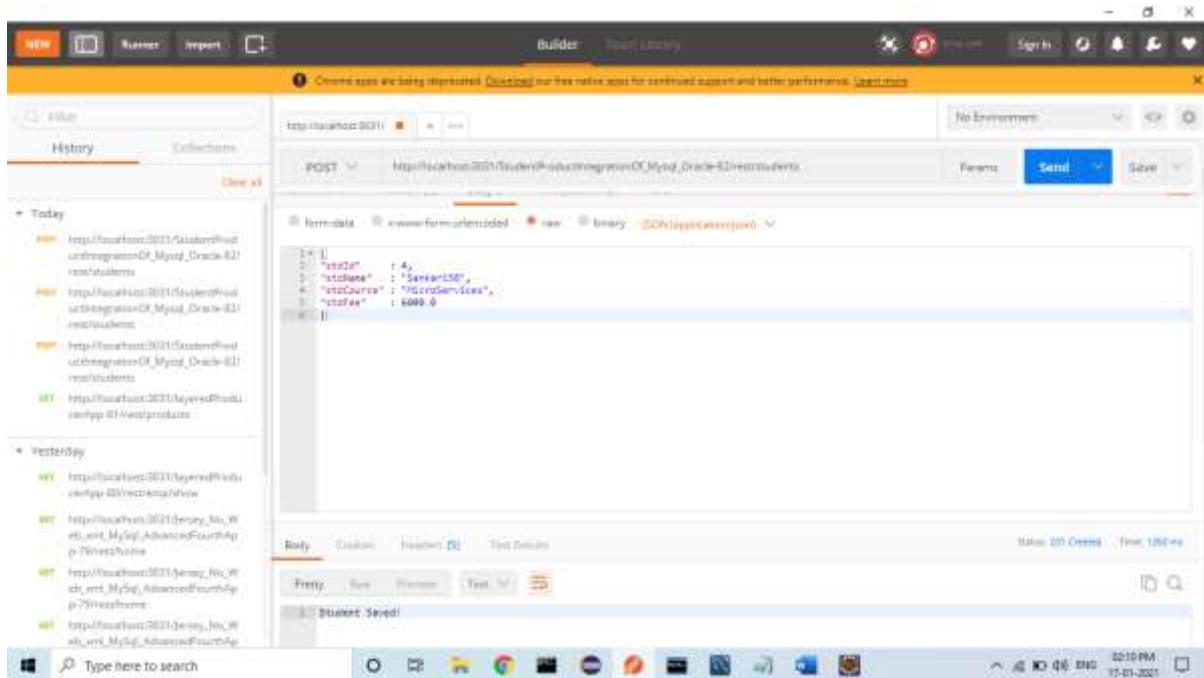
[http://localhost:3031/StudentProductIntegrationOf\\_Mysql\\_Oracle-82/rest/students](http://localhost:3031/StudentProductIntegrationOf_Mysql_Oracle-82/rest/students)

In chrome not possible to see output.

In Postman Tool:

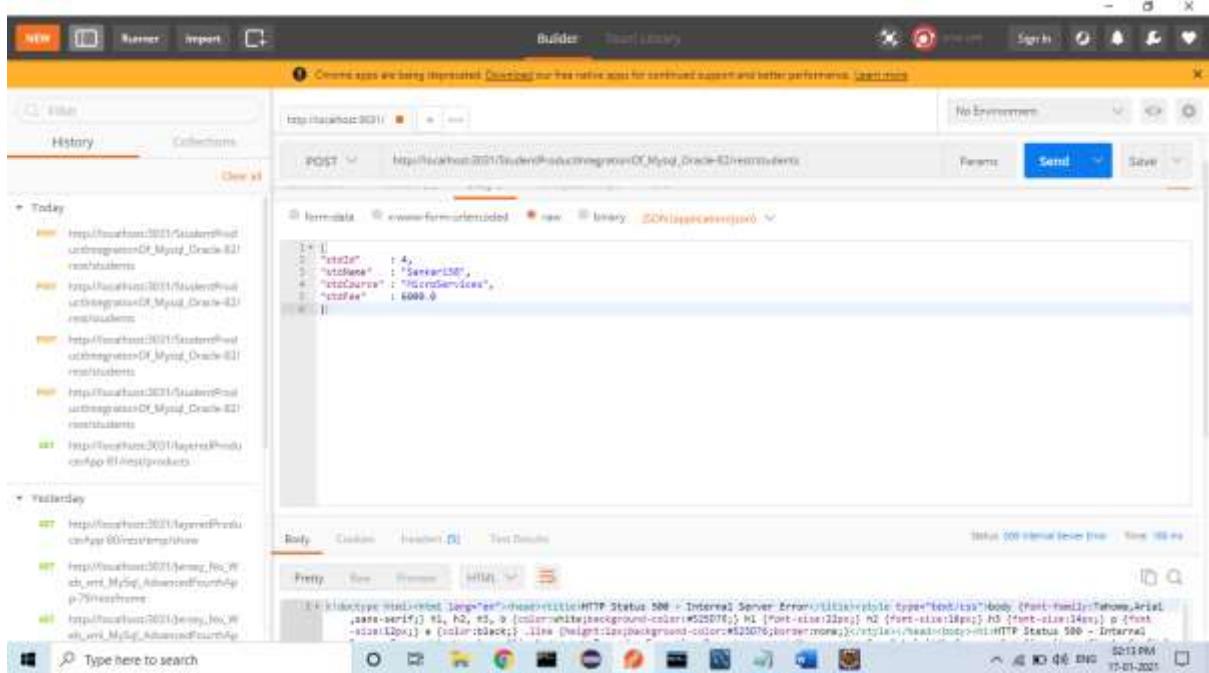


Above Screen table name Wrong.



mysql> select \* from Students;

```
+-----+-----+-----+
| sid | sname | scourse | sfee | sdiscount |
+-----+-----+-----+
| 1 | Sankar | RESTFUL | 3000 | 300 |
| 2 | raju | hibernate | 4000 | 400 |
| 3 | sam | SPRING | 5000 | 500 |
| 4 | Sankar158 | MicroServices | 6000 | 600 |
+-----+-----+-----+
4 rows in set (0.03 sec)
```



## Above Screen Semicolon Missing

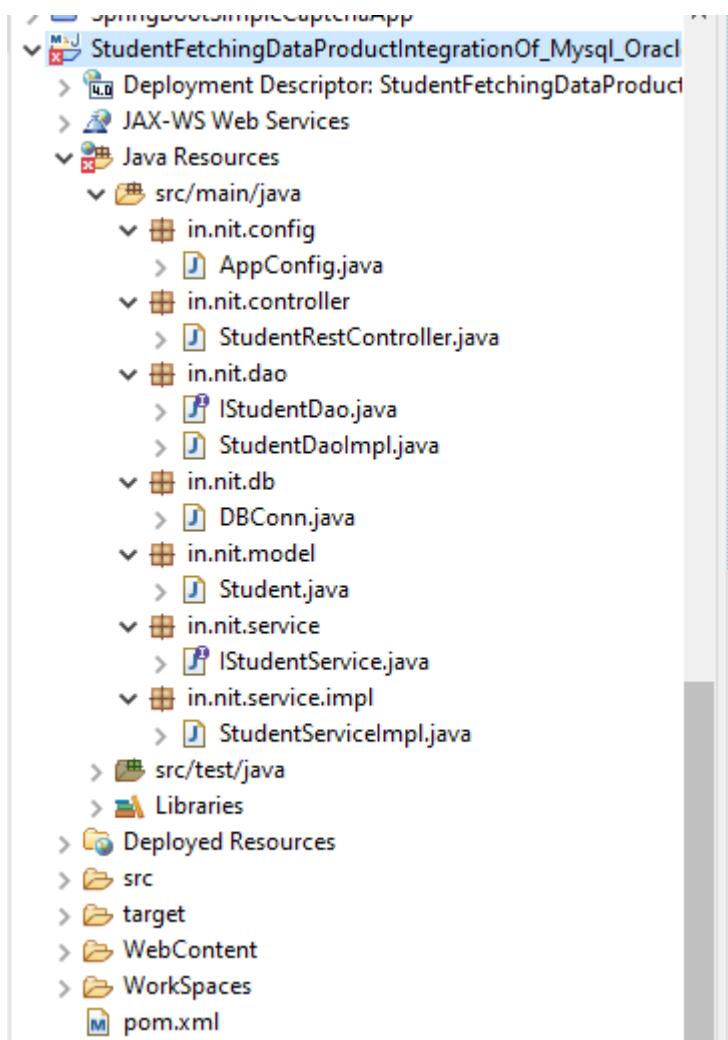
`@Consumes` specifies what MIME type a resource accepts from the client. `@Produces`, however, specifies what MIME type a resources gives to the client. For example, a resource might accept `application/json` (`@Consumes`) and return `text/plain` (`@Produces`).

<https://docs.oracle.com/cd/E19226-01/820-7627/6nisfjmko/index.html>

Share Improve this answer Follow

edited Jan 20 '16 at 2:24

answered Jan 28 '16 at 18:12



## Model:

```
package in.nit.model;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
```

```
@Data
@AllArgsConstructor
public class Student {
 private Integer stdId;
 private String stdName;
```

```
private String stdSource;
private Double stdFee;
private Double stdDiscount;
}
```

DBConnection:

```
package in.nit.db;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
```

```
public class DBConn {
 private static Connection conn=null;
```

```
 static {
 try {
 Class.forName("com.mysql.jdbc.Driver");
```

```
 conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/
webservices","root","root");
```

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

```
 public static Connection getConnection() {

 return conn;
 }
```

}

DAO:

```
package in.nit.dao;
```

```
import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentDao {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
}
```

IMPL:

```
package in.nit.dao;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
import in.nit.db.DBConn;
import in.nit.model.Student;
```

```
public class StudentDaolmpl implements IStudentDao {
 //1.Fetching Data From Database
```

@Override

```
public Integer saveStudent(Student s) {
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?);
 int count=0;
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,s.getStdId());
 ps.setString(2,s.getStdName());
 ps.setString(3,s.getStdCourse());
 ps.setDouble(4,s.getStdFee());
 ps.setDouble(5,s.getStdDiscount());
 count=ps.executeUpdate();
 } catch (Exception e) {
 e.printStackTrace();
 }
 return count;
}
```

@Override

```
public List<Student> getAllStudents() {
 String query="SELECT * FROM STUDENTS";
 List<Student> list=null;
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ResultSet rs=ps.executeQuery();
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
list=new ArrayList<>();
while(rs.next()) {
 list.add(new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount")
));
}
catch(Exception e) {
 e.printStackTrace();
}
return list;
}

}
```

## SERVICE:

```
package in.nit.service;

import java.util.List;

import in.nit.model.Student;
```

```
public interface IStudentService {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
}
```

IMPL:

```
package in.nit.service.impl;
```

```
import java.util.Collections;
import java.util.List;
```

```
import javax.inject.Inject;
```

```
import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;
```

```
public class StudentServiceImpl implements IStudentService {
```

```
 @Inject
 private IStudentDao dao;//HAS-A
```

```
 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
```

//Calling Data Access layer(DAL)

```
 return dao.saveStudent(s);
}

@Override
public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 Collections.sort(list, (o1,o2)->
 o1.getStdId()-o2.getStdId()
);//ID-ASC
 return list;
}

AppConfig:
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IStudentDao;
import in.nit.dao.StudentDaoImpl;
import in.nit.service.IStudentService;
```

```
import in.nit.service.impl.StudentServiceImpl;

@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 //this.packages("in.nit");//nit-param
 //this.register(new AbstractBinder() {
 //or
 packages("in.nit");//nit-param
 register(new AbstractBinder() {
 @Override
 public void configure() {
 bind(StudentDaoImpl.class).to(IStudentDao.class);

 bind(StudentServiceImpl.class).to(IStudentService.class);
 }
 });
 }
}
}
```

Controller:

```
package in.nit.controller;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;
```

```
import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject//gets impl class Object at runtime
 private IStudentService service;//HAS-A

 //Save Student
 @POST
 @Consumes("application/json")
 public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED)//201-
Success
 .entity("Student Saved!")
 .build();
 }
 else {
 resp=Response.status(Status.BAD_REQUEST)//400-
Wrong Input
 .entity("Student Not Saved!")
 .build();
 }
 } catch(Exception e) {
 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 .entity("Student Not Saved!...Error")
 .build();
 e.printStackTrace();
 }
 return resp;
 }
 //Fetch Student
 @GET
 @Produces("application/json")
 public Response getAllStudents(Student student) {
```

```
Response resp=null;
try {
 List<Student> list=service.getAllStudents();
 resp=Response
 .status(Status.OK)
 .entity(list)
 .build();
}
catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 .entity("Unable to Fetch Data....")
 .build();
 e.printStackTrace();
}
return resp;
}
}
```

---

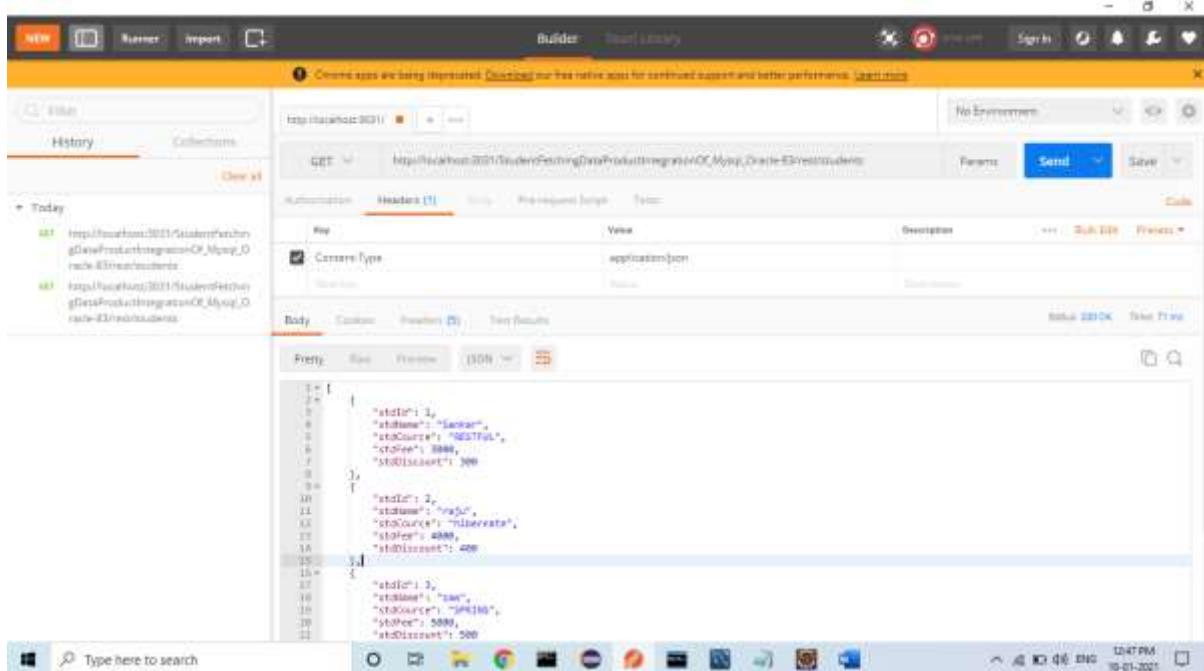
### Output

---

In Chrome:

[http://localhost:3031/StudentFetchingDataProductIntegrationOf\\_Mysql\\_Oracle-83/rest/students](http://localhost:3031/StudentFetchingDataProductIntegrationOf_Mysql_Oracle-83/rest/students)

```
[{"stdId":1,"stdName":"Sankar","stdSource":"RESTFUL","stdFee":3000.0,"stdDiscount":300.0}, {"stdId":2,"stdName":"raju","stdSource":"hibernate","stdFee":4000.0,"stdDiscount":400.0}, {"stdId":3,"stdName":"sam","stdSource":"SPRING","stdFee":5000.0,"stdDiscount":500.0}, {"stdId":4,"stdName":"Sankar158","stdSource":"MicroServices","stdFee":6000.0,"stdDiscount":600.0}]
```



```
[
 {
 "stdId": 1,
 "stdName": "Sankar",
 "stdSource": "RESTFUL",
 "stdFee": 3000,
 "stdDiscount": 300
 },
 {
 "stdId": 2,
 "stdName": "raju",
 "stdSource": "hibernate",
 "stdFee": 4000,
 "stdDiscount": 400
 },
]
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
{
 "stdId": 3,
 "stdName": "sam",
 "stdSource": "SPRING",
 "stdFee": 5000,
 "stdDiscount": 500
},
{
 "stdId": 4,
 "stdName": "Sankar158",
 "stdSource": "MicroServices",
 "stdFee": 6000,
 "stdDiscount": 600
}
```

---

Modified:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Student {
```

```
private Integer stdId;
private String stdName;
private String stdSource;
private Double stdFee;
private Double stdDiscount;
}
```

Modified:

```
package in.nit.dao;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
import in.nit.db.DBConn;
import in.nit.model.Student;
```

```
public class StudentDaoImpl implements IStudentDao {
//1.Fetching Data From Database
 @Override
 public Integer saveStudent(Student s) {
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?)";
 int count=0;
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,s.getStdId());
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
ps.setString(2,s.getStdName());
ps.setString(3,s.getStdCourse());
ps.setDouble(4,s.getStdFee());
ps.setDouble(5,s.getStdDiscount());
count=ps.executeUpdate();
}
} catch (Exception e) {
 e.printStackTrace();
}
return count;
}
```

```
@Override
public List<Student> getAllStudents() {
 String query="SELECT * FROM STUDENTS";
 List<Student> list=null;
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ResultSet rs=ps.executeQuery();
 list=new ArrayList<>();
 /*
 * while(rs.next()) {
 * list.add(new Student(
 * rs.getInt("sid"),
 * rs.getString("sname"),
 * rs.getString("scourse"),
 * rs.getDouble("sfee"),
 * rs.getDouble("sdiscount")));
 */
 }
}
```

```
* }
```

```
*/
```

```
while(rs.next()) {
```

```
//a.Read Data From ResultSet Row
```

```
int stdId=rs.getInt("sid");
```

```
String stdName=rs.getString("sname");
```

```
String stdCourse=rs.getString("scourse");
```

```
Double stdFee=rs.getDouble("sfee");
```

```
Double stdDiscount=rs.getDouble("sdiscount");
```

```
//b. Convert Data into One Student Class Object
```

```
Student s=new Student();
```

```
s.setStdId(stdId);
```

```
s.setStdName(stdName);
```

```
s.setStdCourse(stdCourse);
```

```
s.setStdFee(stdFee);
```

```
s.setStdDiscount(stdDiscount);
```

```
//c.Add Student class Object List
```

```
list.add(s);
```

```
}
```

```
}
```

```
catch(Exception e) {
```

```
e.printStackTrace();
```

```
}
```

```
return list;
```

```
}
```

}

Modified:

```
package in.nit.service.impl;
```

```
import java.util.Collections;
```

```
import java.util.List;
```

```
import javax.inject.Inject;
```

```
import in.nit.dao.IStudentDao;
```

```
import in.nit.model.Student;
```

```
import in.nit.service.IStudentService;
```

```
public class StudentServiceImpl implements IStudentService {
```

```
 @Inject
```

```
 private IStudentDao dao;//HAS-A
```

```
 @Override
```

```
 public Integer saveStudent(Student s) {
```

```
 //Calculations
```

```
 Double fee=s.getStdFee();
```

```
 Double dis=fee*10/100.0;
```

```
 s.setStdDiscount(dis);
```

```
 //Calling Data Access layer(DAL)
```

```
 return dao.saveStudent(s);
 }

 @Override
 public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getStdId()-
 o2.getStdId());//ID-ASC
 Collections.sort(list, (o1,o2)->o2.getStdId()-
 o1.getStdId());//ID-DESC
 return list;
 }

}
```

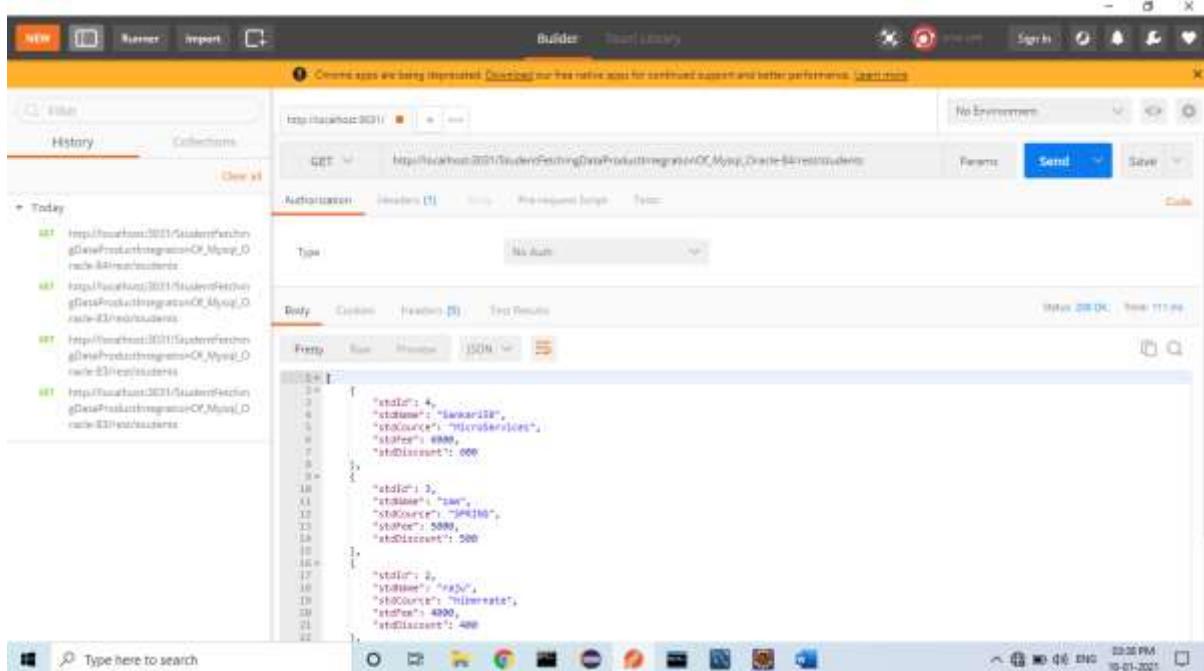
---

## Output:

[http://localhost:3031/StudentFetchingDataProductIntegrationOf\\_Mysql\\_Oracle-84/rest/students](http://localhost:3031/StudentFetchingDataProductIntegrationOf_Mysql_Oracle-84/rest/students)

```
[{"stdId":4,"stdName":"Sankar158","stdSource":"MicroServices","stdFee":6000.0,"stdDiscount":600.0}, {"stdId":3,"stdName":"sam","stdSource":"SPRING","stdFee":5000.0,"stdDiscount":500.0}, {"stdId":2,"stdName":"raju","stdSource":"hibernate","stdFee":4000.0,"stdDiscount":400.0}, {"stdId":1,"stdName":"Sankar","stdSource":"RESTFUL","stdFee":3000.0,"stdDiscount":300.0}]
```

-----Postman Screen-----



Descending :

```
[
 {
 "stdId": 4,
 "stdName": "Sankar158",
 "stdSource": "MicroServices",
 "stdFee": 6000,
 "stdDiscount": 600
 },
 {
 "stdId": 3,
 "stdName": "sam",
 "stdSource": "SPRING",
 "stdFee": 5000,
 "stdDiscount": 500
 },
```

```
{
 "stdId": 2,
 "stdName": "raju",
 "stdSource": "hibernate",
 "stdFee": 4000,
 "stdDiscount": 400
},
{
 "stdId": 1,
 "stdName": "San",
 "stdSource": "RESTFUL",
 "stdFee": 3000,
 "stdDiscount": 300
}
]
```

---

Name Ascending Order:

```
package in.nit.service.impl;

import java.util.Collections;
import java.util.List;

import javax.inject.Inject;

import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;
```

```
public class StudentServiceImpl implements IStudentService {
 @Inject
 private IStudentDao dao;//HAS-A

 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
 //Calling Data Access layer(DAL)

 return dao.saveStudent(s);
 }

 @Override
 public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getStdId()-
 o2.getStdId());//ID-ASC
 //Collections.sort(list, (o1,o2)->o2.getStdId()-
 o1.getStdId());//ID-DESC
 Collections.sort(list, (o1,o2)-
 >o1.getStdName().compareTo(o2.getStdName()));//Name-ASC
 return list;
 }
}
```

}

Output:

[

{

"stdId": 1,  
"stdName": "Sankar",  
"stdSource": "RESTFUL",  
"stdFee": 3000,  
"stdDiscount": 300

},

{

"stdId": 4,  
"stdName": "Sankar158",  
"stdSource": "MicroServices",  
"stdFee": 6000,  
"stdDiscount": 600

},

{

"stdId": 2,  
"stdName": "raju",  
"stdSource": "hibernate",  
"stdFee": 4000,  
"stdDiscount": 400

},

{

"stdId": 3,  
"stdName": "sam",

```
"stdSource": "SPRING",
"stdFee": 5000,
"stdDiscount": 500
}
]
```

---

Name Descending :

```
package in.nit.service.impl;

import java.util.Collections;
import java.util.List;

import javax.inject.Inject;

import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;

public class StudentServiceImpl implements IStudentService {
 @Inject
 private IStudentDao dao;//HAS-A

 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
```

```
s.setStdDiscount(dis);
//Calling Data Access layer(DAL)

return dao.saveStudent(s);
}

@Override
public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getStdId()-
 o2.getStdId());//ID-ASC
 //Collections.sort(list, (o1,o2)->o2.getStdId()-
 o1.getStdId());//ID-DESC
 //Collections.sort(list, (o1,o2)-
 >o1.getStdName().compareTo(o2.getStdName()));//Name-ASC
 Collections.sort(list, (o1,o2)-
 >o2.getStdName().compareTo(o1.getStdName()));//Name-DESC
 return list;
}

}

Output:
[
 {
 "stdId": 3,
 "stdName": "sam",
 "stdCourse": "SPRING",
 "stdFee": 5000,
 "stdDiscount": 500
```

```
},
{
 "stdId": 2,
 "stdName": "raju",
 "stdSource": "hibernate",
 "stdFee": 4000,
 "stdDiscount": 400
},
{
 "stdId": 4,
 "stdName": "Sankar158",
 "stdSource": "MicroServices",
 "stdFee": 6000,
 "stdDiscount": 600
},
{
 "stdId": 1,
 "stdName": "Sankar",
 "stdSource": "RESTFUL",
 "stdFee": 3000,
 "stdDiscount": 300
}
]
```

---

## Functional interface

RT method(Params)

Lambda Syntax:

---

(Params)->{ method body};

---

```
public interface Comparator<Student>{
 int compare(Student o1, Student o2);
}
```

Lambda Exp:

```
(o1,o2)->o1.getStdId()-o2.getStdId();
(o1,o2)->o1.getStdName().compareTo(o2.getStdName()));
```

---

Delete one Student from DataBase:

Dao Interface:

```
package in.nit.dao;
```

```
import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentDao {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
}
```

Dao impl:

```
package in.nit.dao;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
import in.nit.db.DBConn;
import in.nit.model.Student;

public class StudentDaoImpl implements IStudentDao {
 //1.Remove Data From Database based one ID
 @Override
 public Integer saveStudent(Student s) {
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?)";
 int count=0;
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,s.getStdId());
 ps.setString(2,s.getStdName());
 ps.setString(3,s.getStdCourse());
 ps.setDouble(4,s.getStdFee());
 ps.setDouble(5,s.getStdDiscount());
 count=ps.executeUpdate();
 } catch (Exception e) {
 e.printStackTrace();
 }
 return count;
 }

 @Override
 public List<Student> getAllStudents() {
```

```
String query="SELECT * FROM STUDENTS";

List<Student> list=null;

try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ResultSet rs=ps.executeQuery();
 list=new ArrayList<>();

 /* while(rs.next()) {
 list.add(new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount"));
 }*/
 //a.Read Data From ResultSet Row
 while(rs.next()) {
 Integer stdId=rs.getInt("sid");
 String stdName=rs.getString("sname");
 String stdCourse=rs.getString("scourse");
 Double stdFee=rs.getDouble("sfee");
 Double stdDiscount=rs.getDouble("sdiscount");

 //b. Convert Data into One Student Class Object
 Student s=new Student();
 s.setStdId(stdId);
 s.setStdName(stdName);
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
s.setStdCourse(stdCourse);
s.setStdFee(stdFee);
s.setStdDiscount(stdDiscount);

//c.Add Student class Object List
list.add(s);

}

}

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

return list;
}

@Override
public boolean removeOneStudent(Integer id) {
 String query="DELETE FROM STUDENTS WHERE SID=?";
 boolean deleted=false;
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,id);
 deleted=ps.executeUpdate()>0?true:false;
 }
 catch(Exception e) {
 deleted=false;
 e.printStackTrace();
 }
}
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
 }
 return deleted;
}

}
```

Service Interface:

```
package in.nit.service;
```

```
import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentService {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
}
```

Service Impl:

```
package in.nit.service.impl;
```

```
import java.util.Collections;
```

```
import java.util.List;
```

```
import javax.inject.Inject;
```

```
import in.nit.dao.IStudentDao;
```

```
import in.nit.model.Student;

import in.nit.service.IStudentService;

public class StudentServiceImpl implements IStudentService {
 @Inject
 private IStudentDao dao;//HAS-A

 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
 //Calling Data Access layer(DAL)

 return dao.saveStudent(s);
 }

 @Override
 public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getStdId()-
 o2.getStdId());//ID-ASC
 //Collections.sort(list, (o1,o2)->o2.getStdId()-
 o1.getStdId());//ID-DESC
 //Collections.sort(list, (o1,o2)-
 >o1.getStdName().compareTo(o2.getStdName()));//Name-ASC
 Collections.sort(list, (o1,o2)-
 >o2.getStdName().compareTo(o1.getStdName()));//Name-DESC
```

```
 return list;
}

@Override
public boolean removeOneStudent(Integer id) {

 return dao.removeOneStudent(id);
}

}
```

RestController:

```
package in.nit.controller;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject//gets impl class Object at runtime
 private IStudentService service;//HAS-A

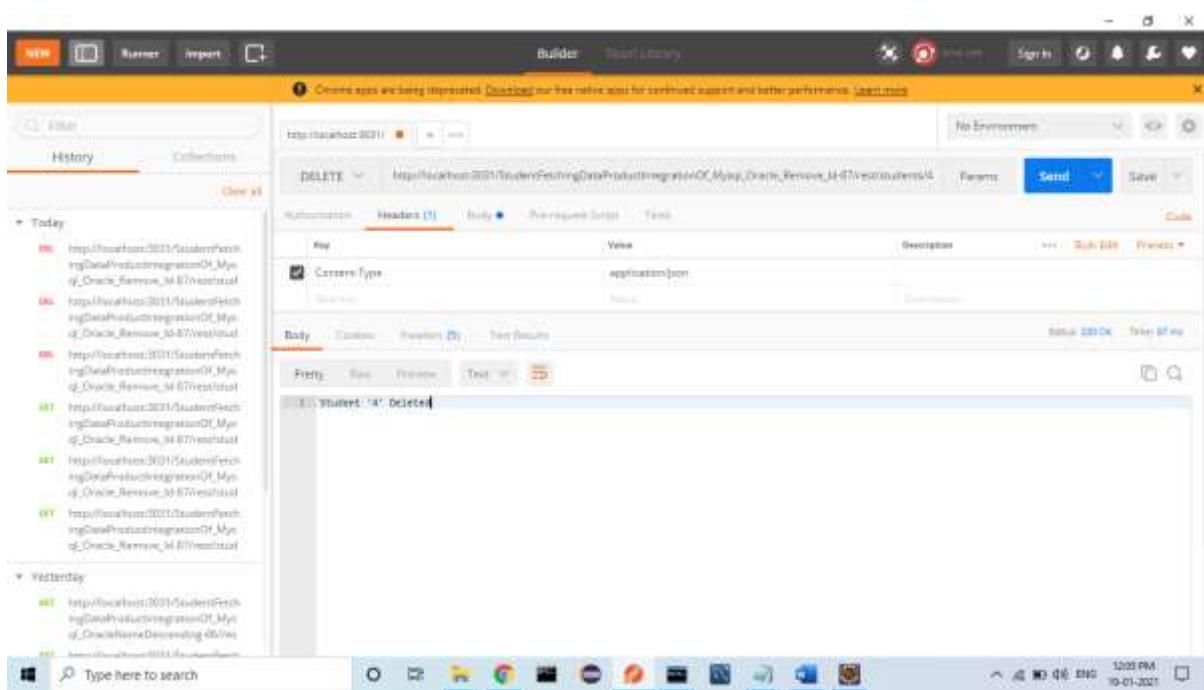
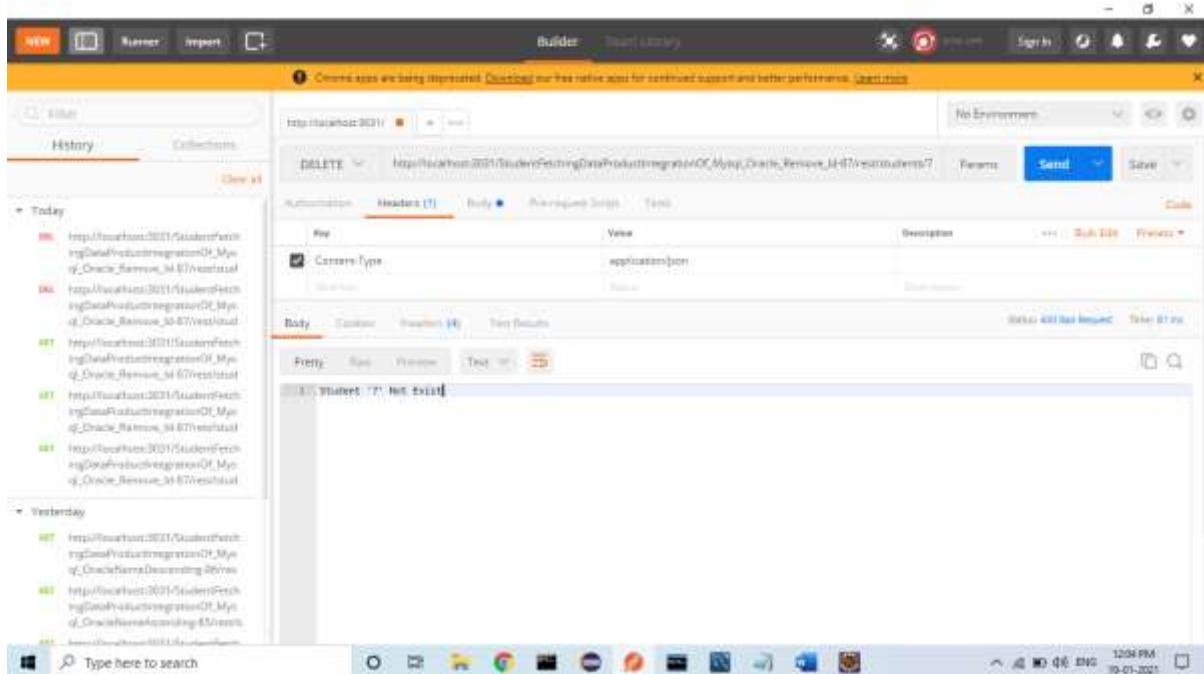
 //Save Student
```

```
@POST
@Consumes("application/json")
public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED)//201-
Success
 .entity("Student Saved!")
 .build();
 }
 else {
 resp=Response.status(Status.BAD_REQUEST)//400-
Wrong Input
 .entity("Student Not Saved!")
 .build();
 }
 }
 catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 .entity("Student Not Saved!...Error")
 .build();
 e.printStackTrace();
 }
 return resp;
}
//Fetch Student
@GET
@Produces("application/json")
public Response getAllStudents(Student student) {
 Response resp=null;
 try {
 List<Student> list=service.getAllStudents();
 resp=Response
 .status(Status.OK)
 .entity(list)
 .build();
 }
 catch(Exception e) {
```

```
resp=Response.status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable to Fetch Data....")
 .build();
 e.printStackTrace();
 }
 return resp;
}
@DELETE
@Path("/{id}")//Dynamic path
public Response deleteOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 boolean status=service.removeOneStudent(id);
 if(status) {
 resp=Response
 .status(Status.OK)
 .entity("Student "+id+" Deleted")
 .build();
 }
 else {
 resp=Response
 .status(Status.BAD_REQUEST)
 .entity("Student "+id+" Not Exist")
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity(" Unable to Delete Data!!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
}
```

-----postman screen-----



Get One Row Data From Database Based One ID

DAO Interface:

package in.nit.dao;

```
import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentDao {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
}
```

DAO IMPL:

```
package in.nit.dao;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
import in.nit.db.DBConn;
import in.nit.model.Student;
```

```
public class StudentDaolmpl implements IStudentDao {
 //1.Get One Row Data From Database based one ID
 @Override
 public Integer saveStudent(Student s) {
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?)";
 int count=0;
 try {
```

```
PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);

 ps.setInt(1,s.getStdId());
 ps.setString(2,s.getStdName());
 ps.setString(3,s.getStdCourse());
 ps.setDouble(4,s.getStdFee());
 ps.setDouble(5,s.getStdDiscount());
 count=ps.executeUpdate();

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

return count;
}
```

```
@Override
public List<Student> getAllStudents() {
 String query="SELECT * FROM STUDENTS";
 List<Student> list=null;
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ResultSet rs=ps.executeQuery();
 list=new ArrayList<>();

 /* while(rs.next()) {
 list.add(new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount")));
 }*/
//a.Read Data From ResultSet Row
while(rs.next()) {
 Integer stdId=rs.getInt("sid");
 String stdName=rs.getString("sname");
 String stdCourse=rs.getString("scourse");
 Double stdFee=rs.getDouble("sfee");
 Double stdDiscount=rs.getDouble("sdiscount");

 //b. Convert Data into One Student Class Object
 Student s=new Student();
 s.setStdId(stdId);
 s.setStdName(stdName);
 s.setStdCourse(stdCourse);
 s.setStdFee(stdFee);
 s.setStdDiscount(stdDiscount);

 //c.Add Student class Object List
 list.add(s);
}
}
catch(Exception e) {
 e.printStackTrace();
 e.getMessage();
```

```
 }

 return list;
}

@Override
public boolean removeOneStudent(Integer id) {
 String query="DELETE FROM STUDENTS WHERE SID=?";
 boolean deleted=false;
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,id);
 deleted=ps.executeUpdate()>0?true:false;
 }
 catch(Exception e) {
 deleted=false;
 e.printStackTrace();
 }
 return deleted;
}

@Override
public Student getOneStudent(Integer id) {
 Student s=null;
 String query="SELECT * FROM STUDENTS WHERE
SID=?";
 try {
```

```
PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);

 ps.setInt(1,id);

 ResultSet rs=ps.executeQuery();

 if(rs.next()) {

 s=new Student(

 rs.getInt("sid"),

 rs.getString("sname"),

 rs.getString("scourse"),

 rs.getDouble("sfee"),

 rs.getDouble("sdiscount")
);

 }

 }

 catch(Exception e) {

 e.printStackTrace();

 }

 return s;
}

}
```

Service Interface:

```
package in.nit.service;

import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentService {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
}
```

In Service Impl:

```
package in.nit.service.impl;
```

```
import java.util.Collections;
```

```
import java.util.List;
```

```
import javax.inject.Inject;
```

```
import in.nit.dao.IStudentDao;
```

```
import in.nit.model.Student;
```

```
import in.nit.service.IStudentService;
```

```
public class StudentServiceImpl implements IStudentService {
```

```
 @Inject
```

```
 private IStudentDao dao;//HAS-A
```

```
 @Override
```

```
 public Integer saveStudent(Student s) {
```

```
//Calculations

 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
 //Calling Data Access layer(DAL)

 return dao.saveStudent(s);
}

@Override
public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getStdId()-
 o2.getStdId());//ID-ASC
 //Collections.sort(list, (o1,o2)->o2.getStdId()-
 o1.getStdId());//ID-DESC
 //Collections.sort(list, (o1,o2)-
 >o1.getStdName().compareTo(o2.getStdName()));//Name-ASC
 Collections.sort(list, (o1,o2)-
 >o2.getStdName().compareTo(o1.getStdName()));//Name-DESC
 return list;
}

@Override
public boolean removeOneStudent(Integer id) {

 return dao.removeOneStudent(id);
}
```

```
@Override
public Student getOneStudent(Integer id) {
 return dao.getOneStudent(id);
}
}
```

In RestController:

```
package in.nit.controller;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rsPathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject//gets impl class Object at runtime
 private IStudentService service;//HAS-A

 //Save Student
 @POST
 @Consumes("application/json")
 public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED).entity(student).build();
 }
 } catch (Exception e) {
 resp=Response.status(Status.INTERNAL_SERVER_ERROR).entity(e.getMessage()).build();
 }
 return resp;
 }
}
```

```
resp=Response.status(Status.CREATED)//201-
Success
}
else {
 resp=Response.status(Status.BAD_REQUEST)//400-
Wrong Input
}
catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
}
//Fetch Student
@GET
@Produces("application/json")
public Response getAllStudents(Student student) {
 Response resp=null;
 try {
 List<Student> list=service.getAllStudents();
 resp=Response
 .status(Status.OK)
 .entity(list)
 .build();
 }
 catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 }
 return resp;
}
```

```
}

@DELETE
@Path("/{id}")//Dynamic path
public Response deleteOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 boolean status=service.removeOneStudent(id);
 if(status) {
 resp=Response
 .status(Status.OK)
 .entity("Student "+id+" Deleted")
 .build();
 }
 else {
 resp=Response
 .status(Status.BAD_REQUEST)
 .entity("Student "+id+" Not Exist")
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity(" Unable to Delete Data!!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
@GET
@Path("/{id}")
@Produces("application/json")
public Response getOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 Student s=service.getOneStudent(id);
 if(s!=null) {
 resp=Response
 .status(Status.OK)
```

```
 .entity(s)
 .build();
 }
} else {
 resp=Response
 .status(Status.NO_CONTENT)
 .build();
}
} catch (Exception e) {
 resp=Response

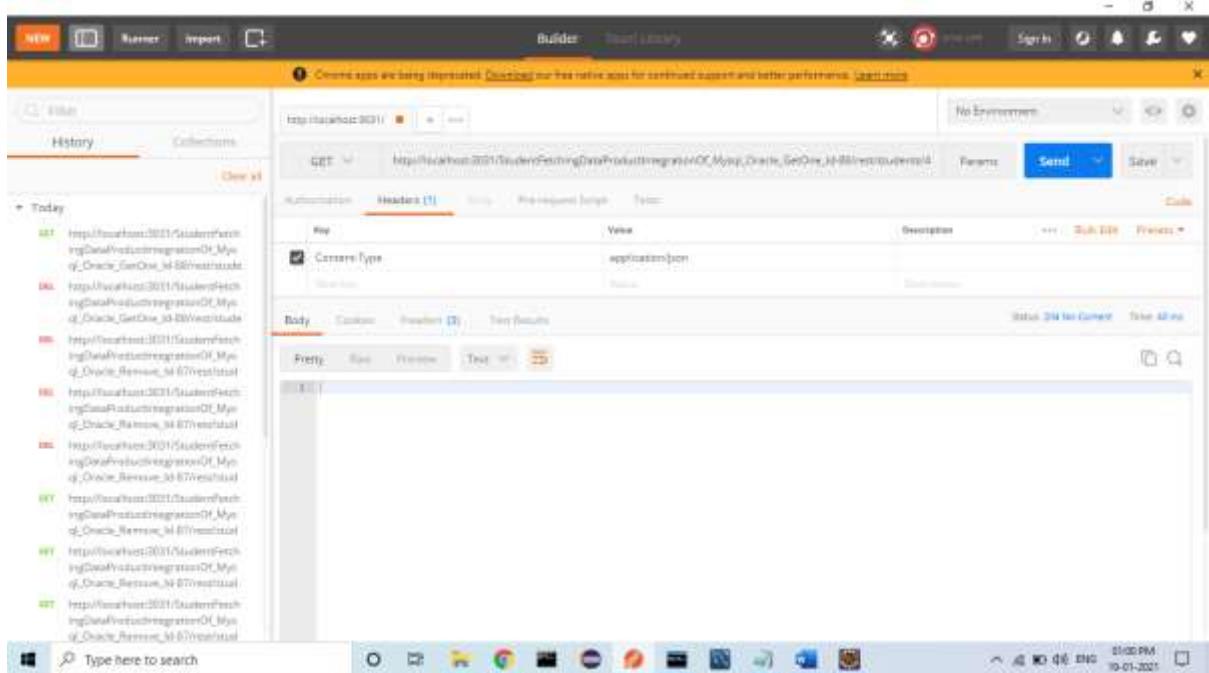
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable To fetch Student Data!!")
 .build();
 e.printStackTrace();
}
return resp;
}
}
```

-----output-----

In Chrome:

[http://localhost:3031/StudentFetchingDataProductIntegrationOf\\_Mysql\\_Oracle\\_GetOne\\_Id-88/rest/students/1](http://localhost:3031/StudentFetchingDataProductIntegrationOf_Mysql_Oracle_GetOne_Id-88/rest/students/1)

```
{"stdId":1,"stdName":"Sankar","stdSource":"RESTFUL","stdFee":3000.0,"stdDiscount":300.0}
```



## Update Data On Database Based On ID

DAO Interface:

```
package in.nit.dao;
```

```
import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentDao {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
 public boolean updateStudent(Student s);
}
```

}

DAO IMPL:

```
package in.nit.dao;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
import com.mysql.cj.xdevapi.DbDoc;
```

```
import in.nit.db.DBConn;
import in.nit.model.Student;
```

```
public class StudentDaoImpl implements IStudentDao {
```

```
 //1. Update Data on Database based on ID
```

```
 @Override
```

```
 public Integer saveStudent(Student s) {
```

```
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?)";
```

```
 int count=0;
```

```
 try {
```

```
 PreparedStatement
```

```
 ps=DBConn.getConnection().prepareStatement(query);
```

```
 ps.setInt(1,s.getStdId());
```

```
 ps.setString(2,s.getStdName());
```

```
 ps.setString(3,s.getStdCourse());
```

```
 ps.setDouble(4,s.getStdFee());
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
ps.setDouble(5,s.getStdDiscount());
count=ps.executeUpdate();
} catch (Exception e) {
 e.printStackTrace();
}
return count;
}

@Override
public List<Student> getAllStudents() {
 String query="SELECT * FROM STUDENTS";
 List<Student> list=null;
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ResultSet rs=ps.executeQuery();
 list=new ArrayList<>();

 /* while(rs.next()) {
 list.add(new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount"));
 }*/
 //a.Read Data From ResultSet Row
 while(rs.next()) {
 }
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
Integer stdId=rs.getInt("sid");
String stdName=rs.getString("sname");
String stdCourse=rs.getString("scourse");
Double stdFee=rs.getDouble("sfee");
Double stdDiscount=rs.getDouble("sdiscount");

//b. Convert Data into One Student Class Object
Student s=new Student();
s.setStdId(stdId);
s.setStdName(stdName);
s.setStdCourse(stdCourse);
s.setStdFee(stdFee);
s.setStdDiscount(stdDiscount);

//c.Add Student class Object List
list.add(s);

}

}

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

return list;
}

@Override
public boolean removeOneStudent(Integer id) {
```

```
String query="DELETE FROM STUDENTS WHERE SID=?";
boolean deleted=false;
try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,id);
 deleted=ps.executeUpdate()>0?true:false;
}
catch(Exception e) {
 deleted=false;
 e.printStackTrace();
}
return deleted;
}
```

```
@Override
public Student getOneStudent(Integer id) {
 Student s=null;
 String query="SELECT * FROM STUDENTS WHERE
 SID=?";
 try {
 PreparedStatement
 ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,id);
 ResultSet rs=ps.executeQuery();
 if(rs.next()) {
 s=new Student(
 rs.getInt("sid"),
```

```
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount")
);
}

}

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

return s;
}
```

```
@Override
public boolean updateStudent(Student s) {
 boolean status=false;
 String query="UPDATE STUDENTS SET
SNAME=?,SCOURSE=?,SFEE=?,SDISCOUNT=? WHERE SID=?";
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setString(1,s.getStdName());
 ps.setString(2, s.getStdCourse());
 ps.setDouble(3,s.getStdFee());
 ps.setDouble(4,s.getStdDiscount());
 ps.setInt(5, s.getStdId());
 status=ps.executeUpdate()>0?true:false;
 } catch (Exception e) {
```

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

```
 status=false;
 e.printStackTrace();
 }
 return status;
}

}
```

Service Interfcae:

```
package in.nit.service;
```

```
import java.util.List;
```

```
import in.nit.model.Student;
```

```
public interface IStudentService {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
 public boolean updateStudent(Student s);
}
```

Service IMPL:

```
package in.nit.service.impl;
```

```
import java.util.Collections;
import java.util.List;
```

```
import javax.inject.Inject;

import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;

public class StudentServiceImpl implements IStudentService {

 @Inject
 private IStudentDao dao;//HAS-A

 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
 //Calling Data Access layer(DAL)

 return dao.saveStudent(s);
 }

 @Override
 public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getId()-o2.getId());//ID-ASC
 }
}
```

```
//Collections.sort(list, (o1,o2)->o2.getStdId()-
o1.getStdId());//ID-DESC

 //Collections.sort(list, (o1,o2)-
>o1.getStdName().compareTo(o2.getStdName()));//Name-ASC

 Collections.sort(list, (o1,o2)-
>o2.getStdName().compareTo(o1.getStdName()));//Name-DESC

 return list;
}

@Override
public boolean removeOneStudent(Integer id) {

 return dao.removeOneStudent(id);
}

@Override
public Student getOneStudent(Integer id) {
 return dao.getOneStudent(id);
}

@Override
public boolean updateStudent(Student s) {
 return dao.updateStudent(s);
}

}
}
```

REST Controller:

**package** in.nit.controller;

```
import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject//gets impl class Object at runtime
 private IStudentService service;//HAS-A

 //Save Student
 @POST
 @Consumes("application/json")
 public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED)//201-
Success
 .entity("Student Saved!")
 .build();
 }
 else {
 resp=Response.status(Status.BAD_REQUEST)//400-
Wrong Input
 .entity("Student Not Saved!")
 .build();
 }
 }
 }
}
```

```
 }
 catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 .entity("Student Not Saved!...Error")
 .build();
 e.printStackTrace();
 }
 return resp;
}
//Fetch Student
@GET
@Produces("application/json")
public Response getAllStudents(Student student) {
 Response resp=null;
 try {
 List<Student> list=service.getAllStudents();
 resp=Response
 .status(Status.OK)
 .entity(list)
 .build();
 }
 catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 .entity("Unable to Fetch Data....")
 .build();
 e.printStackTrace();
 }
 return resp;
}
@DELETE
@Path("/{id}")//Dynamic path
public Response deleteOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 boolean status=service.removeOneStudent(id);
 if(status) {
 resp=Response
```

```
.status(Status.OK)
.entity("Student "+id+" Deleted")
.build();
}
else {
 resp=Response
 .status(Status.BAD_REQUEST)
 .entity("Student "+id+" Not Exist")
 .build();
 }
} catch (Exception e) {
 resp=Response

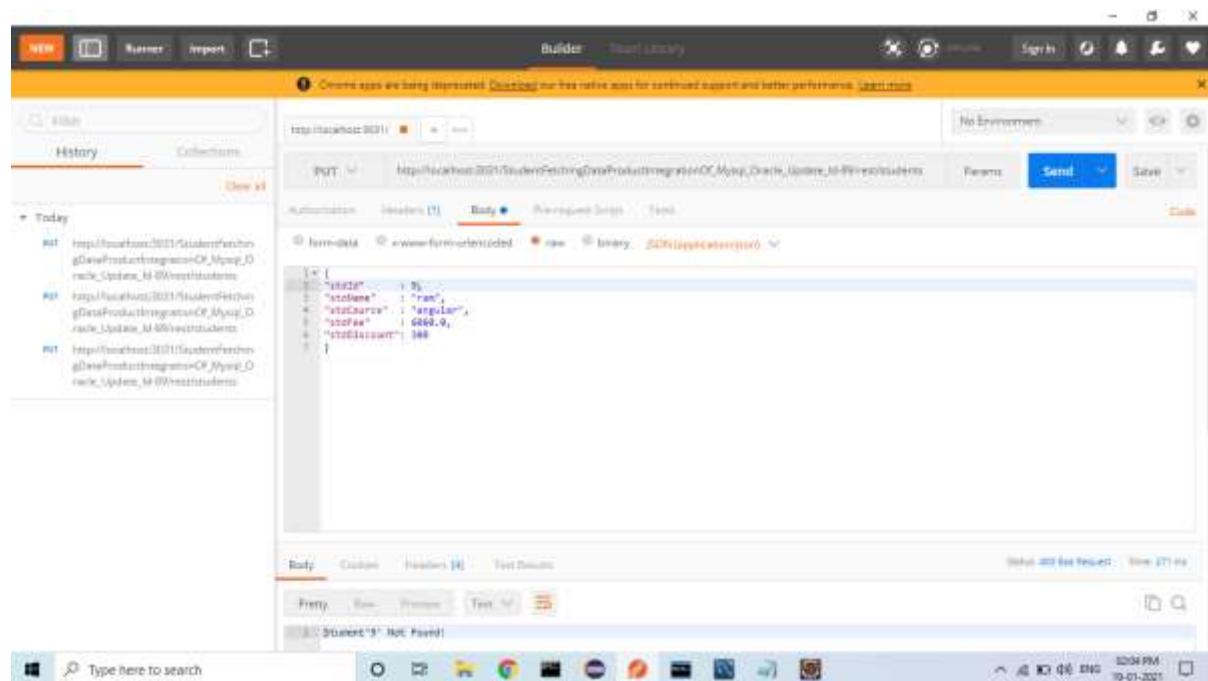
.status(Status.INTERNAL_SERVER_ERROR)//500-Exception
.entity(" Unable to Delete Data!!")
.build();
 e.printStackTrace();
}
return resp;
}
@GET
@Path("/{id}")
@Produces("application/json")
public Response getOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 Student s=service.getOneStudent(id);
 if(s!=null) {
 resp=Response
 .status(Status.OK)
 .entity(s)
 .build();
 }
 else {
 resp=Response
 .status(Status.NO_CONTENT)
 .build();
 }
 } catch (Exception e) {
 resp=Response
 }
}
```

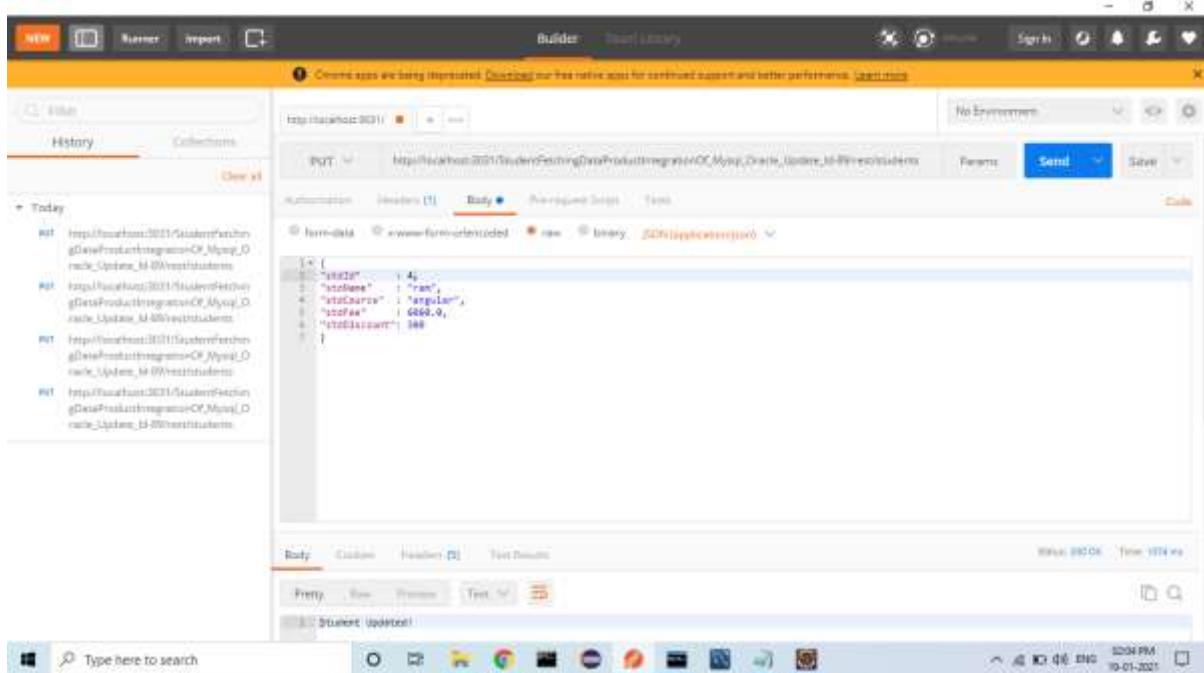
```
.status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable To fetch Student Data!")
 .build();
 e.printStackTrace();
}
return resp;
}
@PUT
@Consumes("application/json")
//@Produces(MediaType.TEXT_PLAIN)
//@Produces("text/plain")
public Response updateOneStudent(
 Student s
) {
Response resp=null;
try {
 boolean status=service.updateStudent(s);
 if(status) {
 resp=Response
 .status(Status.OK)//200
 .entity("Student Updated!")
 .build();
 }
 else {
 resp=Response
 .status(Status.BAD_REQUEST)//400
 .entity("Student"+s.getId()+" Not
Found!")
 .build();
 }
} catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable To Update Student Data!")
 .build();
 e.printStackTrace();
}
return resp;
}
```

## Mysql CommandLine:

```
mysql> select * from students;
+----+-----+-----+-----+-----+
| sid | sname | scourse | sfee | sdiscount |
+----+-----+-----+-----+-----+
| 1 | Sankar | RESTFUL | 3000 | 300 |
| 2 | raju | hibernate | 4000 | 400 |
| 3 | sam | SPRING | 5000 | 500 |
| 4 | king | microservices | 3500 | 300 |
+----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from students;
+----+-----+-----+-----+-----+
| sid | sname | scourse | sfee | sdiscount |
+----+-----+-----+-----+-----+
| 1 | Sankar | RESTFUL | 3000 | 300 |
| 2 | raju | hibernate | 4000 | 400 |
| 3 | sam | SPRING | 5000 | 500 |
| 4 | ram | angular | 6060 | 300 |
+----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```





## Modified Rest Controller:

```

package in.nit.controller;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {

```

@Inject//gets impl class Object at runtime  
**private** IStudentService **service**//HAS-A

```

//Save Student
@POST
@Consumes("application/json")
public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED)//201-
Success
 .entity("Student Saved!")
 .build();
 }
 else {
 resp=Response.status(Status.BAD_REQUEST)//400-
Wrong Input
 .entity("Student Not Saved!")
 .build();
 }
 }
 catch(Exception e) {
 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//50
0-Exception
 .entity("Student Not Saved!...Error")
 .build();
 e.printStackTrace();
 }
 return resp;
}
//Fetch Student
@GET
@Produces("application/json")
public Response getAllStudents(Student student) {
 Response resp=null;
 try {
 List<Student> list=service.getAllStudents();
 resp=Response
 .status(Status.OK)

```

```
 .entity(list)
 .build();
 }
 catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable to Fetch Data....")
 .build();
 e.printStackTrace();
 }
 return resp;
}
@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)
public Response deleteOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 boolean status=service.removeOneStudent(id);
 if(status) {
 resp=Response
 .status(Status.OK)
 .entity("Student "+id+" Deleted")
 .build();
 }
 else {
 resp=Response
 .status(Status.BAD_REQUEST)
 .entity("Student "+id+" Not Exist")
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity(" Unable to Delete Data!!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
```

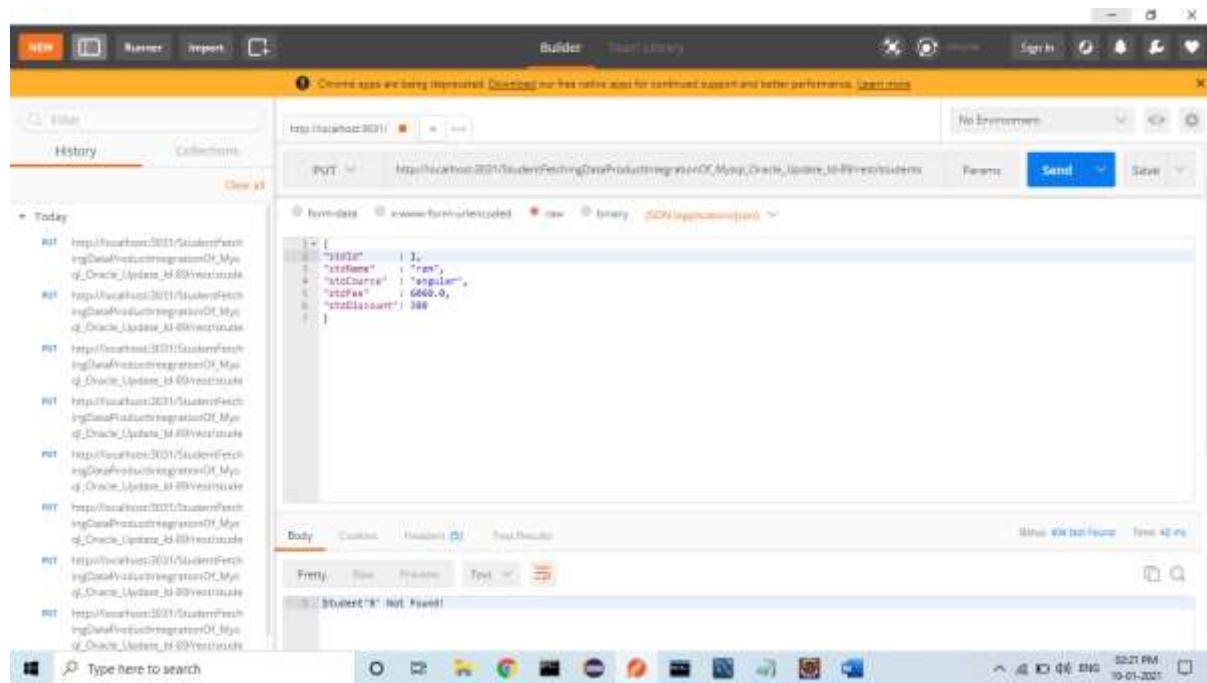
```
@GET
@Path("/{id}")
@Produces("application/json")
public Response getOneStudent(
 @PathParam("id")Integer id
){
 Response resp=null;
 try {
 Student s=service.getOneStudent(id);
 if(s!=null) {
 resp=Response
 .status(Status.OK)
 .entity(s)
 .build();
 }
 else {
 resp=Response
 .status(Status.NO_CONTENT)
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable To fetch Student Data!!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
@PUT
@Consumes("application/json")
//@Produces(MediaType.TEXT_PLAIN)
//@Produces("text/plain")
public Response updateOneStudent(
 Student s
){
 Response resp=null;
 try {
 boolean status=service.updateStudent(s);
 if(status) {
 resp=Response
 .status(Status.OK)//200
```

```

 .status(Status.RESET_CONTENT)//205
 .entity("Student Updated!")
 .build();
 }
 else {
 resp=Response
 // .status(Status.BAD_REQUEST)//400
 .status(Status.NOT_FOUND)//404
 .entity("Student"+s.getStdId()+" Not
Found!")
 .build();
 }
} catch (Exception e) {
 resp=Response

 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable To Update Student Data!!")
 .build();
 e.printStackTrace();
}
return resp;
}
}

```



-----CURD Operations On RestAPI-----

(Save,Update,Get,Delete)

1. In pom.xml File:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/mysql/mysql-
 connector-java -->
 <dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.22</version>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.16</version>
 <scope>provided</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
 <dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
 </dependency>

 </dependencies>
```

## 2. In Model class:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Student {
 private Integer stdId;
 private String stdName;
 private String stdCourse;
 private Double stdFee;
```

```
 private Double stdDiscount;
}
```

3. In DbConn:

```
package in.nit.db;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConn {
 private static Connection conn=null;

 static {
 try {
 Class.forName("com.mysql.jdbc.Driver");

 conn=DriverManager.getConnection("jdbc:mysql://localhost:
3306/webservices","root","root");
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 public static Connection getConnection() {

 return conn;
 }
}
```

4. In Dao :

```
package in.nit.dao;

import java.util.List;

import in.nit.model.Student;

public interface IStudentDao {
 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
 public boolean updateStudent(Student s);
}
```

5. dao impl:

```
package in.nit.dao;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import com.mysql.cj.xdevapi.DbDoc;

import in.nit.db.DBConn;
import in.nit.model.Student;

public class StudentDaolmpl implements IStudentDao {
 //1. Update Data on Database based on ID
 @Override
 public Integer saveStudent(Student s) {
 String query="INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?)";
 int count=0;
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,s.getStdId());
 ps.setString(2,s.getStdName());
 ps.setString(3,s.getStdCourse());
 ps.setDouble(4,s.getStdFee());
 ps.setDouble(5,s.getStdDiscount());
 count=ps.executeUpdate();
 } catch (Exception e) {
 e.printStackTrace();
 }
 return count;
 }

 @Override
 public List<Student> getAllStudents() {
 String query="SELECT * FROM STUDENTS";
 List<Student> list=null;
```

```
try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ResultSet rs=ps.executeQuery();
 list=new ArrayList<>();

/* while(rs.next()) {
 list.add(new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount")));
}
//a.Read Data From ResultSet Row
while(rs.next()) {
 Integer stdId=rs.getInt("sid");
 String stdName=rs.getString("sname");
 String stdCourse=rs.getString("scourse");
 Double stdFee=rs.getDouble("sfee");
 Double
stdDiscount=rs.getDouble("sdiscount");

//b. Convert Data into One Student Class
Object
 Student s=new Student();
 s.setStdId(stdId);
 s.setStdName(stdName);
 s.setStdCourse(stdCourse);
 s.setStdFee(stdFee);
 s.setStdDiscount(stdDiscount);

//c.Add Student class Object List
 list.add(s);
}
catch(Exception e) {
 e.printStackTrace();
 e.getMessage();
}
```

```
 return list;
 }

 @Override
 public boolean removeOneStudent(Integer id) {
 String query="DELETE FROM STUDENTS WHERE
SID=?";
 boolean deleted=false;
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,id);
 deleted=ps.executeUpdate()>0?true:false;
 }
 catch(Exception e) {
 deleted=false;
 e.printStackTrace();
 }
 return deleted;
 }

 @Override
 public Student getOneStudent(Integer id) {
 Student s=null;
 String query="SELECT * FROM STUDENTS WHERE
SID=?";
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setInt(1,id);
 ResultSet rs=ps.executeQuery();
 if(rs.next()) {
 s=new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount")
);
 }
 }
```

```
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 return s;
}

@Override
public boolean updateStudent(Student s) {
 boolean status=false;
 String query="UPDATE STUDENTS SET
SNAME=?,SCOURSE=?,SFEE=?,SDISCOUNT=? WHERE
SID=?";
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(query);
 ps.setString(1,s.getStdName());
 ps.setString(2, s.getStdCourse());
 ps.setDouble(3,s.getStdFee());
 ps.setDouble(4,s.getStdDiscount());
 ps.setInt(5, s.getStdId());
 status=ps.executeUpdate()>0?true:false;
 } catch (Exception e) {
 status=false;
 e.printStackTrace();
 }
 return status;
}

}
```

## 6. In Service:

```
package in.nit.service;

import java.util.List;

import in.nit.model.Student;

public interface IStudentService {
 public Integer saveStudent(Student s);
```

```
public List<Student> getAllStudents();
public boolean removeOneStudent(Integer id);
public Student getOneStudent(Integer id);
public boolean updateStudent(Student s);
}
```

7. In Service Impl:

```
package in.nit.service.impl;

import java.util.Collections;
import java.util.List;

import javax.inject.Inject;

import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;

public class StudentServiceImpl implements IStudentService {
 @Inject
 private IStudentDao dao;//HAS-A

 @Override
 public Integer saveStudent(Student s) {
 //Calculations
 Double fee=s.getStdFee();
 Double dis=fee*10/100.0;
 s.setStdDiscount(dis);
 //Calling Data Access layer(DAL)

 return dao.saveStudent(s);
 }

 @Override
 public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list, (o1,o2)->o1.getStdId()-
 o2.getStdId());//ID-ASC
 //Collections.sort(list, (o1,o2)->o2.getStdId()-
 o1.getStdId());//ID-DESC
```

```
//Collections.sort(list, (o1,o2)-
>o1.getStdName().compareTo(o2.getStdName()));//Name-ASC
 Collections.sort(list, (o1,o2)-
>o2.getStdName().compareTo(o1.getStdName()));//Name-DESC
 return list;
}

@Override
public boolean removeOneStudent(Integer id) {

 return dao.removeOneStudent(id);
}

@Override
public Student getOneStudent(Integer id) {
 return dao.getOneStudent(id);
}

@Override
public boolean updateStudent(Student s) {
 return dao.updateStudent(s);
}
}
```

#### 8. AppConfig:

```
package in.nit.config;
import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IStudentDao;
import in.nit.dao.StudentDaoImpl;
import in.nit.service.IStudentService;
import in.nit.service.impl.StudentServiceImpl;
@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 //this.packages("in.nit");//nit-param
```

```
//this.register(new AbstractBinder() {
//or
 packages("in.nit");//nit-param
 register(new AbstractBinder() {
 @Override
 public void configure() {

 bind(StudentDaoImpl.class).to(IStudentDao.class);

 bind(StudentServiceImpl.class).to(IStudentService.class);
 }
 });
}
}
9. RestController:
package in.nit.controller;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject//gets impl class Object at runtime
 private IStudentService service;//HAS-A
```

```
//Save Student
@POST
@Consumes("application/json")
public Response saveStudent(Student student) {
 Response resp=null;
 try {
 //Call Service Layer(SL) to Save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp=Response.status(Status.CREATED)//201-
Success
 .entity("Student Saved!")
 .build();
 }
 else {
 resp=Response.status(Status.BAD_REQUEST)//400-Wrong
Input
 .entity("Student Not Saved!")
 .build();
 }
 }
 catch(Exception e) {
 resp=Response.status(Status.INTERNAL_SERVER_ERRO
R)//500-Exception
 .entity("Student Not Saved!...Error")
 .build();
 e.printStackTrace();
 }
 return resp;
}
//Fetch Student
@GET
@Produces("application/json")
public Response getAllStudents(Student student) {
 Response resp=null;
 try {
 List<Student> list=service.getAllStudents();
 resp=Response
```

```
 .status(Status.OK)
 .entity(list)
 .build();
 }
 catch(Exception e) {

 resp=Response.status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity("Unable to Fetch Data....")
 .build();
 e.printStackTrace();
 }
 return resp;
}

@DELETE
@Path("/{id}")//Dynamic path
public Response deleteOneStudent(
 @PathParam("id")Integer id
) {
 Response resp=null;
 try {
 boolean status=service.removeOneStudent(id);
 if(status) {
 resp=Response
 .status(Status.OK)
 .entity("Student "+id+" Deleted")
 .build();
 }
 else {
 resp=Response
 .status(Status.BAD_REQUEST)
 .entity("Student "+id+" Not Exist")
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-Exception
 .entity(" Unable to Delete Data!!")
 }
}
```

```
 .build();
 e.printStackTrace();
 }
 return resp;
}
@GetMapping
@Path("/{id}")
@Produces("application/json")
public Response getOneStudent(
 @PathParam("id") Integer id
) {
 Response resp=null;
 try {
 Student s=service.getOneStudent(id);
 if(s!=null) {
 resp=Response
 .status(Status.OK)
 .entity(s)
 .build();
 }
 else {
 resp=Response
 .status(Status.NO_CONTENT)
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-
 .entity("Unable To fetch Student Data!!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
@PUT
@Consumes("application/json")
//@Produces(MediaType.TEXT_PLAIN)
//@Produces("text/plain")
```

```
public Response updateOneStudent(
 Student s
) {
 Response resp=null;
 try {
 boolean status=service.updateStudent(s);
 if(status) {
 resp=Response
 //.status(Status.OK)//200
 .status(Status.RESET_CONTENT)//205
 .entity("Student Updated!")
 .build();
 }
 else {
 resp=Response
 //.status(Status.BAD_REQUEST)//400
 .status(Status.NOT_FOUND)//404
 .entity("Student"+s.getId()+" Not
Found!")
 .build();
 }
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)//500-
Exception
 .entity("Unable To Update Student Data!!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
```

-----outputs you can check with each Operation-----

-----CRUD Operations On Oracle Database-----

[ It is web App]

1. In pom.xml file:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
 <!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
 <dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
 </dependency>

 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>5.1.46</version>
 </dependency>
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.12</version>
 <scope>provided</scope>
 </dependency>
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
```

```
<artifactId>jersey-media-json-jackson</artifactId>
<version>2.30</version>
</dependency>
</dependencies>
```

## 2. In Model class:

```
package in.nit.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Student {

 private Integer stdId;
 private String stdName;
 private String stdCourse;
 private Double stdFee;
 private Double stdDiscount;

}

/**
create table students (
 sid int,sname varchar(25),
 scourse varchar(15),
 sfee double, sdiscount double);
**/
```

## 3. In DbConn

```
package in.nit.db;

import java.sql.Connection;
import java.sql.DriverManager;

public class DbConn {
```

```
private static Connection conn=null;

static {
 try {
 Class.forName("oracle.jdbc.driver.OracleDriver");
 conn=DriverManager.getConnection(
 "jdbc:oracle:thin:@localhost:1521:ORCL",
 "scott", "tiger");
 } catch (Exception e) {
 e.printStackTrace();
 }
}

public static Connection getConn() {
 return conn;
}
}
```

#### 4. In Dao Interface:

```
package in.nit.dao;

import java.util.List;

import in.nit.model.Student;

public interface IStudentDao {

 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
 public boolean updateStudent(Student s);

}
```

#### 5. In Dao IMPL:

```
package in.nit.dao.impl;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import in.nit.dao.IStudentDao;
import in.nit.db.DbConn;
import in.nit.model.Student;

public class StudentDaolmpl implements IStudentDao {

 @Override
 public Integer saveStudent(Student s) {
 String sql=" INSERT INTO STUDENTS
VALUES(?,?,?,?,?,?) ";
 int count=0;
 try {
 PreparedStatement
stmt=DbConn.getConn().prepareStatement(sql);

 pstmt.setInt(1, s.getStdId());
 pstmt.setString(2, s.getStdName());
 pstmt.setString(3, s.getStdCourse());
 pstmt.setDouble(4, s.getStdFee());
 pstmt.setDouble(5, s.getStdDiscount());

 count=pstmt.executeUpdate();

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

 @Override
 public List<Student> getAllStudents() {
 String sql=" SELECT * FROM STUDENTS ";
```

```
List<Student> list=null;
try {
 PreparedStatement
 pstmt=DbConn.getConn().prepareStatement(sql);
 ResultSet rs=pstmt.executeQuery();
 list=new ArrayList<>();
 while(rs.next()) {
 list.add(new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),
 rs.getDouble("sdiscount"))
);
 /*
 //a.Read data from ResultSet row
 int stdId=rs.getInt("sid");
 String stdName=rs.getString("sname");
 String stdCourse=rs.getString("scourse");
 double stdFee=rs.getDouble("sfee");
 double
 stdDiscount=rs.getDouble("sdiscount");

 //b.convert data into one Student class
 object
 Student s=new Student();
 s.setStdId(stdId);
 s.setStdName(stdName);
 s.setStdCourse(stdCourse);
 s.setStdFee(stdFee);
 s.setStdDiscount(stdDiscount);

 //c. add student class object to list
 list.add(s);
 */
 }
}

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

```
 return list;
 }

 @Override
 public boolean removeOneStudent(Integer id) {
 String sql=" DELETE FROM STUDENTS WHERE SID
= ? ";
 boolean deleted=false;
 try {

 PreparedStatement
 pstmt=DbConn.getConn().prepareStatement(sql);
 pstmt.setInt(1, id);

 deleted=pstmt.executeUpdate()>0?true:false;
 } catch (Exception e) {
 deleted=false;
 e.printStackTrace();
 }
 return deleted;
 }

 @Override
 public Student getOneStudent(Integer id) {
 Student s=null;
 String sql=" SELECT * FROM STUDENTS WHERE
SID = ? ";
 try {
 PreparedStatement
 pstmt=DbConn.getConn().prepareStatement(sql);
 pstmt.setInt(1, id);

 ResultSet rs=pstmt.executeQuery();
 if(rs.next()) {
 s=new Student(
 rs.getInt("sid"),
 rs.getString("sname"),
 rs.getString("scourse"),
 rs.getDouble("sfee"),

```

```
 rs.getDouble("sdiscount")
);
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 return s;
}

@Override
public boolean updateStudent(Student s) {
 boolean status=false;
 String sql= " UPDATE STUDENTS "
 + " SET
SNAME=?,SCOURSE=?,SFEE=?,SDISCOUNT=? "
 + " WHERE SID=? ";
 try {

 PreparedStatement
 pstmt=DbConn.getConn().prepareStatement(sql);
 pstmt.setString(1, s.getStdName());
 pstmt.setString(2, s.getStdCourse());
 pstmt.setDouble(3, s.getStdFee());
 pstmt.setDouble(4, s.getStdDiscount());
 pstmt.setInt(5, s.getStdId());

 status=pstmt.executeUpdate()>0?true:false;
 } catch (Exception e) {
 status=false;
 e.printStackTrace();
 }
 return status;
}
}
```

## 6. In Service Interface:

```
package in.nit.service;

import java.util.List;

import in.nit.model.Student;

public interface IStudentService {

 public Integer saveStudent(Student s);
 public List<Student> getAllStudents();
 public boolean removeOneStudent(Integer id);
 public Student getOneStudent(Integer id);
 public boolean updateStudent(Student s);
}
```

## 7. In Service IMPL:

```
package in.nit.service.impl;

import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import javax.inject.Inject;

import in.nit.dao.IStudentDao;
import in.nit.model.Student;
import in.nit.service.IStudentService;

public class StudentServiceImpl implements IStudentService {
 @Inject
```

```
private IStudentDao dao;//HAS-A

@Override
public Integer saveStudent(Student s) {
 //calculations
 double fee=s.getStdFee();
 double disc=fee * 10/100.0;
 s.setStdDiscount(disc);
 //calling DAL
 return dao.saveStudent(s);
}

@Override
public List<Student> getAllStudents() {
 List<Student> list=dao.getAllStudents();
 //Collections.sort(list,(o1,o2)->o1.getStdId()-o2.getStdId()); //ID-ASC
 Collections.sort(list,(o1,o2)->o2.getStdId()-o1.getStdId()); //ID-DESC
 //Collections.sort(list,(o1,o2)->o1.getStdName().compareTo(o2.getStdName())); //NAME-ASC
 //Collections.sort(list,(o1,o2)->o2.getStdName().compareTo(o1.getStdName())); //NAME-DESC

 Collections.sort(list,new Comparator<Student>() {
 public int compare(Student o1, Student o2) {
 return o2.getStdId()-o1.getStdId();
 }
 });
 return list;
}

@Override
public boolean removeOneStudent(Integer id) {
 return dao.removeOneStudent(id);
}
```

```
@Override
public Student getOneStudent(Integer id) {
 return dao.getOneStudent(id);
}

@Override
public boolean updateStudent(Student s) {
 //calculations
 double fee=s.getStdFee();
 double disc=fee * 10/100.0;
 s.setStdDiscount(disc);
 //calling DAL
 return dao.updateStudent(s);
}
}
```

## 8. In App Config:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IStudentDao;
import in.nit.dao.impl.StudentDaolmpl;
import in.nit.service.IStudentService;
import in.nit.service.impl.StudentServiceimpl;

@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {

 public AppConfig() {
 this.packages("in.nit");
 this.register(new AbstractBinder() {
```

```
 @Override
 protected void configure() {

 bind(StudentDaoImpl.class).to(IStudentDao.class);

 bind(StudentServiceImpl.class).to(IStudentService.class);
 }
});
```

## 9. RestController:

```
package in.nit.controller;

import java.util.List;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Student;
import in.nit.service.IStudentService;

@Path("/students")
public class StudentRestController {
 @Inject
 private IStudentService service; //HAS-A
```

```
//1. save data into DB
@POST
@Consumes("application/json")
@Produces("text/plain")
public Response saveStudent(Student student) {
 Response resp=null;

 try {
 //call service layer to save
 Integer count=service.saveStudent(student);
 if(count>0) {
 resp = Response.status(Status.CREATED)
//201 -success
 .entity("Student Saved!")
 .build();
 }else {
 resp =
Response.status(Status.BAD_REQUEST) //400-wrong input
 .entity("Student not saved!")
 .build();
 }
 } catch (Exception e) {
 resp =
Response.status(Status.INTERNAL_SERVER_ERROR)//500-
exception
 .entity("Student not saved!")
 .build();
 e.printStackTrace();
 }

 return resp;
}

//2. Fetching data from Database
@GET
@Produces("application/json")
public Response getAllStudents() {
 Response resp=null;
 try {
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
List<Student> list=service.getAllStudents();
resp = Response
 .status(Status.OK)
 .entity(list)
 .build();

} catch (Exception e) {
 resp =
Response.status(Status.INTERNAL_SERVER_ERROR)//500-
exception
 .entity("Unable to Fetch Data!!")
 .build();

 e.printStackTrace();
}
return resp;
}

//3. Delete based on ID
@DELETE
@Path("/{id}")
@Produces(MediaType.TEXT_PLAIN)
public Response deleteOneStudent(
 @PathParam("id")Integer id)
{
 Response resp=null;
 try {
 boolean status=service.removeOneStudent(id);
 if(status) {
 resp = Response
 .status(Status.OK)
 .entity("Student "+id+
Deleted")
 .build();
 }else {
 resp = Response
 .status(Status.BAD_REQUEST)
 .entity("Student "+id+" Not
exist")
 }
 }
}
```

```
 .build();
 }

} catch (Exception e) {
 resp =
Response.status(Status.INTERNAL_SERVER_ERROR)//500-
exception
 .entity("Unable to Delete Data!!")
 .build();
 e.printStackTrace();
}

return resp;
}

//4. fetch one student by Id
@GET
@Path("/{id}")
@Produces("application/json")
public Response getOneStudent(
 @PathParam("id")Integer id
)
{
 Response resp=null;
 try {

 Student s=service.getOneStudent(id);
 if(s!=null) {
 resp = Response.status(Status.OK)
 .entity(s)
 .build();
 }else {
 resp =
Response.status(Status.NO_CONTENT)
 .build();
 }
 }

} catch (Exception e) {
```

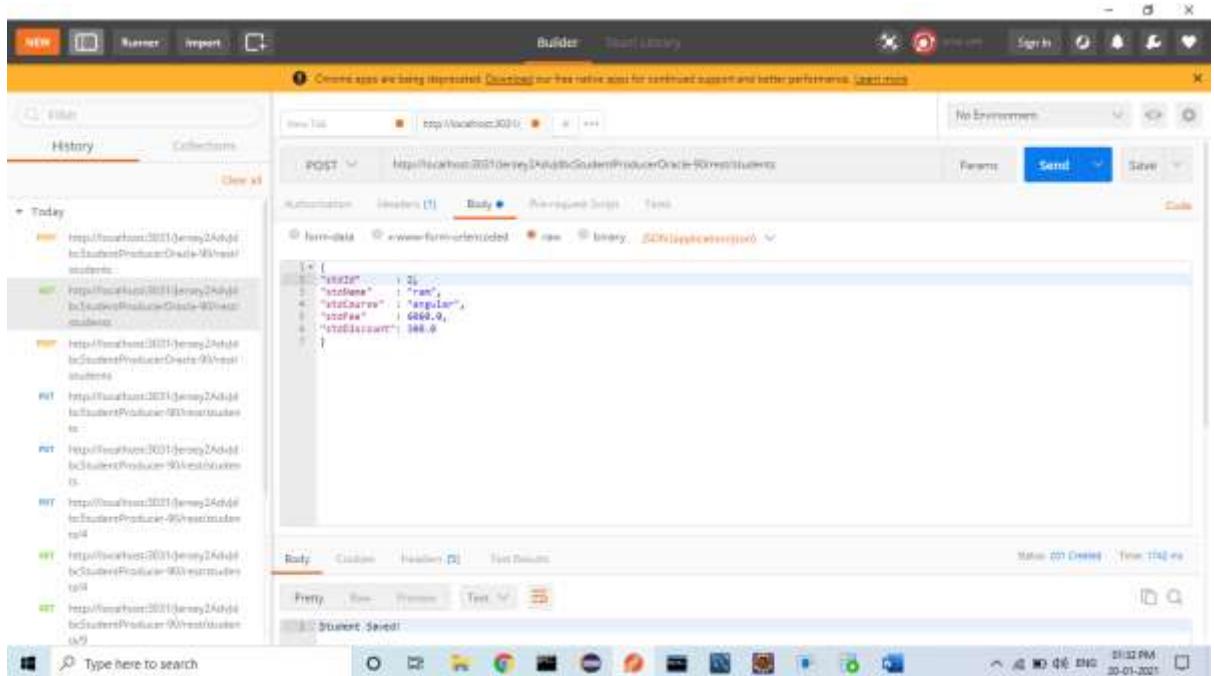
```
 resp =
Response.status(Status.INTERNAL_SERVER_ERROR)//500-
exception
 .entity("Unable to Fetch Student
Data!!")
 .build();
e.printStackTrace();
}
return resp;
}

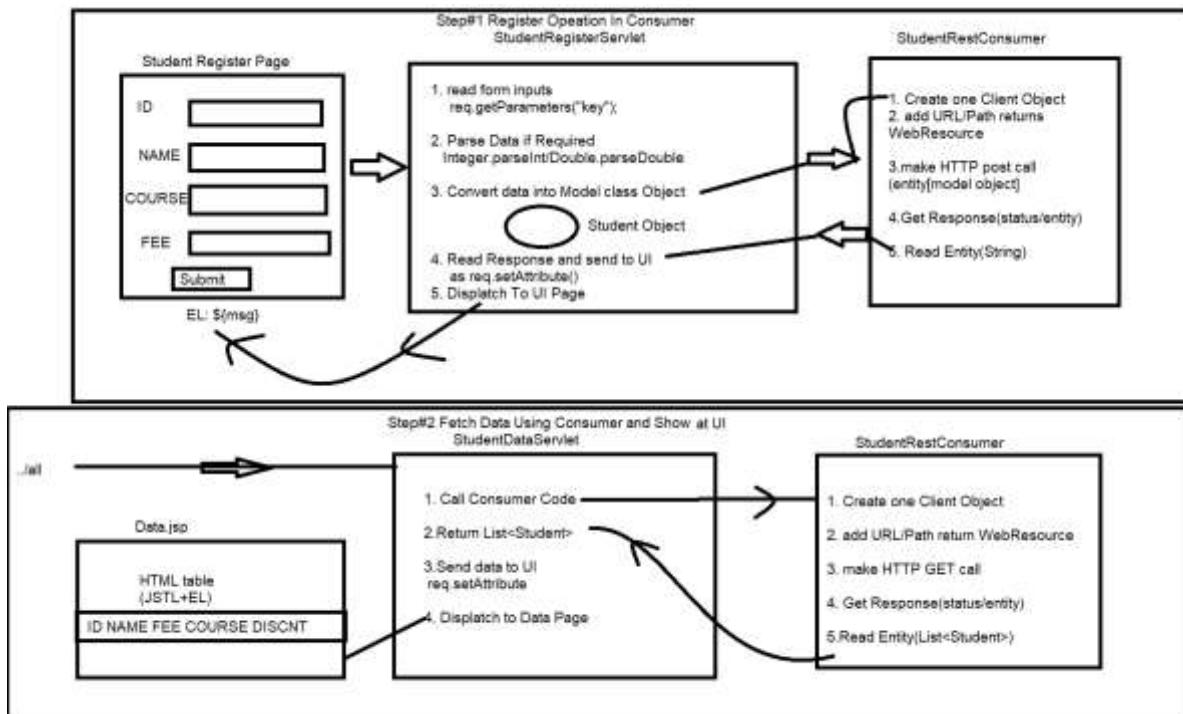
//5. Update one Student based on
@PUT
@Consumes("application/json")
//@Produces(MediaType.TEXT_PLAIN)
@Produces("text/plain")
public Response updateOneStudent(
 Student student)
{
 Response resp=null;
 try {
 boolean status=service.updateStudent(student);
 if(status) {
 resp = Response
 //.status(Status.RESET_CONTENT) //205
 .status(Status.OK) //200
 .entity("Student Updated!")
 .build();
 }else {
 resp = Response
 .status(Status.BAD_REQUEST)
 //.status(Status.NOT_FOUND)
 //400
 //.status(Status.NOT_FOUND)
 //404
 .entity("Student
"+student.getStdId()+" Not Found!")
 .build();
 }
}
```

```
 } catch (Exception e) {
 resp =
Response.status(Status.INTERNAL_SERVER_ERROR)//500-
exception
 .entity("Unable to Update Student
Data!!")
 .build();
 e.printStackTrace();
 }
 }

}
```

-----Check Outputs in postman tool-----  
I will show you insert(Save) Operation Output:





## CURD Operations using Consumer App(Servlets+JSP+JAX-RS Consumer):

Note: 1. Consumer App also is WebApp

- 2.URL is upto class path
- 3.path is upto method path

### Task#1 Consumer Code Insert Operation

#### 1. In Pom.xml File:

```

<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
 <!-- https://mvnrepository.com/artifact/jstl/jstl -->
 <dependency>
 <groupId>jstl</groupId>
 <artifactId>jstl</artifactId>
 <version>1.2</version>
 </dependency>

```

```
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.12</version>
 <scope>provided</scope>
</dependency>
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
</dependency>
</dependencies>
```

## 2.menu.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Menu</title>
</head>
<body>
<div align="center">

<table>
<tr>
<td>Register</td>
</tr>
</table>
</div>
```

```
</body>
</html>
```

### 3.index.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1" isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Index Page</title>
</head>
<body>
<%@include file="menu.jsp" %>
<div align="center">
<h2><marquee>Welcome To Student Registration
Page</marquee></h2>
<form action="insert" method="post">
<pre>
ID :<input type="text" name="stdId">

NAME :<input type="text" name="stdName">

COURSE:<input type="text" name="stdCourse">

FEE :<input type="text" name="stdFee">

<input type="submit" value="Register">

</pre>
${message}
</form>
</div>
</body>
</html>
```

### 4.Model Class:

```
package in.nit.model;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Data

@NoArgsConstructor

@AllArgsConstructor

public class Student {

 private Integer stdId;
 private String stdName;
 private String stdCourse;
 private Double stdFee;
 private Double stdDiscount;

}
```

## 5. RestConsumer Code:

```
package in.nit.consumer;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

import in.nit.model.Student;

public class StudentRestConsumer {
 public static final String
URL="http://localhost:3031/Jersey2AdvJdbcStudentProducerOracl
e-90";
 public static String saveStudent(Student s) {
 String path="rest/students";
```

```
String respMsg=null;
try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.post(Entity.json(s));//Convert JSON and post
 int status=resp.getStatus();
 //System.out.println(status); To get Response in
Console Screen
 respMsg=resp.readEntity(String.class);
}
catch(Exception e) {
 e.printStackTrace();
}
return respMsg;
}
```

## 6.Student Registration Servlet:

```
package in.nit.servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;
@WebServlet("/insert")//Servlets3.x
public class StudentRegisterServlet extends HttpServlet {
```

```
public void doPost(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
 //1. Read Form Data
 String id=req.getParameter("stdId");
 String stdName=req.getParameter("stdName");
 String stdCourse=req.getParameter("stdCourse");
 String fee=req.getParameter("stdFee");

 //2. Parse Data If Required
 Integer stdId=Integer.parseInt(id);
 Double stdFee=Double.parseDouble(fee);

 //3. Convert to Model class Object
 Student std=new Student(stdId, stdName, stdCourse,
 stdFee, 0.0);

 //4. Call Consumer Code and get Response
 String msg=StudentRestConsumer.saveStudent(std);
 //5. Send Message to UI
 req.setAttribute("message",msg);

 //6. Dispatch To UI Back
 RequestDispatcher
rd=req.getRequestDispatcher("index.jsp");
 rd.forward(req, res);
}

}
```

Note: Input Data is given as JSON for List<Student>, which is converted into Collection Object Using GenericType<T>.

Example Code:

```
//which will convert JSON to one Model format
List<Student> list
=resp.readEntity(new GenericType<List<Student>>(){});
```

-----Output In Consumer page-----

Index Page      X      +

AdvJdbcStudentConsumerOracle-90/insert

ig Boot & Micr...    Folder - Google Dri...    Difference between...    Different types of w...    Never Stop Learnin...

 **NARESH  
technologies**  
An ISO 9001 : 2008 Certified Company

**Student Management App**  
**RAGHU SIR STUDNETS**

[Register](#)

## Welcome To Student Registration Page

ID :

NAME :

COURSE:

FEE :

[Register](#)

Student Saved!

```
SQL> set lines 100;
SQL> set pages 100;
SQL> select * from students;

 SID SNAME SCOURSE SFEE SDISCOUNT
----- -----
 1 ram angular 6060 606
 2 ram angular 6060 606
 34 mohan new frame 3400 340

SQL> select * from students;

 SID SNAME SCOURSE SFEE SDISCOUNT
----- -----
 1 ram angular 6060 606
 2 ram angular 6060 606
 34 mohan new frame 3400 340
 35 akhila linux 2000 200

SQL>
```

Type here to search

Windows Start button    Search icon    Taskbar icons: File Explorer, Google Chrome, File Manager, Task View, Taskbar Icons, Taskbar Icons

## Task2: Consumer Code Data Fetch Operations

---

1. In StudentRestConsumer Code add below method:

```
package in.nit.consumer;

import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;

import in.nit.model.Student;

public class StudentRestConsumer {
 public static final String
URL="http://localhost:3031/Jersey2AdvJdbcStudentProducerOracle-90";
 public static String saveStudent(Student s) {
 //Consumer Code For Insert Operation
 String path="rest/students";
 String respMsg=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.post(Entity.json(s));//
Convert JSON and post
 int status=resp.getStatus();
 //System.out.println(status); To get Response in
Console Screen
 respMsg=resp.readEntity(String.class);
 }
 }
}
```

```
 catch(Exception e) {
 e.printStackTrace();
 }
 return respMsg;
 }
 public static List<Student> getAllStudents(){
 String path="/rest/students";
 List<Student> list=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.get();

 //JSON [{} ,{} ,{}] => List<Student>
 list=resp.readEntity(new
GenericType<List<Student>>() {});

 } catch (Exception e) {
 e.printStackTrace();
 }
 return list;
 }
}
```

## 2. Student Data Servlet:

```
package in.nit.servlet;

import java.io.IOException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;
@WebServlet("/all")//Servlets3.x
public class StudentDataServlet extends HttpServlet {
 public void doGet(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
 //1. Call to Consumer Code
 //2. Return List Student
 List<Student> list=StudentRestConsumer.getAllStudents();
 //3.Send Data To UI
 req.setAttribute("list",list);
 //4.Dispatch To UI Page
 RequestDispatcher rd=req.getRequestDispatcher("data.jsp");
 rd.forward(req, res);
 }
}
```

### 3. In menu.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-
1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Menu</title>
</head>
<body>
<div align="center">

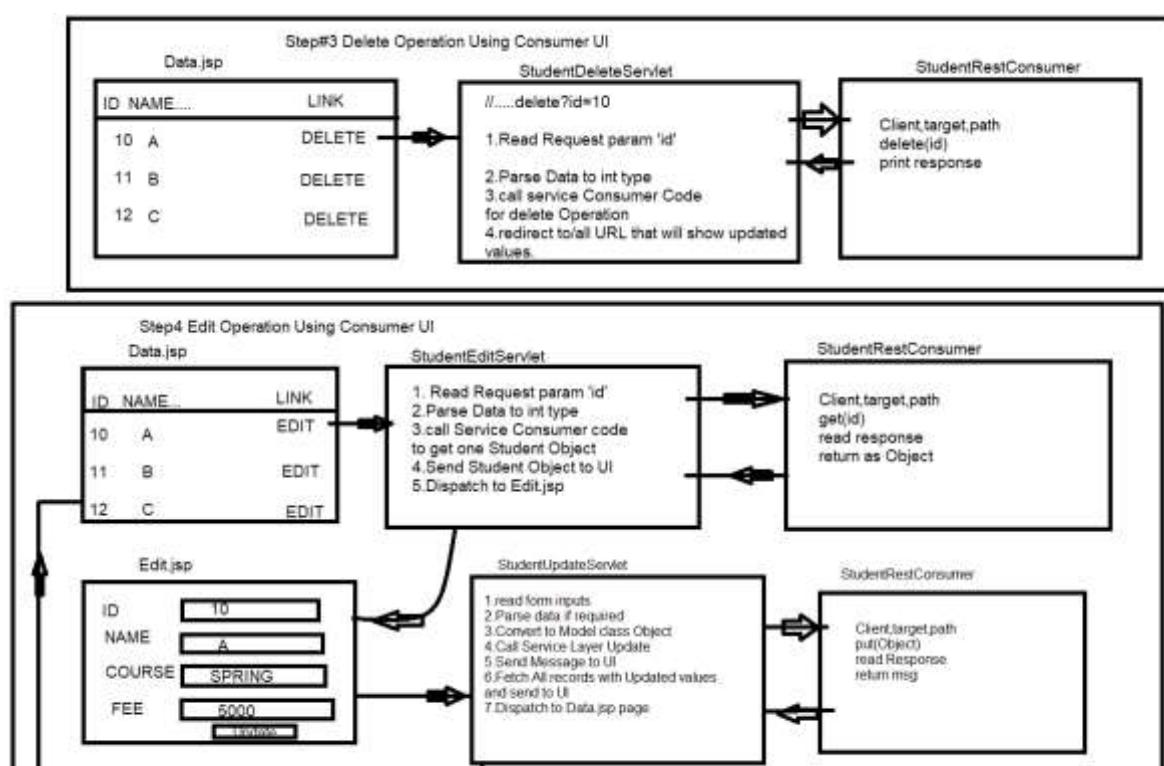
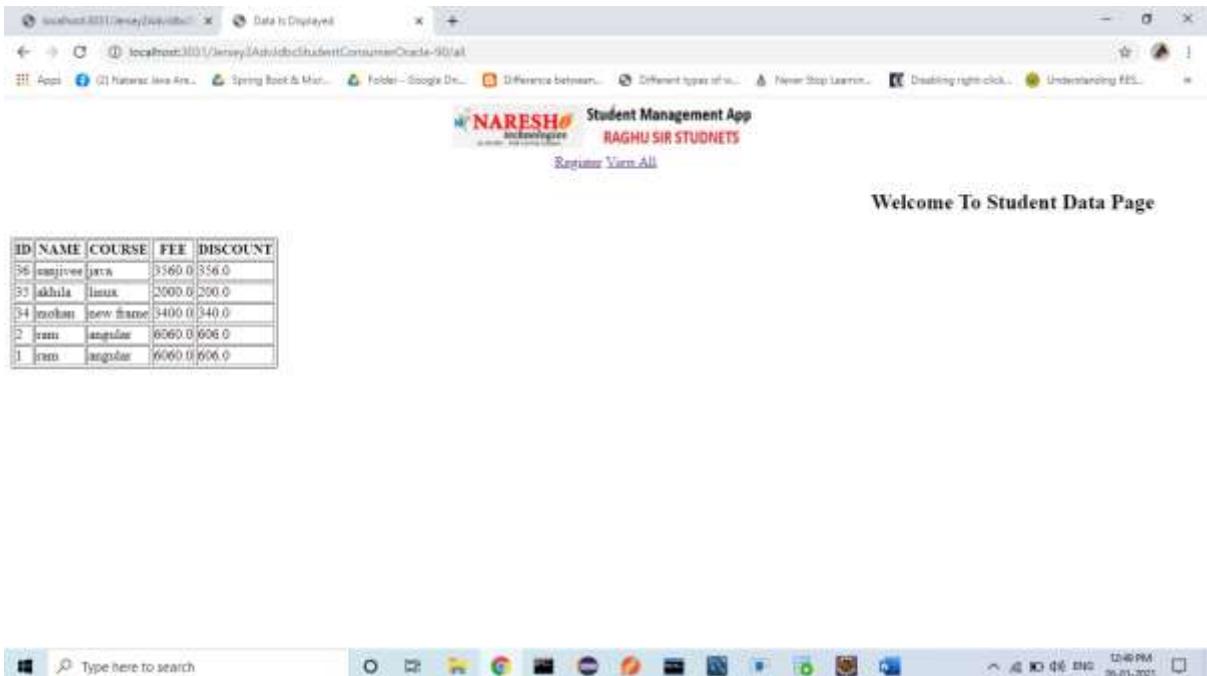
<table>
<tr>
<td>Register</td>
<td>View All</td>
</tr>
</table>
</div>
```

```
</body>
</html>
```

#### 4. Data.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Data Is Displayed</title>
</head>
<body>
<%@include file="menu.jsp" %>
<h2><marquee>Welcome To Student Data Page</marquee></h2>
<table border="1">
<tr>
<th>ID</th>
<th>NAME</th>
<th>COURSE</th>
<th>FEE</th>
<th>DISCOUNT</th>
<c:forEach items="${list}" var="ob">
<tr>
<td>${ob.stdId}</td>
<td>${ob.stdName}</td>
<td>${ob.stdCourse}</td>
<td>${ob.stdFee}</td>
<td>${ob.stdDiscount}</td>
</tr>
</c:forEach>
</tr>
</table>
</body>
</html>
```

-----Output-----



### Task#3: Consumer Code-Delete Operation

- URL Rewriting: Creating One URL using static path(delete?id=) And dynamic path(\${ob.stdId}) is called as URL Rewriting
- Example:

<a href="delete?id=\${ob.stdId}">Delete</a>

-----Code-----

Step1: Add below code in Data.jsp under <c:forEach> and <tr>

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Data Is Displayed</title>
</head>
<body>
<%@include file="menu.jsp" %>
<h2><marquee>Welcome To Student Data Page</marquee></h2>
<table border="1">
<tr>
<th>ID</th>
<th>NAME</th>
<th>COURSE</th>
<th>FEE</th>
<th>DISCOUNT</th>
<c:forEach items="${list}" var="ob">
<tr>
<td>DELETE</td>
<td>${ob.stdId}</td>
<td>${ob.stdName}</td>
<td>${ob.stdCourse}</td>
<td>${ob.stdFee}</td>
<td>${ob.stdDiscount}</td>
</tr>
</c:forEach>
</tr>
</table>
</body>
</html>
```

Step2: Add below method in StudentRestController:

```
package in.nit.consumer;

import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;

import in.nit.model.Student;

public class StudentRestConsumer {
 public static final String
URL="http://localhost:3031/Jersey2AdvJdbcStudentProducerOracle-90";
 public static String saveStudent(Student s) {
 //Consumer Code For Insert Operation
 String path="rest/students";
 String respMsg=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.post(Entity.json(s));// Convert
JSON and post
 }
 }
}
```

```
int status=resp.getStatus();
//System.out.println(status); To get Response in
Console Screen

 respMsg=resp.readEntity(String.class);

 }

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

 return respMsg;
}

public static List<Student> getAllStudents(){
 String path="/rest/students";
 List<Student> list=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.get();

 //JSON [{} ,{} ,{}] => List<Student>
 list=resp.readEntity(new
GenericType<List<Student>>() {});

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

 return list;
}
```

```
public static String deleteStudent(int id) {
 String msg=null;
 String path="/rest/students/"+id;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.delete();
 System.out.println(resp.getStatus());// to get response
status in Console
 msg=resp.readEntity(String.class);
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 return msg;
}
}
```

Step3: Create on Servlet To Perform Delete Operation

```
package in.nit.servlet;

import java.io.IOException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
```

```
import in.nit.model.Student;

@WebServlet("/delete")
public class StudentDeleteServlet extends HttpServlet {

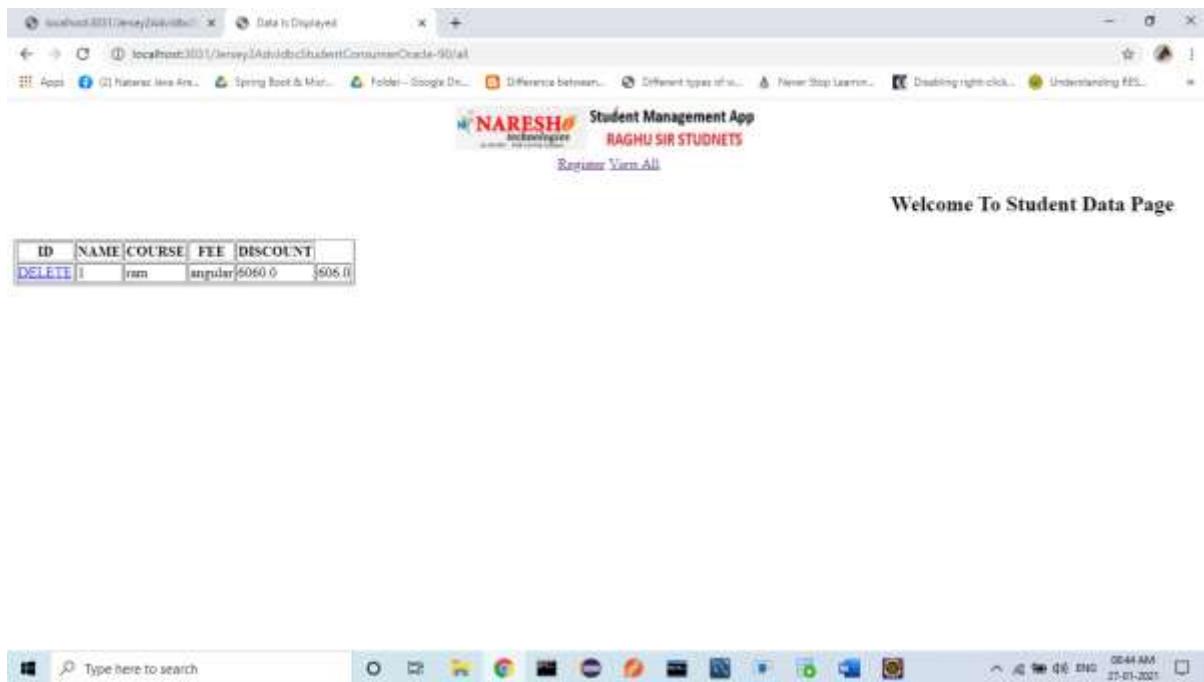
 public void doGet(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
 //read request param 'id' and parse
 Integer stdId=Integer.parseInt(req.getParameter("id"));
 //Call Service Consumer Code:
 String msg=StudentRestConsumer.deleteStudent(stdId);
 //Send Message to UI
 req.setAttribute("message", msg);

 /* //redirect to /all
 res.sendRedirect("all");*/
 //or

 //----Show to message, Fetch Latest Data-----
 List<Student> list=StudentRestConsumer.getAllStudents();
 //Send Data to UI
 req.setAttribute("list",list);
 //Dispatch to UI Page
 RequestDispatcher rd=req.getRequestDispatcher("data.jsp");
 rd.forward(req, res);
 }

}
```

-----output-----



## Task#4: Consumer Code- Update Operation

Step1: Add below Code in data.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Data Is Displayed</title>
</head>
<body>
<%@include file="menu.jsp" %>
<h2><marquee>Welcome To Student Data Page</marquee></h2>
<table border="1">
<tr>
<th>ID</th>
<th>NAME</th>
<th>COURSE</th>
<th>FEE</th>
<th>DISCOUNT</th>
```

```
<c:forEach items="${list}" var="ob">
<tr>
<td>DELETE</td>
<td>EDIT</td>
<td>${ob.stdId}</td>
<td>${ob.stdName}</td>
<td>${ob.stdCourse}</td>
<td>${ob.stdFee}</td>
<td>${ob.stdDiscount}</td>
</tr>
</c:forEach>
</tr>
</table>
</body>
</html>
```

Step2: Add below two methods in Student Rest Consumer

```
package in.nit.consumer;

import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;

import in.nit.model.Student;

public class StudentRestConsumer {
 public static final String
URL="http://localhost:3031/Jersey2AdvJdbcStudentProducerOracle-
90";
 public static String saveStudent(Student s) {
 //Consumer Code For Insert Operation
 String path="rest/students";
 String respMsg=null;
 try {
```

```
Client c=ClientBuilder.newClient();
WebTarget wt=c.target(URL).path(path);
Invocation.Builder builder=wt.request();

Response resp=builder.post(Entity.json(s));// Convert
JSON and post
int status=resp.getStatus();
//System.out.println(status); To get Response in
Console Screen
respMsg=resp.readEntity(String.class);
}
catch(Exception e) {
e.printStackTrace();
}
return respMsg;
}
public static List<Student> getAllStudents(){
String path="/rest/students";
List<Student> list=null;
try {
Client c=ClientBuilder.newClient();
WebTarget wt=c.target(URL).path(path);
Invocation.Builder builder=wt.request();
Response resp=builder.get();

//JSON [{} ,{} ,{}] => List<Student>
list=resp.readEntity(new
GenericType<List<Student>>() {});

} catch (Exception e) {
e.printStackTrace();
}
return list;
}
public static String deleteStudent(int id) {
String msg=null;
String path="/rest/students/"+id;
try {
Client c=ClientBuilder.newClient();
WebTarget wt=c.target(URL).path(path);
```

```
Invocation.Builder builder=wt.request();
Response resp=builder.delete();
System.out.println(resp.getStatus());// to get response
status in Console
msg=resp.readEntity(String.class);
}
catch(Exception e) {
 e.printStackTrace();
}
return msg;
}
public static Student getOneStudent(int id) {
 Student std=null;
 String path="/rest/students/"+id;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.get();
 std=resp.readEntity(Student.class);
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 return std;
}
}
```

Second method updateStudent:

```
package in.nit.consumer;

import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
```

```
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;

import in.nit.model.Student;

public class StudentRestConsumer {
 public static final String
URL="http://localhost:3031/Jersey2AdvJdbcStudentProducerOracle-
90";
 public static String saveStudent(Student s) {
 //Consumer Code For Insert Operation
 String path="rest/students";
 String respMsg=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.post(Entity.json(s));// Convert
JSON and post
 int status=resp.getStatus();
 //System.out.println(status); To get Response in
Console Screen
 respMsg=resp.readEntity(String.class);
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 return respMsg;
 }
 public static List<Student> getAllStudents(){
 String path="/rest/students";
 List<Student> list=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.get();

 //JSON [{} ,{} ,{}] => List<Student>
 }
 }
}
```

```
list=resp.readEntity(new
GenericType<List<Student>>() {});

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

public static String deleteStudent(int id) {
String msg=null;
String path="/rest/students/"+id;
try {
Client c=ClientBuilder.newClient();
WebTarget wt=c.target(URL).path(path);
Invocation.Builder builder=wt.request();
Response resp=builder.delete();
System.out.println(resp.getStatus());// to get response
status in Console
msg=resp.readEntity(String.class);
}
catch(Exception e) {
e.printStackTrace();
}
return msg;
}

public static Student getOneStudent(int id) {
Student std=null;
String path="/rest/students/"+id;
try {
Client c=ClientBuilder.newClient();
WebTarget wt=c.target(URL).path(path);
Invocation.Builder builder=wt.request();
Response resp=builder.get();
std=resp.readEntity(Student.class);
}
catch(Exception e) {
e.printStackTrace();
}
return std;
}
```

```
public static String updateStudent(Student s) {
 String msg=null;
 String path="/rest/students";
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();
 Response resp=builder.post(Entity.json(s));// Convert
Object to JSON
 msg=resp.readEntity(String.class);
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 return msg;
}
}
```

Step3: Show Edit Page Servlet:

```
package in.nit.servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;

@WebServlet("/edit")
public class StudentEditServlet extends HttpServlet {
 public void doGet(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
 //1. Read id param
 Integer stdId=Integer.parseInt(req.getParameter("id"));
 }
}
```

```
//2.call service layer to get Student Object
Student s=StudentRestConsumer.getOneStudent(stdId);
//3.Send this data to UI
req.setAttribute("sob",s);
//4. Dispatch to Edit JSP
RequestDispatcher rd=req.getRequestDispatcher("edit.jsp");
rd.forward(req, res);
}

}
```

Step4: Student Update Servlet:

```
package in.nit.servlet;

import java.io.IOException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;
@WebServlet("/update")
public class StudentUpdateServlet extends HttpServlet {
 public void doPost(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
//1. Read form input:
 Integer stdId=Integer.parseInt(req.getParameter("stdId"));
 String stdName=req.getParameter("stdName");
 String stdCourse=req.getParameter("stdCourse");
 Double
stdFee=Double.parseDouble(req.getParameter("stdFee"));
 //2.Convert data into model class Object
 Student sob=new Student(stdId, stdName, stdCourse,
stdFee,0.0);
 //3.call Service Consumer Code
 String msg=StudentRestConsumer.updateStudent(sob);
```

```
//4.Send Message to UI
req.setAttribute("message",msg);
//5.Fetch Latest All Rows:
List<Student> list=StudentRestConsumer.getAllStudents();
req.setAttribute("list",list);
//5.Dispatch to data.jsp page
RequestDispatcher rd=req.getRequestDispatcher("data.jsp");
rd.forward(req, res);
}

}
```

Step5: edit.jsp Page:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1" isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Edit Page</title>
</head>
<body>
<%@include file="menu.jsp" %>
<div align="center">
<h2> <marquee>Welcome To Student Edit Page</marquee></h2>
<form action="update" method="post">
<pre>
ID :<input type="text" name="stdId" value="${sob.stdId}"
readonly="readonly">

NAME :<input type="text" name="stdName" value="${sob.stdName}"
>

COURSE:<input type="text" name="stdCourse"
value="${sob.stdCourse}" >

FEE :<input type="text" name="stdFee" value="${sob.stdFee}" >

<input type="submit" value="Update">
</pre>
</form>
</div>
</body>
</html>
```

-----Consumer CRUD Operations App-----

1.index.jsp:

```
<%@page isELIgnored="false" %>
<html>
<body>
<%@include file="Menu.jsp" %>
<div align="center">
<H3>WELCOME TO STUDENT REGISTER PAGE</H3>
<form action="insert" method="POST">
<pre>
ID : <input type="text" name="stdId"/>
NAME : <input type="text" name="stdName"/>
COURSE: <input type="text" name="stdCourse"/>
FEE : <input type="text" name="stdFee"/>
 <input type="submit" value="Register"/>
</pre>
${message }
</form>
</div>
</body>
</html>
```

2.Menu.jsp:

```
<div align="center">

<table>
 <tr>
 <td>REGISTER</td>
 <td>VIEW ALL</td>
 </tr>
</table>
</div>
```

3.Student.class:

package in.nit.model;

```
import lombok.AllArgsConstructorArgsConstructor;
```

```
import lombok.Data;

import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Student {

 private Integer stdId;
 private String stdName;
 private String stdCourse;
 private Double stdFee;
 private Double stdDiscount;

}
```

#### 4.Edit.jsp:

```
<%@page isELIgnored="false" %>
<html>
<body>
<%@include file="Menu.jsp" %>
<div align="center">
<H3>WELCOME TO STUDENT EDIT PAGE</H3>
<form action="update" method="POST">
<pre>
ID : <input type="text" name="stdId" value="${sob.stdId}"
readonly="readonly"/>
NAME : <input type="text" name="stdName" value="${sob.stdName}" />
COURSE: <input type="text" name="stdCourse"
value="${sob.stdCourse}" />
FEE : <input type="text" name="stdFee" value="${sob.stdFee}" />
 <input type="submit" value="Update"/>
</pre>
</form>
</div>
```

```
</body>
</html>
```

## 5.Data.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"
 isELIgnored="false"
%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="Menu.jsp" %>
<div align="center">
<H3>WELCOME TO STUDENT DATA PAGE</H3>
<table border="1">
<tr>
 <th>ID</th>
 <th>NAME</th>
 <th>FEE</th>
 <th>COURSE</th>
 <th>DISCOUNT</th>
 <th colspan="2">LINK</th>
</tr>
<c:forEach items="${list}" var="ob">
 <tr>
 <td>${ob.stdId}</td>
 <td>${ob.stdName}</td>
 <td>${ob.stdFee}</td>
 <td>${ob.stdCourse}</td>
 <td>${ob.stdDiscount}</td>
 <td>
 DELETE
 </td>
 <td>
 EDIT
 </td>
 </tr>
</c:forEach>
</table>
</div>
</body>
</html>
```

```
</tr>
</c:forEach>
</table>
${message}
</div>
</body>
</html>
```

## 6.StudentRestConsumer:

```
package in.nit.consumer;

import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.Response;

import in.nit.model.Student;

public class StudentRestConsumer {

 private static final String URL =
"http://localhost:3031/Jersey2AdvJdbcStudentProducerOracle-90";

 public static String saveStudent(Student s) {
 String path="/rest/students";
 String respMsg=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);

 Invocation.Builder builder=wt.request();
 Response resp=builder.post(Entity.json(s));//convert to
JSON and post

 int status=resp.getStatus();
 System.out.println(status);
 }
 }
}
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
 respMsg=resp.readEntity(String.class);
 System.out.println(respMsg);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return respMsg;
}

public static List<Student> getAllStudents(){
 String path="/rest/students";
 List<Student> list=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.get();

 //JSON [{} ,{} ,{}] => List<Student>
 list=resp.readEntity(new
GenericType<List<Student>>(){});
 } catch (Exception e) {
 e.printStackTrace();
 }
 return list;
}

public static String deleteStudent(int id) {
 String msg=null;
 String path="/rest/students/"+id;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.delete();
 System.out.println(resp.getStatus());
 msg=resp.readEntity(String.class);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return msg;
}
```

```
}

public static Student getOneStudent(int id) {
 Student std=null;
 String path="/rest/students/"+id;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.get();
 std=resp.readEntity(Student.class);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return std;
}

public static String updateStudent(Student s) {
 String path="/rest/students";
 String msg=null;
 try {
 Client c=ClientBuilder.newClient();
 WebTarget wt=c.target(URL).path(path);
 Invocation.Builder builder=wt.request();

 Response resp=builder.put(Entity.json(s));//convert obj
to JSON
 msg=resp.readEntity(String.class);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return msg;
}
}
```

## 7.StudentDataServlet:

```
package in.nit.servlet;

import java.io.IOException;
import java.util.List;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;

@Override
protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
 //1. call to Consumer code
 //2. Returns List<Student>
 List<Student> list=StudentRestConsumer.getAllStudents();

 //3. Send data to UI
 req.setAttribute("list",list);
 //4. Dispatch to UI page
 RequestDispatcher
rd=req.getRequestDispatcher("Data.jsp");
 rd.forward(req, resp);

}
```

#### 8.StudentRegisterServlet:

```
package in.nit.servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;

@WebServlet("/insert") //Servlets 3.x
public class StudentRegisterServlet
 extends HttpServlet
{

 @Override
 protected void doPost(
 HttpServletRequest req,
 HttpServletResponse resp)
 throws ServletException, IOException
 {
 //1. Read Form data
 String sid=req.getParameter("stdId");
 String stdName=req.getParameter("stdName");
 String stdCourse=req.getParameter("stdCourse");
 String sfee=req.getParameter("stdFee");

 //2. Parse Data if required
 int stdId=Integer.parseInt(sid);
 double stdFee=Double.parseDouble(sfee);

 //3. Convert to Model class object
 Student std=new Student(stdId, stdName, stdCourse,
 stdFee, 0.0);

 //4. Call Consumer Code and get Response
 String msg=StudentRestConsumer.saveStudent(std);

 //5. send message to UI
 req.setAttribute("message", msg);
 System.out.println(msg);
 //6. Dispatch to Ui back
 RequestDispatcher
rd=req.getRequestDispatcher("index.jsp");
 rd.forward(req, resp);

 }
}
```

}

## 9.StudentUpdateServlet:

```
package in.nit.servlet;

import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;

@WebServlet("/update")
public class StudentUpdateServlet
 extends HttpServlet
{

 @Override
 protected void doPost(
 HttpServletRequest req,
 HttpServletResponse resp)
 throws ServletException, IOException
 {
 //1. read form inputs
 String sid=req.getParameter("stdId");
 String stdName=req.getParameter("stdName");
 String sfee=req.getParameter("stdFee");
 String stdCourse=req.getParameter("stdCourse");

 //2. parse data if required
 int stdId=Integer.parseInt(sid);
 double stdFee=Double.parseDouble(sfee);

 //3. Convert data into model class object
 Student sob=new Student(stdId, stdName, stdCourse,
 stdFee, 0.0);
```

```
//4. call service consumer code
String msg=StudentRestConsumer.updateStudent(sob);

//5. send message to UI
req.setAttribute("message", msg);

//6. fetch latest all rows
List<Student> list=StudentRestConsumer.getAllStudents();
req.setAttribute("list", list);

//7. Dispatch to Data.jsp page
req.getRequestDispatcher("Data.jsp")
.forward(req, resp);

 }
}
```

#### 10.StudentEditServlet:

```
package in.nit.servlet;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;

@WebServlet("/edit")
public class StudentEditServlet
 extends HttpServlet
{

 @Override
 protected void doGet(
 HttpServletRequest req,
 HttpServletResponse resp)
```

```
throws ServletException, IOException
{
 //1. read id param
 String id=req.getParameter("id");
 //2. parse data
 int sid=Integer.parseInt(id);
 //3. call service layer to get Student object
 Student s=StudentRestConsumer.getOneStudent(sid);
 //4. send this data to UI
 req.setAttribute("sob", s);
 //5. Dispatch to Edit JSP
 RequestDispatcher rd=req.getRequestDispatcher("Edit.jsp");
 rd.forward(req, resp);

}
}
```

## 11.StudentDeleteServlet:

```
package in.nit.servlet;

import java.io.IOException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.nit.consumer.StudentRestConsumer;
import in.nit.model.Student;

@WebServlet("/delete")
public class StudentDeleteServlet
 extends HttpServlet
{

 @Override
 protected void doGet(
 HttpServletRequest req,
```

```
HttpServletResponse resp)
throws ServletException, IOException {
//read request param 'id'
String id=req.getParameter("id");

//parse data if required
int stdId=Integer.parseInt(id);

//call service consumer code
String msg=StudentRestConsumer.deleteStudent(stdId);

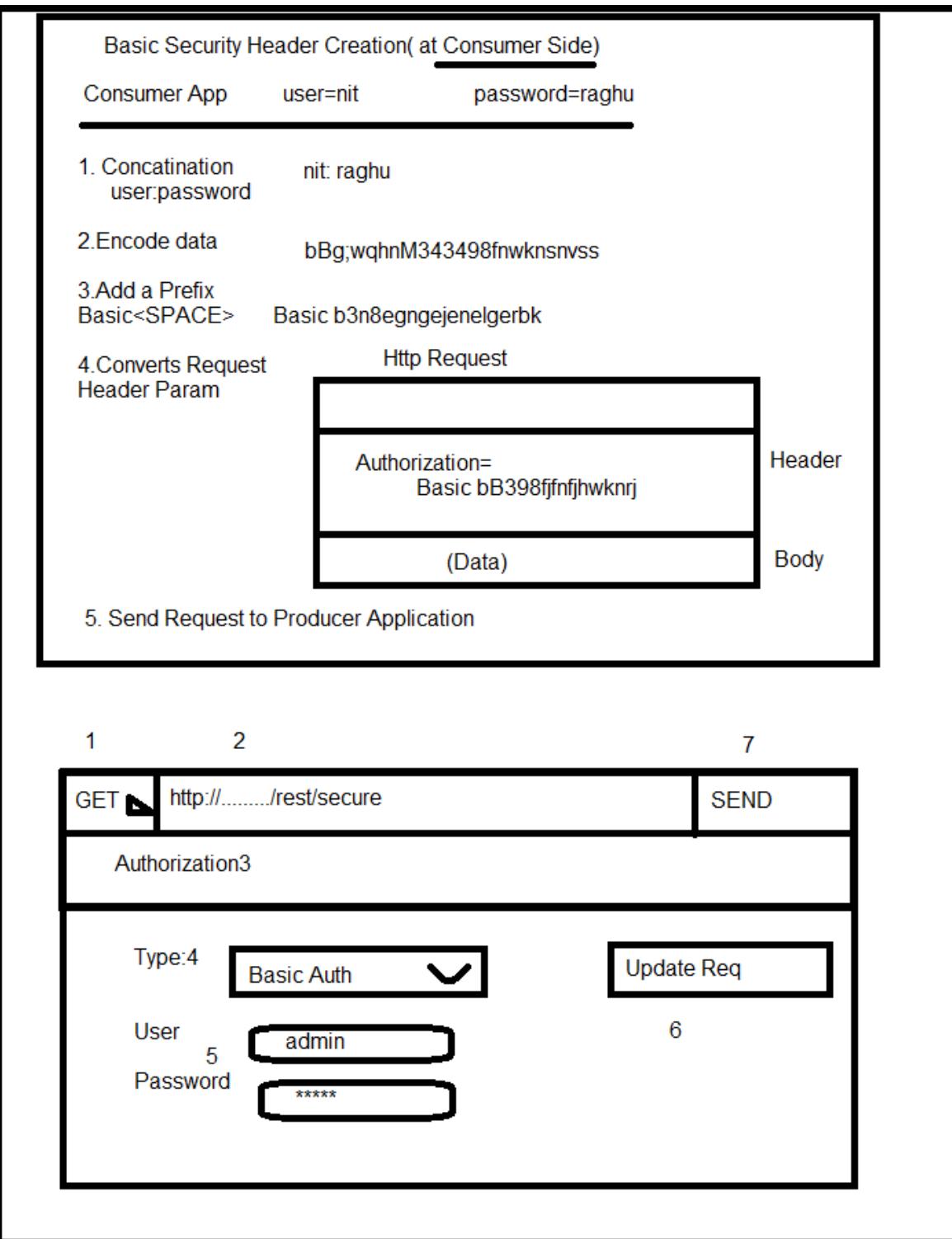
//send message to UI
req.setAttribute("message", msg);
/*
//redirect to /all
resp.sendRedirect("all");
*/
-----to show message, fetch latest data-----
List<Student> list=StudentRestConsumer.getAllStudents();

//Send data to UI
req.setAttribute("list",list);
//Dispatch to UI page
RequestDispatcher
rd=req.getRequestDispatcher("Data.jsp");
rd.forward(req, resp);

}
}
```

Note: outputs you can check....

---



### Basic Security:( Header Param+CODEC+Filter):

- Consumer creates a security token by using CODEC and sends details to the Producer as Header Param.
- If it is valid, the Producer gives a Success Response; otherwise, it returns an Error Response.

→ Consumer Request Header Param looks like:  
Authorization=Basic<SPACE><ENCODEDTOKEN>

Example:

Authorization= Basic Gdfhi4wu43u4h3gneltn48npp

--Consumer App steps for Http Request Creation for Basic Security---

1. Consumer provides user and password they are Concatenated using colon user:password
2. Above String is converted to Unreadable format (Encoded)
3. One prefix added “Basic<SPACE>”
4. It is placed in Http Request Header section with key=value format  
Authorization=Basic<SPACE><ENCODE-DATA>

-----Producer Application-----

→ \*\*\*\* We need to define one Filter (Security Filter).  
→ Try to read Header Param  
If Authorization header is null-> 400 BAD\_REQUEST (No Security Details Found)  
Else  
Read HeaderParam( as List index 0) ‘Authorization’  
Remove Basic<SPACE>  
DECODED  
Tokenize un/pwd  
Validate un/pwd (with DB)  
If(not valid ) 401 UN AUTHORIZED  
Else just do nothing-continue to RestController

Example: Request Creation: Request.status(\_\_\_.).entity(\_\_\_.).build();

---

```
SQL> create table clienttab(cid varchar2(40),csecr varchar2(40));
Table created.
SQL> insert into clienttab values('sam','ram');
1 row created.
SQL> insert into clienttab values('sankar','sankar');
1 row created.
SQL> insert into clienttab values('nit','raghu');
1 row created.
SQL> commit;
Commit complete.
```

## -----BasicSecurityProducerApp-----

Note: it is Producer App

Few of maven versions mandatory to

```
<failOnMissingWebXml>false</failOnMissingWebXml>
```

1. In pom.xml File:

```
<properties>
<failOnMissingWebXml>false</failOnMissingWebXml>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>3.8.1</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
```

```
<artifactId>jersey-container-servlet</artifactId>
<version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.30</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-codec/commons-
codec -->
<dependency>
<groupId>commons-codec</groupId>
<artifactId>commons-codec</artifactId>
<version>1.15</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
<dependency>
<groupId>com.jslsolucoes</groupId>
<artifactId>ojdbc6</artifactId>
<version>11.2.0.1.0</version>
</dependency>
```

2.DB Connection:

```
package in.nit.conn;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBCoonection {
private static Connection conn=null;
static {
try {
Class.forName("oracle.jdbc.driver.OracleDriver");

conn=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1
521:orcl","scott","tiger");
} catch (Exception e) {
e.printStackTrace();
}
}
```

```
}
```

```
public static Connection getConn() {
```

```
 return conn;
```

```
}
```

```
}
```

3.Validator class:

```
package in.nit.validator;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
import in.nit.conn.DBCoonection;
```

```
public class UserValidator {
 public static boolean isUserExist(String un, String pwd) {
 boolean flag=false;
 String sql="SELECT COUNT(CID) FROM CLIENTTAB WHERE
CID=? AND CSECR=?";
 try {
 PreparedStatement
ps=DBCoonection.getConn().prepareStatement(sql);
 ps.setString(1, un);
 ps.setString(2, pwd);
 ResultSet rs=ps.executeQuery();
 if(rs.next()) {
 long count=rs.getLong(1);
 if(count>0)
 flag=true;// User Exist
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
 return flag;
 }
}
```

StringTokenizer:

```
package in.nit.token;
```

```
import java.util.StringTokenizer;

public class TokenTest {
/*
 * StringTokenizer is used to break a String into Tokens.
 *
 */
 public static void main(String[] args) {

//StringTokenizer st=new StringTokenizer("MY,NAME IS SANKAR","","");
 StringTokenizer st=new StringTokenizer("MY NAME IS
SANKAR", " ");
 while(st.hasMoreTokens()) {
 System.out.println(st.nextToken());
 }
 System.out.println("-----");
 StringTokenizer st1=new StringTokenizer("MY NAME IS SANKAR");
 System.out.println(st1.nextToken(" "));
 }

}
```

#### 4. Security Filter:

```
package in.nit.filter;

import java.io.IOException;
import java.util.List;
import java.util.StringTokenizer;

import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;
```

```
import org.apache.commons.codec.binary.Base64;

import in.nit.validator.UserValidator;
public class SecurityFilter implements ContainerRequestFilter {
 @Context//Read Container Object
 private HttpHeaders headers;
 public void filter(ContainerRequestContext req) throws
java.io.IOException {
 System.out.println("From Filter");
 List<String>
authList=headers.getRequestHeader("Authorization");
 if(authList==null || authList.isEmpty()) {
 req.abortWith(
 Response
 .status(Status.BAD_REQUEST)
 .entity("No User Details Found")
 .build()
);
 }
 return;
 }
 //Reader Auth String from Index#0
 String auth=authList.get(0);

 //Remove Basic<SPACE>
 auth=auth.replaceAll("Basic ","");

 //Decode Auth String
 byte[] arr=Base64.decodeBase64(auth.getBytes());
 auth=new String(arr);
 //Tokenizer String
 StringTokenizer st=new StringTokenizer(auth, " : ");
 String un=st.nextToken();
 String pwd=st.nextToken();

 //Validate
 // if!("nit".equals(un) && "raghu".equals(pwd))) {
 if(!UserValidator.isUserExist(un, pwd)) {
 req.abortWith(
 Response
 .status(Status.UNAUTHORIZED)
 .entity("INVALID UN/PWD FOUND")
 .build()
);
 }
}
```

```
 return;
 }
 //else it will Continue to restController
}
}
}
```

## 5. RestController:

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/user")
public class UserRestController {

 @GET
 public String showMsg() {

 return "Welcome To Consumer";
 }
}
```

## 6.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.server.ResourceConfig;

import in.nit.filter.SecurityFilter;

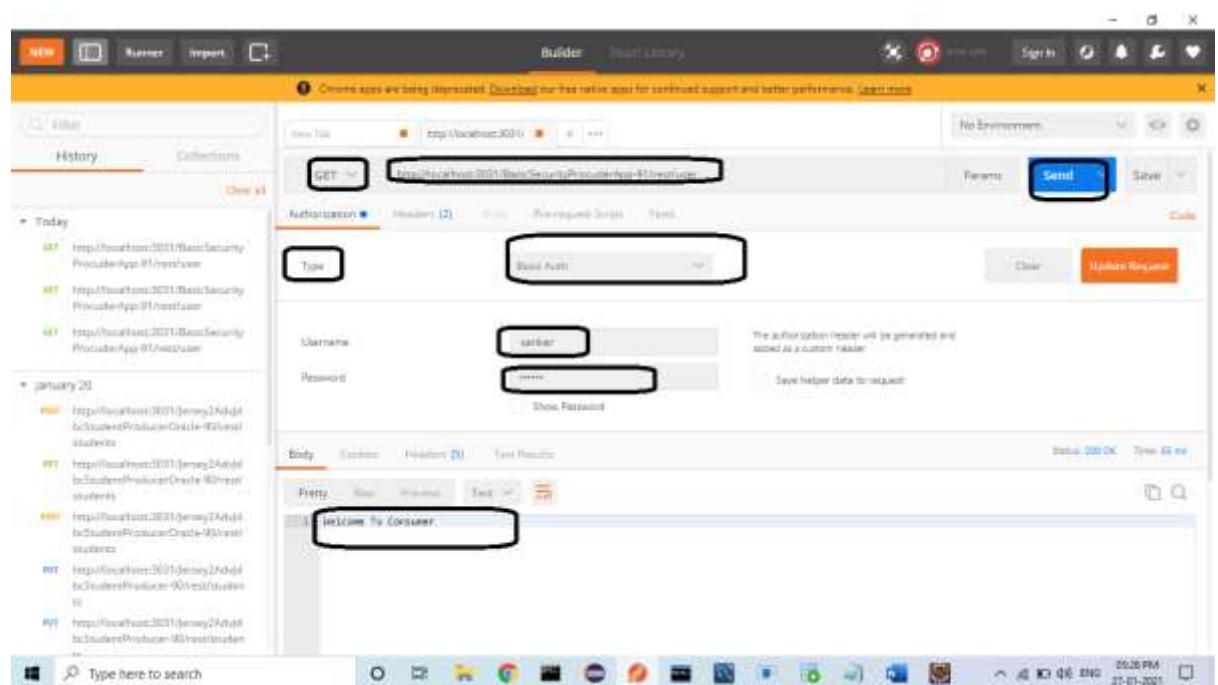
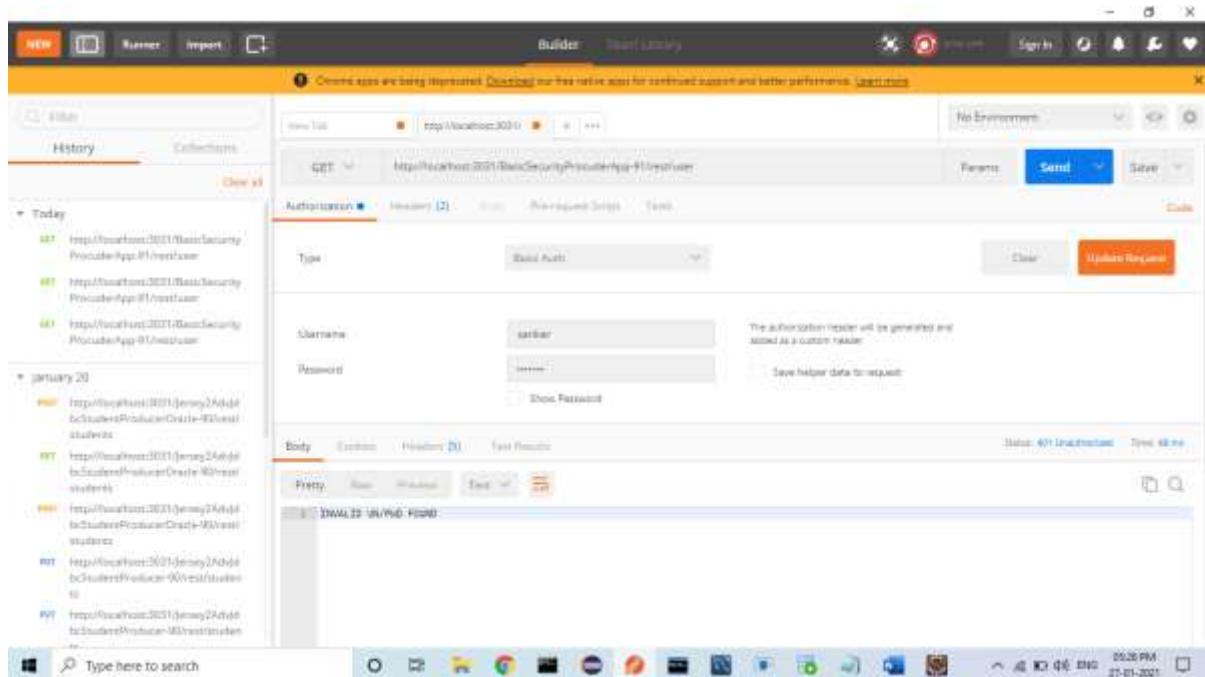
@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 packages("in.nit");
 register(SecurityFilter.class); // Activates Filter
 }
}
```

-----Output-----

In Chrome:

<http://localhost:3031/BasicSecurityProcuderApp-91/rest/user>

No User Details Found



## Security Using Role Management: [javax.annotation.security]

- This concept is used to provide access restriction to Consumer. i.e which method(which Service) can be accessed by which Consumer is decided by ROLE.
- Consider Example: Bank App having Roles like:User,Manager,Admin....etc which method can be accessed by which role given as:

transferMoney() -----User,Clerk  
openFd() -----User,Manager,Admin  
deposite() -----Cashier

Role Annotations:

1. @PermitAll : Every one access. No Security Required to access that method.(No un/pwd and No Role).
2. @DenyAll : No Consumer access this method(It might be incomplete, non-exposed, may be deprecated in future....etc).
3. @RolesAllowed : Consumer must be given valid un/pwd and matching role. Then only method can be accessed.

-----Example-----

```
@Path("/sample")
class PaymentRestController{
 @GET
 @Path("/all")
 @PermitAll
 public String showWelcome(){

 }
}
```

```
@DenyAll
@POST
Public String doExport(){

}
```

```
@RolesAllowed({"BMS","GPAY"})
@GET
public String doMerchantPay(){
.....
}
```

---

### Role Management Work Flow

- a. Consumer makes request to Producer
  - b. Producer Filter reader Request URL and compares with method URL.
  - c. If requested method is **@PermitAll**, then no consumer verification is required directly execute.
  - d. If requested method is **@DenyAll**, then no consumer is allowed to access this. Stop Execution(Abort with) Response 403 FORBIDDEN.
  - e. If requested method is **@RolesAllowed** then Read Authorization Header. If it is null OR Empty Abort with Response: 400 BAD\_REQUEST.
  - f. If Authorization is not null then Read un/pwd using Decoding and Tokenizing.
  - g. Compare UN/PWD with DB, if not matching Abort with Request: 401 UN AUTHORIZED
  - h. Read Role from DB based on UserName and compare with Method Level Roles List if not matching, Abort with Response 403 FORBIDDEN, else execute RestController.
- 

### Note:

By Using Context Object ResourceInfo we can get details of class and method which is called by Consumer.

Code Like:

```
@Context
private ResourceInfo info;
```

- From above Code get Java Method Details which is called  
→ Code Is:

```
Method method=info.getResourceMethod();
s.o.p(method.getName());
```

→ To identify method has which role annotation, Code is:  
s.o.p("PERMITALL  
METHOD="+method.isAnnotationPresent(PermitAll.class));  
s.op("DENYALL  
METHOD="+method.isAnnotationPresent(DenyAll.class));  
s.o.p("ROLES ALLOWED  
METHOD="+method.isAnnotationPresent(RolesAllowed.class));  
-----CODE PART-1-----

### 1. Pom.xml File:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>3.8.1</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/commons-
 codec/commons-codec -->
 <dependency>
 <groupId>commons-codec</groupId>
```

```
<artifactId>commons-codec</artifactId>
<version>1.15</version>
</dependency>
<!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
<dependency>
<groupId>com.jslsolucoes</groupId>
<artifactId>ojdbc6</artifactId>
<version>11.2.0.1.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.22</version>
</dependency>

</dependencies>
```

## 2. RestController:

```
package in.nit.controller;

import javax.annotation.security.DenyAll;
import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/user")
public class UserRestController {

 @GET
 @Path("/all")
 @PermitAll
 public String common() {

 return "Welcome To All Consumer!";
 }

 @GET
 @Path("/none")
```

```
@DenyAll
public String noMsg() {

 return "you can not see this!!";
}

@GET
@Path("/info")
@RolesAllowed({"EMPLOYEE","CUSTOMER"})
public String welcomeToService() {

 return "welcome to EMPLOYEE/CUSTOMER!!";
}
@GET
@Path("/form")
@RolesAllowed({"EMPLOYEE","ADMIN"})
public String welcomeToForm() {
 return "welcome to EMPLOYEE/ADMIN!!!";
}
}
```

### 3. Security Filter:

```
package in.nit.filter;
import java.lang.reflect.Method;
import java.util.List;

import javax.annotation.security.DenyAll;
import javax.annotation.security.PermitAll;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.container.ResourceInfo;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;
public class SecureFilter implements ContainerRequestFilter {
 @Context
 private ResourceInfo info;
 @Context//Read Container Object
 private HttpHeaders headers;
 @Override
```

```
public void filter(ContainerRequestContext req) throws
java.io.IOException {
 Method method=info.getResourceMethod();
 if(!method.isAnnotationPresent(PermitAll.class)) {
 if(method.isAnnotationPresent(DenyAll.class)) {
 req.abortWith(
 Response
 .status(Status.FORBIDDEN)
 .entity("NO ACCESS")
 .build());
 }
 }

 //Reader Auth Header
 List<String> authList=headers.getRequestHeader("Authorization");
 if(authList==null || authList.isEmpty()) {
 req.abortWith(
 Response
 .status(Status.BAD_REQUEST)
 .entity("No Security Details Found")
 .build()
);
 return;
 }
 //else it will Continue to restController
}
}
```

#### 4. App Config:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;
import org.glassfish.jersey.server.ResourceConfig;
import in.nit.filter.SecureFilter;

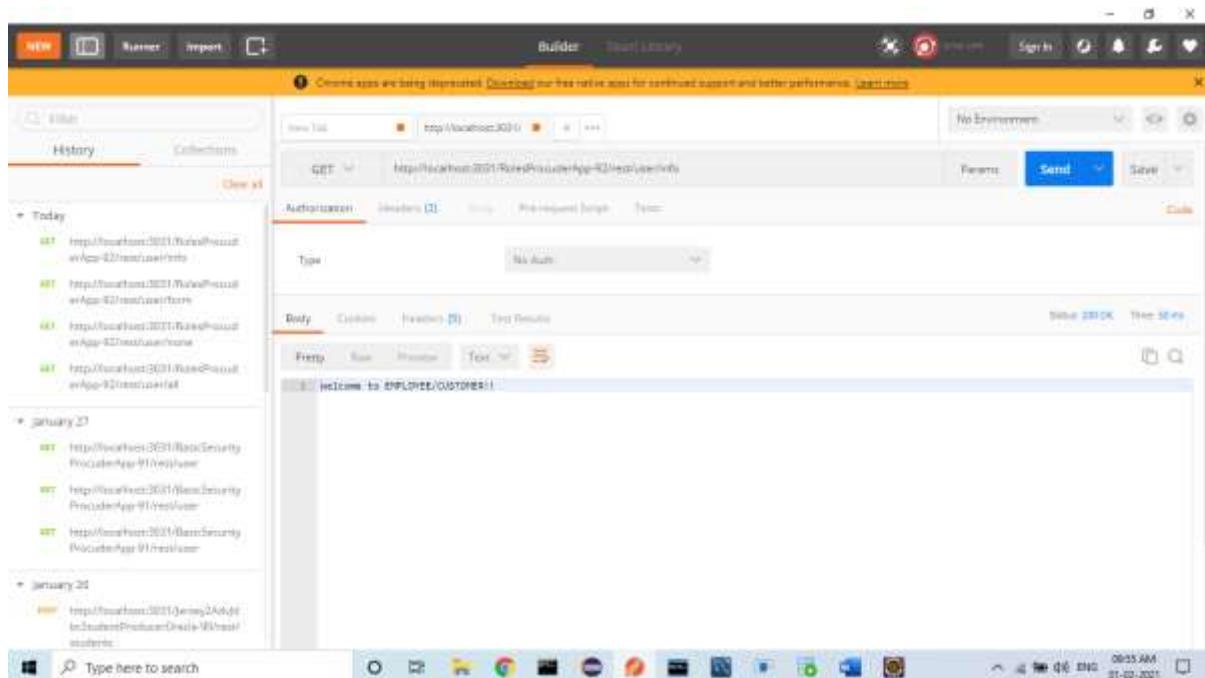
@Path("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 packages("in.nit");
 register(SecureFilter.class); // Activates Filter
 }
}
```

```
}
```

### -----Output-----

Note: not possible to See output in Chrome

Use Postman Tool:



### Role Based Authentication Using Database:

```
SQL> create table clienttabrole(cid varchar(20),csecr varchar(20),crole varchar(20));
```

Table created.

```
SQL> insert into clienttabrole values('nit','nit','ADMIN');
```

1 row created.

```
SQL> insert into clienttabrole values('sam','sam','EMPLOYEE');
```

1 row created.

```
SQL> insert into clienttabrole values('khan','khan','STUDENT');
```

1 row created.

```
SQL> create table clienttabrole(cid varchar(20),csecr varchar(20),crole varchar(20));
Table created.
SQL> insert into clienttabrole values('nit','nit','ADMIN');
1 row created.
SQL> insert into clienttabrole values('sam','sam','EMPLOYEE');
1 row created.
SQL> insert into clienttabrole values('khan','khan','STUDENT');
1 row created.
```

→ SELECT COUNT(CID) FROM CLIENTTABROLE WHERE CID='SAM' AND CSECR='AA';

-----Code – PART-2-----

1. In pom.xml file:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>3.8.1</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
```

```
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-
codec/commons-codec -->
<dependency>
 <groupId>commons-codec</groupId>
 <artifactId>commons-codec</artifactId>
 <version>1.15</version>
</dependency>
<!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
<dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.22</version>
</dependency>

</dependencies>
```

## 2.DBConnection:

```
package in.nit.conn;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBCoonection {
 private static Connection conn=null;
 static {
```

```
try {
 Class.forName("oracle.jdbc.driver.OracleDriver");

 conn=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","scott","tiger");
} catch (Exception e) {
 e.printStackTrace();
}
}

public static Connection getConn() {

 return conn;
}
}
```

### 3.UserValidator:

```
package in.nit.validator;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import in.nit.conn.DBCoonection;

public class UserValidator {
 public static boolean validateUser(String un,String pwd) {
 boolean flag=false;
 String sql="SELECT COUNT(CID) FROM CLIENTTABROLE
WHERE CID=? AND CSECR=?";
 try {
 PreparedStatement
 ps=DBCoonection.getConn().prepareStatement(sql);
 ps.setString(1, un);
 ps.setString(2, pwd);
 ResultSet rs=ps.executeQuery();
 rs.next();
 long count=rs.getLong(1);
 if(count>0)
 flag=true;// User Exist

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

```
 }
 return flag;
 }
 public static String getRoleByUser(String un) {
 String role=null;
 String sql="SELECT CROLE FROM CLIENTTABROLE
WHERE CID=?";
 try {
 PreparedStatement
ps=DBCoonection.getConn().prepareStatement(sql);
 ps.setString(1,un);
 ResultSet rs=ps.executeQuery();
 rs.next();
 role=rs.getString(1);
 }
 catch(Exception e) {
 e.printStackTrace();
 }
 return role;
 }
}
```

#### 4. SecurityRoleFilter:

```
package in.nit.filter;
import java.lang.reflect.Method;
import java.util.Arrays;
import java.util.List;
import java.util.StringTokenizer;

import javax.annotation.security.DenyAll;
import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.container.ResourceInfo;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import org.apache.tomcat.util.codec.binary.Base64;
```

```
import in.nit.validator.UserValidator;
public class SecureRoleFilter implements ContainerRequestFilter {
 @Context
 private ResourceInfo info;
 @Context//Read Container Object
 private HttpHeaders headers;
 @Override
 public void filter(ContainerRequestContext req) throws
 java.io.IOException {
 Method method=info.getResourceMethod();
 /*PERMIT ALL AND DENY ALL */
 if(method.isAnnotationPresent(PermitAll.class)) {
 //No Checking Required
 return;
 }
 if(method.isAnnotationPresent(DenyAll.class)) {
 req.abortWith(
 Response
 .status(Status.FORBIDDEN)
 .entity("NO ACCESS")
 .build());
 return;
 }
 /* AUTHORIZATION HEADER EMPTY CHECK*/
 List<String>
 authList=headers.getRequestHeader("Authorization");
 if(authList==null || authList.isEmpty()) {
 req.abortWith(
 Response
 .status(Status.BAD_REQUEST)
 .entity("No User Details Found")
 .build()
);
 return;
 }
 /* AUTHORIZATION=>UN/PWD */
 //To do Read User/pwd
 String un=null;String pwd=null;
 try {
```

```
String auth=authList.get(0);

//Remove Basic<SPACE>
auth=auth.replaceAll("Basic ", "");

//Decode Auth
byte[]
arr=Base64.decodeBase64(auth.getBytes());

//Convert to String
auth=new String(arr);

//Tokenize Data
StringTokenizer st=new StringTokenizer(auth,":");

un=st.nextToken();
pwd=st.nextToken();

} catch (Exception e) {
e.printStackTrace();
req.abortWith(
 Response

.status(Status.INTERNAL_SERVER_ERROR)
.entity("UNABLE TO GET
AUTHORIZATION DATA")
.build()
);
return;
}
/* VERIFY UN/PWD WITH DB */
if(!UserValidator.validateUser(un, pwd)) {
req.abortWith(
 Response
.status(Status.UNAUTHORIZED)
.entity("INVALID UN/PWD [User/Pwd not Exist]")
.build()
);
return;
}
/* Read User Role and Verify Here*/
//read current request method level roles
```

```
String[]
rolesArr=method.getAnnotation(RolesAllowed.class).value();

//Convert into List
List<String> methodRoles=Arrays.asList(rolesArr);
System.out.println(methodRoles);

//Get User Roles from DB based on UN
String userRole=UserValidator.getRoleByUser(un);

//check user role is allowed for request method
if(!methodRoles.contains(userRole)) {
 req.abortWith(
 Response
 .status(Status.FORBIDDEN)
 .entity("INVALID ROLE TO ACCESS THIS
SERVICE!")
 .build()
);
 return;
}
}
}
```

## 5.UserRestController:

```
package in.nit.controller;

import javax.annotation.security.DenyAll;
import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/user")
public class UserRestController {

 @GET
 @Path("/all")
 @PermitAll
 public String common() {

 return "Welcome To All Consumer!";
 }
}
```

```
}

@GET
@Path("/none")
@DenyAll
public String noMsg() {

 return "you can not see this!!";
}

@GET
@Path("/info")
@RolesAllowed({"EMPLOYEE","CUSTOMER"})
public String welcomeToService() {

 return "welcome to EMPLOYEE/CUSTOMER!!";
}
@GET
@Path("/form")
@RolesAllowed({"EMPLOYEE","ADMIN"})
public String welcomeToForm() {
 return "welcome to EMPLOYEE/ADMIN!!!";
}
}
```

## 6.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

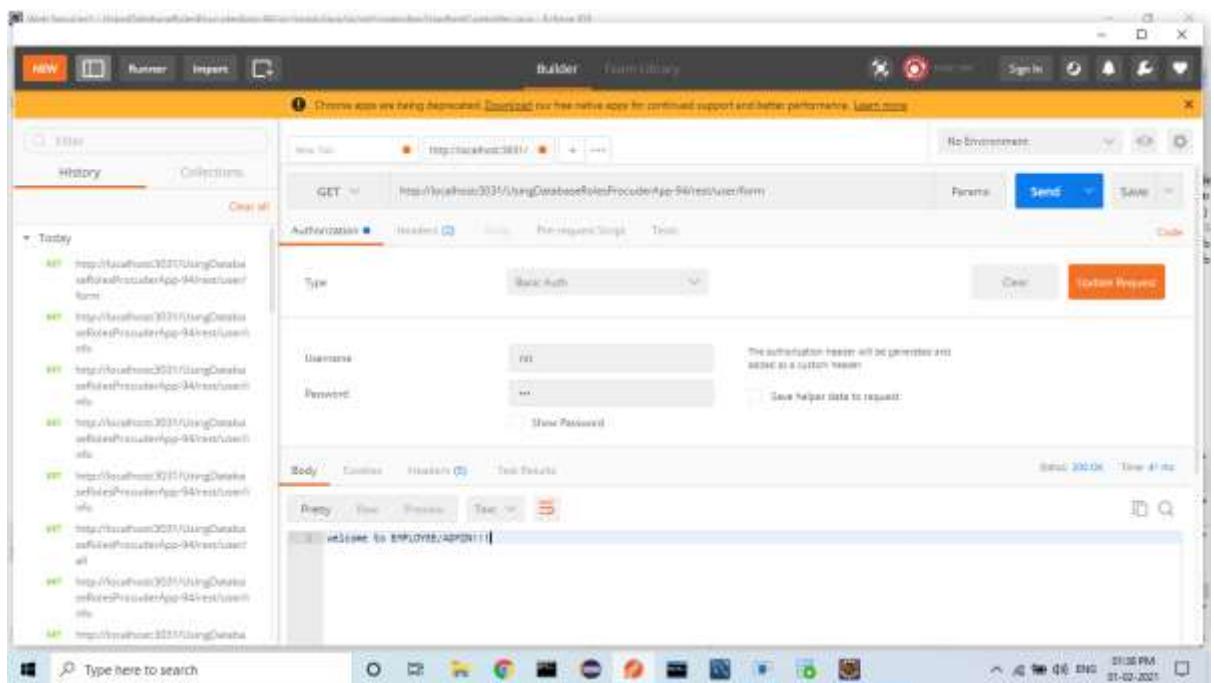
import org.glassfish.jersey.server.ResourceConfig;
import in.nit.filter.SecureRoleFilter;

@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 packages("in.nit");
 register(SecureRoleFilter.class); // Activates Filter
 }
}
```

-----output-----

Not possible to see output in chrome:

To see output in POSTMAN Screen:



## HttpHeaders:

### HttpHeaders In Rest API:

|                        |                       |                         |                          |
|------------------------|-----------------------|-------------------------|--------------------------|
| 1. ACCEPT              | 11. CONTENT_LANGUAGE  | 21. IF_NONE_MATCH       | 31. SET_COOKIE           |
| 2. ACCEPT_CHARSET      | 12. CONTENT_LENGTH    | 22. IF_UNMODIFIED_SINCE | 32. LAST_EVENT_ID_HEADER |
| 3. ACCEPT_ENCODING     | 13. CONTENT_LOCATION  | 23. LAST_MODIFIED       |                          |
| 4. ACCEPT_LANGUAGE     | 14. CONTENT_TYPE      | 24. LOCATION            |                          |
| 5. ALLOW               | 15. DATE              | 25. LINK                |                          |
| 6. AUTHORIZATION       | 16. ETag              | 26. RETRY_AFTER         |                          |
| 7. CHACHE_CONTROL      | 17. EXPIRES           | 27. USER_AGENT          |                          |
| 8. CONTENT_DISPOSITION | 18. HOST              | 28. VARY                |                          |
| 9. CONTENT_ENCODING    | 19. IF_MATCH          | 29. WWW_AUTHENTICATE    |                          |
| 10. CONTENT_ID         | 20. IF_MODIFIED_SINCE | 30. COOKIE              |                          |

Accept: Request for certain media types

Authorization: Checking UserName and Password(Credentials).

Content-Type: Indicates which type of media Entity-Body is giving request.

User-Agent: Information about user request.

## Hibernate properties:

## hibernate properties

1. hibernate.c3p0
  2. hibernate.connection
  3. hibernate.dialect
  4. hibernate.format
  5. hibernate.hdm2ddl
  6. hibernate.jdbc
  7. hibernate.jndi
  8. hibernate.proxool
  9. hibernate.session
  10. hibernate.session\_factory
  11. hibernate.show
- 

## Mini Project#2-Spring WEB MVC(Consumer)+ JAX-RS with Hibernate(Producer)

---

-----Full Code for Producer App-----

1.pom.xml file:

Depencencies:

Jersey Container Servlet  
Jersey HK2  
Jersey Media JSON Jackson  
MySQL Connector  
Hibernate Core  
Project Lombok

<properties>

```
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
 <dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
 </dependency>

 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.18</version>
 </dependency>
```

```
<scope>provided</scope>
</dependency>
```

## 2. Model class:

```
package in.nit.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="product")
public class Product {
@Id
@GeneratedValue
@Column(name="pid")
private Integer proId;

@Column(name="PCODE")
private String prodCode;

//@Column(name="pcost")
private Double prodCost;

@Column(name="pdisc")
private Double prodDiscount;

@Column(name="pgst")
private Double prodGst;
}
```

## 3.Configuration File:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
```

```
<hibernate-configuration>
 <session-factory>
 <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
 <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/webservices</property>
 <property name="hibernate.connection.username">root</property>
 <property name="hibernate.connection.password">root</property>

 <property
name="hibernate.dialect">org.hibernate.dialect.MySQL55Dialect</property>
 <property name="hibernate.show_sql">true</property>
 <property name="hibernate.format_sql">true</property>
 <property name="hibernate.hbm2ddl.auto">create</property>
 <mapping class="in.nit.model.Product"></mapping>
 </session-factory>
</hibernate-configuration>
```

#### 4.HibernateUtil

```
package in.nit.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
 private static SessionFactory sf;
 static {
 try {
 sf=new Configuration().configure("/in/nit/cfgs/hibernate.cfg.xml").buildSessionFactory();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 public static SessionFactory getSF() {
 return sf;
 }
}
```

}

5.dao:

```
package in.nit.dao;

import in.nit.model.Product;

public interface IProductDAO {
 public Integer saveProduct(Product p);
}
```

6.impl:

```
package in.nit.dao.impl;

import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.dao.IProductDAO;
import in.nit.model.Product;
import in.nit.util.HibernateUtil;

public class ProductDAOImpl implements IProductDAO {

 @Override
 public Integer saveProduct(Product p) {
 Session ses=HibernateUtil.getSF().openSession();
 Transaction tx=null;
 Integer id=null;
 try(ses) {
 tx=ses.beginTransaction();
 id=(Integer)ses.save(p);
 tx.commit();
 } catch (Exception e) {
 if(tx!=null && tx.getStatus().canRollback()) {
 tx.rollback();
 }
 e.printStackTrace();
 }
 return id;
 }
}
```

```
}
```

```
}
```

7.service:

```
package in.nit.service;

import in.nit.model.Product;

public interface IProductService {
 public Integer saveProduct(Product p);
}
```

8.impl:

```
package in.nit.service.impl;
import javax.inject.Inject;

import in.nit.dao.IProductDAO;
import in.nit.model.Product;
import in.nit.service.IProductService;

public class ProductServiceImpl implements IProductService {
 @Inject
 private IProductDAO dao;

 @Override
 public Integer saveProduct(Product p) {
 var cost=p.getProdCost();
 var discount=cost*8/100.0;
 var gst=cost*6/100.0;
 p.setProdGst(gst);
 p.setProdDiscount(discount);

 return dao.saveProduct(p);
 }
}
```

9.AppConfig:

```
package in.nit.config;
```

```
import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IProductDAO;
import in.nit.dao.impl.ProductDAOImpl;
import in.nit.service.IProductService;
import in.nit.service.impl.ProductServiceImpl;

@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig(){
 packages("in.nit");
 register(new AbstractBinder() {
 public void configure() {
 //IProductDAO dao=new ProductDAOImpl();

 bind(ProductDAOImpl.class).to(IProductDAO.class);

 bind(ProductServiceImpl.class).to(IProductService.class);
 }
 });
 }
}
```

## 10.RestController:

```
package in.nit.controller;

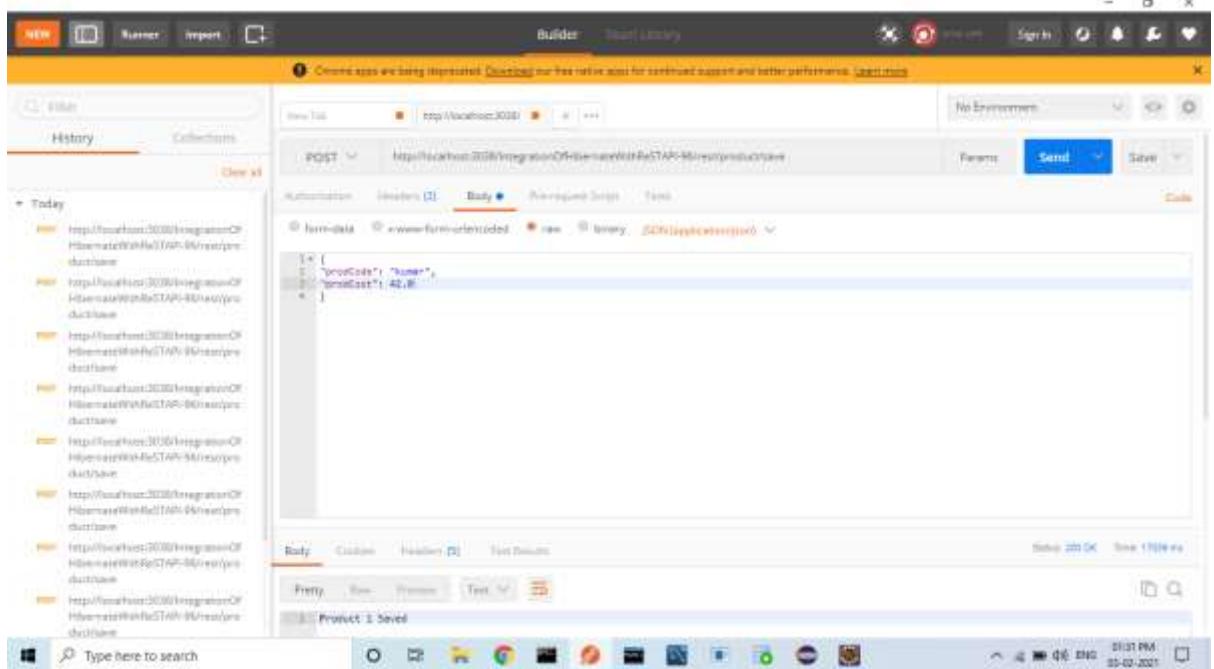
import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Product;
import in.nit.service.IProductService;

@Path("/product")
```

```
public class ProductRestController {
 @Inject
 private IProductService service;
 //1.Save Products
 @POST
 @Path("/save")
 @Consumes("application/json")
 public Response saveProduct(Product p) {
 Response resp=null;
 Integer id=null;
 try {
 id=service.saveProduct(p);
 resp=Response
 .status(Status.OK)
 .entity("Product "+id+" Saved")
 .build();
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)
 .entity("Product unable to Save")
 .build();
 e.printStackTrace();
 }
 return resp;
 }
}
```

-----Output-----



```
mysql> use webservices;
Database changed
mysql> select * from product
-> ;
+----+----+----+----+
| pid | pcode | prodCost | pdisc | pgst |
+----+----+----+----+
| 1 | sam | 87 | 6.96 | 1450 |
+----+----+----+----+
1 row in set (0.00 sec)
```

Note: if you are saving data then use

```
<property name="hibernate.hbm2ddl.auto">create</property>
```

### Update Operation :

#### 1.pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<maven.compiler.source>15</maven.compiler.source>
<maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
 <dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
 </dependency>

 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.18</version>
 <scope>provided</scope>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-
media-json-jackson -->
 <dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-json-jackson</artifactId>
 <version>2.30</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
 <dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.22</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
 <dependency>
 <groupId>org.hibernate</groupId>
 <artifactId>hibernate-core</artifactId>
 <version>5.4.12.Final</version>
 </dependency>

</dependencies>
```

## 2. Model class:

```
package in.nit.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="product")
public class Product {
 @Id
 @GeneratedValue
```

```
@Column(name="pid")
private Integer proId;

@Column(name="PCODE")
private String prodCode;

//@Column(name="PCOST")
private Double prodCost;

@Column(name="PDISC")
private Double prodDiscount;

@Column(name="PGST")
private Double prodGst;
}
```

### 3.Configuration File:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
 <session-factory>
 <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</prop
erty>
 <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/webservic
es</property>
 <property name="hibernate.connection.username">root</property>
 <property name="hibernate.connection.password">root</property>

 <property
name="hibernate.dialect">org.hibernate.dialect.MySQL55Dialect</prope
rty>
 <property name="hibernate.show_sql">true</property>
 <property name="hibernate.format_sql">true</property>
 <!-- <property name="hibernate.hbm2ddl.auto">create</property>-
->
 <property name="hibernate.hbm2ddl.auto">update</property>
 <mapping class="in.nit.model.Product"></mapping>
```

```
</session-factory>
</hibernate-configuration>
```

4.dao interface:

```
package in.nit.dao;

import in.nit.model.Product;

public interface IProductDAO {
 public Integer saveProduct(Product p);
 public void updateProduct(Product p);
}
```

5. Dao impl:

```
package in.nit.dao.impl;

import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.dao.IProductDAO;
import in.nit.model.Product;
import in.nit.util.HibernateUtil;

public class ProductDAOImpl implements IProductDAO {

 @Override
 public Integer saveProduct(Product p) {
 Session ses=HibernateUtil.getSF().openSession();
 Transaction tx=null;
 Integer id=null;
 try(ses) {
 tx=ses.beginTransaction();
 id=(Integer)ses.save(p);
 tx.commit();
 } catch (Exception e) {
 if(tx!=null && tx.getStatus().canRollback()) {
 tx.rollback();
 }
 e.printStackTrace();
 }
 return id;
 }
}
```

```
 }

 @Override
 public void updateProduct(Product p) {
 Session ses=HibernateUtil.getSF().openSession();
 Transaction tx=null;
 try(ses) {
 tx=ses.beginTransaction();
 ses.update(p);
 tx.commit();
 } catch (Exception e) {
 if(tx!=null && tx.getStatus().canRollback()) {
 tx.rollback();
 }
 e.printStackTrace();
 }
 }
}
```

## 6. Service Interface:

```
package in.nit.service;

import in.nit.model.Product;

public interface IProductService {
 public Integer saveProduct(Product p);
 public void updateProduct(Product p);
}
```

## 7. Service impl:

```
package in.nit.service.impl;
import javax.inject.Inject;

import in.nit.dao.IProductDAO;
import in.nit.model.Product;
import in.nit.service.IProductService;

public class ProductServiceImpl implements IProductService {
 @Inject
 private IProductDAO dao;
```

```
@Override
public Integer saveProduct(Product p) {
 var cost=p.getProdCost();
 var discount=cost*8/100.0;
 var gst=cost*6/100.0;
 p.setProdGst(gst);
 p.setProdDiscount(discount);

 return dao.saveProduct(p);
}

@Override
public void updateProduct(Product p) {
 var cost=p.getProdCost();
 var discount=cost*8/100.0;
 var gst=cost*6/100.0;
 p.setProdCost(cost);
 p.setProdDiscount(discount);
 p.setProdGst(gst);
 dao.updateProduct(p);
}
}
```

## 8. HibernateUtil:

```
package in.nit.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
private static SessionFactory sf;
static {
 try {
 sf=new
Configuration().configure("/in/nit/cfgs/hibernate.cfg.xml").buildSessionFactory();
 } catch (Exception e) {
 e.printStackTrace();
 }
}
```

```
public static SessionFactory getSF() {
 return sf;
}
}
```

## 9.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.hk2.utilities.binding.AbstractBinder;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.dao.IProductDAO;
import in.nit.dao.impl.ProductDAOImpl;
import in.nit.service.IProductService;
import in.nit.service.impl.ProductServiceImpl;

@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig(){
 packages("in.nit");
 register(new AbstractBinder() {
 public void configure() {
 //IProductDAO dao=new ProductDAOImpl();
 bind(ProductDAOImpl.class).to(IProductDAO.class);

 bind(ProductServiceImpl.class).to(IProductService.class);
 }
 });
 }
}
```

## 10.RestController:

```
package in.nit.controller;

import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
```

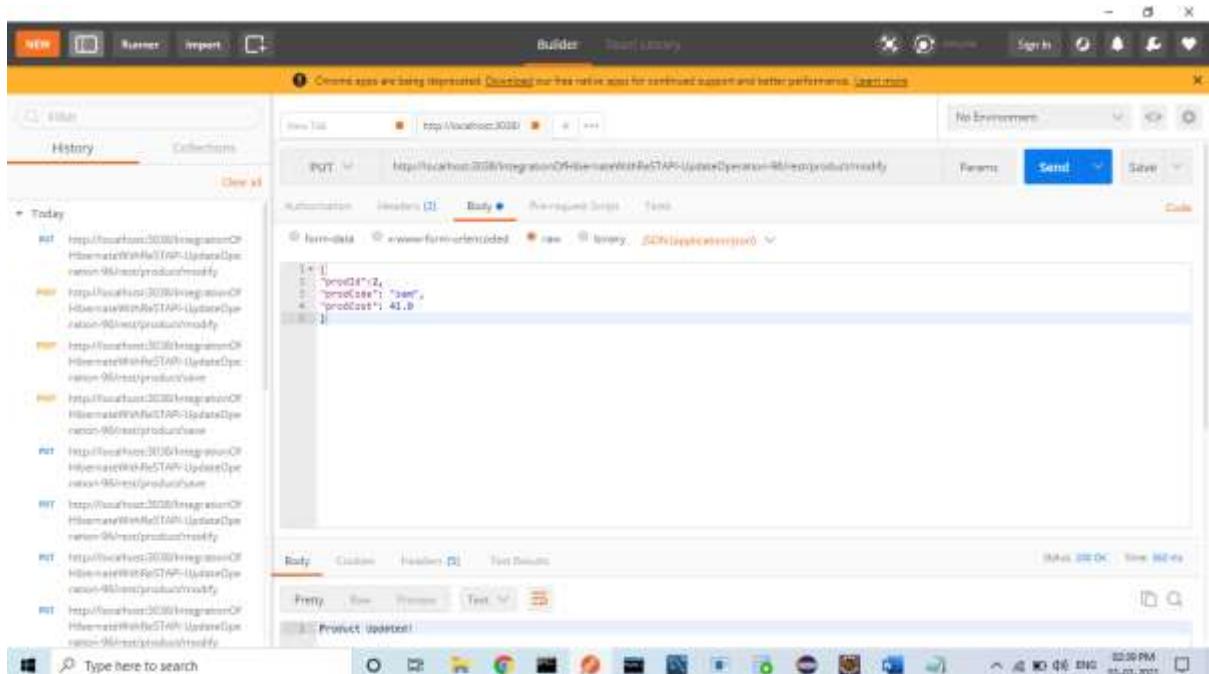
```
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.model.Product;
import in.nit.service.IProductService;

@Path("/product")
public class ProductRestController {
 @Inject
 private IProductService service;
 //1.Save Products
 @POST
 @Path("/save")
 @Consumes("application/json")
 public Response saveProduct(Product p) {
 Response resp=null;
 Integer id=null;
 try {
 id=service.saveProduct(p);
 resp=Response
 .status(Status.OK)
 .entity("Product "+id+" Saved")
 .build();
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)
 .entity("Product unable to Save")
 .build();
 e.printStackTrace();
 }
 return resp;
 }
 //Update Product
 @PUT
 @Path("/modify")
 @Consumes("application/json")
 public Response updateProduct(Product product) {
 Response resp=null;
 try {
 service.updateProduct(product);
 resp=Response
 .status(Status.OK)
 .entity("Product Updated!")
 }
```

```
 .build();
 } catch (Exception e) {
 resp=Response
 .status(Status.INTERNAL_SERVER_ERROR)
 .entity("Unable to Update!")
 .build();
 e.printStackTrace();
 }
 return resp;
}
}
```

-----Output-----



```
mysql> select * from product;
+----+----+----+----+
| pid | pcode | prodCost | pdisc | pgst |
+----+----+----+----+
| 1 | likitha | 41 | 3.28 | 2.46 |
| 2 | Sankar | 41 | 3.28 | 2.46 |
+----+----+----+----+
2 rows in set (0.04 sec)
```

```
mysql> select * from product;
+----+----+----+----+
| pid | pcode | prodCost | pdisc | pgst |
```

```
+-----+-----+
| 1 | likitha | 41 | 3.28 | 2.46 |
| 2 | sam | 41 | 3.28 | 2.46 |
+-----+-----+
2 rows in set (0.00 sec)
```

Note: if you are performing update operation then use:

```
<property name="hibernate.hbm2ddl.auto">update</property>
```

---

## load() and get() difference in Hibernate:

Example:

1. Pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!--
 https://mvnrepository.com/artifact/org.projectlombok/lombok -->
 <dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.18</version>
 <scope>provided</scope>
 </dependency>

 <!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
 <dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.22</version>
```

```
</dependency>

<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
 <groupId>org.hibernate</groupId>
 <artifactId>hibernate-core</artifactId>
 <version>5.4.27.Final</version>
</dependency>
<!--
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6 -->
<dependency>
 <groupId>com.jslsolucoes</groupId>
 <artifactId>ojdbc6</artifactId>
 <version>11.2.0.1.0</version>
</dependency>
</dependencies>
```

## 2. Configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>

 <!-- Connection Properties -->
 <property
 name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
 <property
 name="connection.url">jdbc:mysql://localhost:3306/webservices</proper
ty>
 <property name="connection.username">root</property>
 <property name="connection.password">root</property>

 <!-- dialect,show sql,format properties -->
```

```
<property
name="dialect">org.hibernate.dialect.MySQL55Dialect</property>
<property name="show_sql">true</property>
<property name="format_sql">true</property>

<!-- Table Creation and Updation properties -->
<property name="hbm2ddl.auto">update</property>

<!-- Model class linking -->
<mapping class="in.nit.model.Company"/>

</session-factory>
</hibernate-configuration>
```

### 3. Model class Company

```
package in.nit.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.Data;
@SuppressWarnings("serial")
@Data
@Entity
//if not mention @table then Model class as table first letter small
public class Company implements Serializable{
 @Id
 @GeneratedValue(strategy = GenerationType.SEQUENCE)
 private Integer cmpId;
 private String cmpName;
 private String cmpContact;
}
```

### 4. Hibernate Util

```
package in.nit.util;
```

```
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

 private static SessionFactory sf;

 static {
 try {
 sf=new
Configuration().configure("/in/nit/cfgs/hibernate.cfg.xml").buildSessionFactory();
 } catch (Exception e) {
 e.printStackTrace();
 }
 }

 public static SessionFactory getSF() {
 return sf;
 }
}
```

### 5. SaveTest:

```
package in.nit.test;
import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.model.Company;
import in.nit.util.HibernateUtil;

public class SaveTest {
 public static void main(String[] args) {
 Session ses=HibernateUtil.getSF().openSession();
 Transaction tx=null;
 try(ses){
 tx=ses.beginTransaction();
 //save Record into db
 Company c=new Company();
 c.setCmpName("Wipro");
 c.setCmpContact("wipro@gmail.com");
 ses.save(c);
 }
 }
}
```

```
 System.out.println("Record Saved ");
 tx.commit();
 }
 catch(Exception e) {
 if(tx!=null && tx.getStatus().canRollback()) {
 tx.rollback();
 e.printStackTrace();
 System.out.println("Record Not Saved ");
 }
 }
}
```

## 6. LoadTest:

```
package in.nit.test;
import java.util.Scanner;

import org.hibernate.Session;
import org.hibernate.Transaction;

import in.nit.model.Company;
import in.nit.util.HibernateUtil;

public class LoadTest {
 public static void main(String[] args) {
 Session ses=HibernateUtil.getSF().openSession();
 Transaction tx=null;
 try(ses){
 tx=ses.beginTransaction();
 Scanner sc=new Scanner(System.in);
 System.out.println("Enter ID Integer Value: ");
 int id=sc.nextInt();
 Company c=ses.load(Company.class,id);
 System.out.println("Fetch data"+c);
 tx.commit();
 sc.close();
 }
 catch(Exception e) {
 if(tx!=null && tx.getStatus().canRollback()) {
 tx.rollback();
 e.printStackTrace();
 System.out.println("Failed to fetch data");
 }
 }
 }
}
```

```
 }
}
}
```

## 7. GetTest:

```
package in.nit.test;
import java.util.Scanner;

import org.hibernate.Session;
import org.hibernate.Transaction;

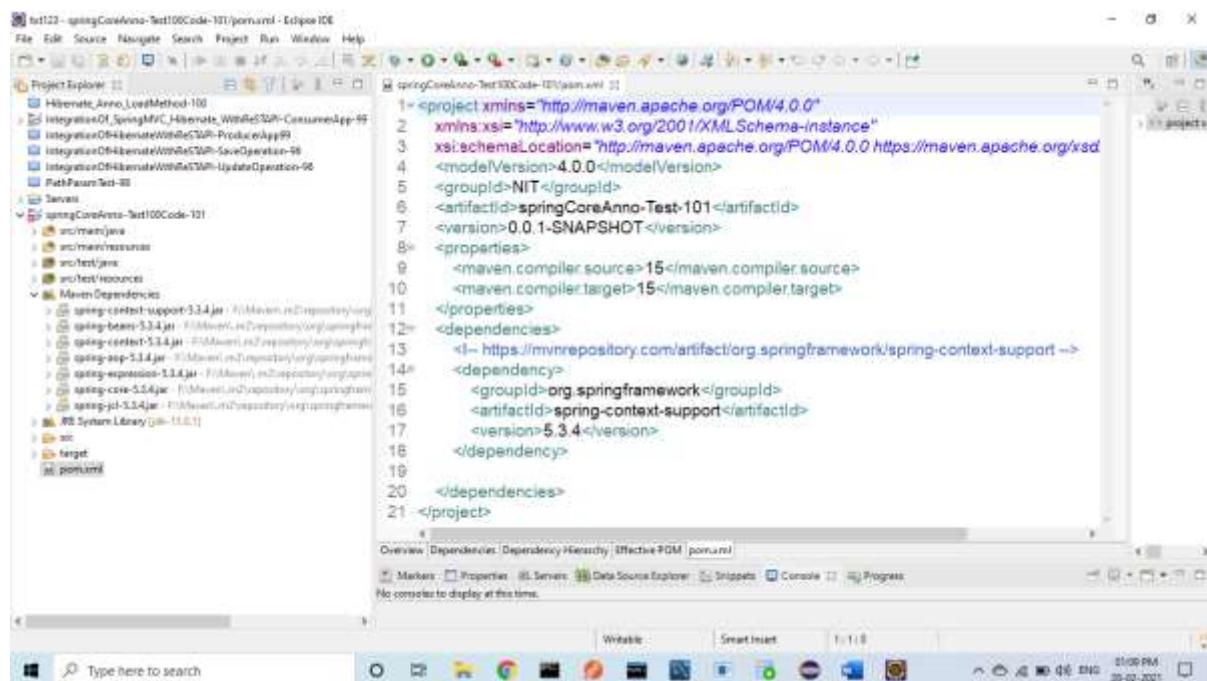
import in.nit.model.Company;
import in.nit.util.HibernateUtil;

public class GetTest {
 public static void main(String[] args) {
 Session ses=HibernateUtil.getSF().openSession();
 Transaction tx=null;
 try(ses){
 tx=ses.beginTransaction();
 Scanner sc=new Scanner(System.in);
 System.out.println("Enter ID Integer Value: ");
 int id=sc.nextInt();
 Company c=ses.get(Company.class,id);
 System.out.println("Fetch data"+c);
 tx.commit();
 sc.close();
 }
 catch(Exception e) {
 if(tx!=null && tx.getStatus().canRollback()) {
 tx.rollback();
 e.printStackTrace();
 System.out.println("Failed to fetch data");
 }
 }
 }
}
```

|                   | load()                                                                | get()                                 |
|-------------------|-----------------------------------------------------------------------|---------------------------------------|
| Opration:         | Used for Fetch Data                                                   | Used for Fetch Data                   |
| Record not Found: | If Record not Fount getting NULL                                      | If Record not Found getting Exception |
| Performance:      | slower                                                                | slightly faster                       |
| Lazy Loading      | it return Proxy Object[Fake Object] will not hit Database Immediately | No                                    |
| Eager Loading     | No                                                                    | Will hit Database Immediately         |
| Use Case:         | If you not known ID value exist or not                                | If you known ID value exist           |

## Spring Core 100% Program:

### Step1: pom.xml and dependencies



### Step2:

#### Welcome.java:

```

package in.nit.bean;

import org.springframework.stereotype.Component;

@Component(value = "wel")
public class Welcome {

```

```
public String msg() {
 return "Message From Welcome class";
}
}
```

Step3:

AppConfig:

```
package in.nit.config;
```

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan(basePackages = "in.nit.bean")
public class AppConfig {
```

```
}
```

Step4:

Spring Test.java:

```
package in.nit.test;
```

```
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
import in.nit.bean.Welcome;
import in.nit.config.AppConfig;
```

```
public class SpringTest {
```

```
 public static void main(String[] args) {
 //Create AnnotationConfigApplicationContext Container
```

Object

```
 AnnotationConfigApplicationContext acc=new
AnnotationConfigApplicationContext();
 //provide configuration class
 acc.register(AppConfig.class);
 acc.refresh();
```

```
Welcome we=acc.getBean("wel",Welcome.class);
System.out.println/we.msg());
acc.close();
}
}
```

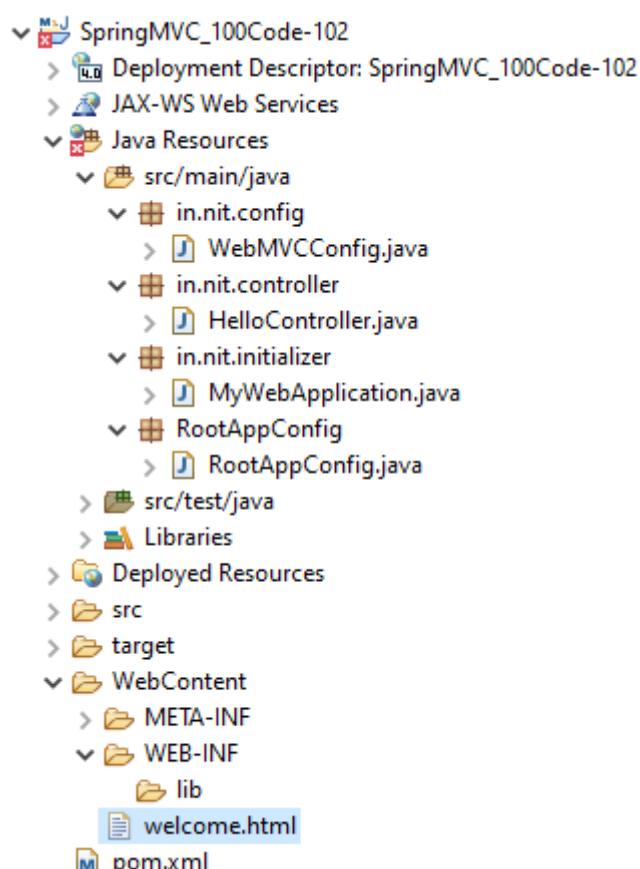
Output:

In Console:

Message From Welcome class

---

### Spring MVC Program:



Step1:

Pom.xml:

<properties>

```
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
 <dependency>
 <groupId>javax.servlet</groupId>
 <artifactId>javax.servlet-api</artifactId>
 <version>4.0.1</version>
 <scope>provided</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
 <dependency>
 <groupId>org.springframework</groupId>
 <artifactId>spring-webmvc</artifactId>
 <version>5.3.4</version>
 </dependency>
 <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
 <dependency>
 <groupId>javax.servlet</groupId>
 <artifactId>jstl</artifactId>
 <version>1.2</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.springframework/spring-context-support -->
 <dependency>
 <groupId>org.springframework</groupId>
 <artifactId>spring-context-support</artifactId>
 <version>5.3.2</version>
 </dependency>
```

</dependencies>

Step2:

Config:

```
package in.nit.config;
```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.ViewResolver;
import
org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.view.InternalResourceViewResolver;
```

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "in.nit.controller")
public class WebMVCCConfig {
```

```
//ViewResolver Configuration
@Bean
public ViewResolver createIVResolver(){
 System.out.println("createIVResolver()");
 InternalResourceViewResolver ivr=null;
 ivr=new InternalResourceViewResolver();
 ivr.setPrefix("/");
 ivr.setSuffix(".html");
 return ivr;
}
```

Step3:

rootAppConfig:

```
package Root AppConfig;

public class Root AppConfig {
```

Step4:

Controller:

```
package in.nit.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
@Controller
public class HelloController {
 @GetMapping("/w")
 public String showHomePage(){
 return "welcome";
 }
}
```

Step5:

Initializer:

```
package in.nit.initializer;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration;

import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.ContextLoaderListener;
import
org.springframework.web.context.support.AnnotationConfigWebApplicati
onContext;
import org.springframework.web.servlet.DispatcherServlet;

import RootAppConfig.RootAppConfig;
import in.nit.config.WebMVCConfig;

public class MyWebApplication implements WebApplicationInitializer {

 @Override
 public void onStartup(ServletContext sc) throws ServletException {
 System.out.println("onStarup(-)");
 AnnotationConfigWebApplicationContext
rootCtx=null,webCtx=null;
 ContextLoaderListener listener=null;
 DispatcherServlet ds=null;
 ServletRegistration.Dynamic registration=null;
 }
}
```

```
//create IOC containers having Configuration classes
rootCtx=new AnnotationConfigWebApplicationContext();
rootCtx.register(RootAppConfig.class);
webCtx=new AnnotationConfigWebApplicationContext();
webCtx.register(WebMVCConfig.class);
//create and register ContextLoaderListener
listener=new ContextLoaderListener(rootCtx);
sc.addListener(listener);
//create and DispatcherServlet
ds=new DispatcherServlet(webCtx);
registration=sc.addServlet("dispatcher",ds);
registration.addMapping("/w");
registration.setLoadOnStartup(1);
}//onStartup

}
```

Step6:

```
<h1 style="color:red;text-align:center">Welcome to 100% Code based
Spring Web MVC</h1>
```

Output:

[http://localhost:3038/SpringMVC\\_100Code-102/w](http://localhost:3038/SpringMVC_100Code-102/w)

---

Welcome to 100% Code based Spring Web MVC

Mini Project2:

IntegrationOf\_SpringMVC\_Hibernate\_WithReSTAPI-ConsumerApp-99

1.Pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
```

```
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc --
>
<dependency>
 <groupId>org.springframework</groupId>
 <artifactId>spring-webmvc</artifactId>
 <version>5.2.10.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
 <groupId>jstl</groupId>
 <artifactId>jstl</artifactId>
 <version>1.2</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.18.18</version>
 <scope>provided</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
databind -->
<dependency>
 <groupId>com.fasterxml.jackson.core</groupId>
 <artifactId>jackson-databind</artifactId>
 <version>2.11.0</version>
</dependency>

</dependencies>
```

## 2.AppConfig:

**package** in.nit.config;

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;
import
org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.view.InternalResourceViewResolver;
```

```
@EnableWebMvc
@Configuration
@ComponentScan("in.nit")//Package Name
public class AppConfig {
 @Bean
 public RestTemplate rt() {
 return new RestTemplate();
 }
 public InternalResourceViewResolver ivr() {
 return new InternalResourceViewResolver("/WEB-
INF/views", ".jsp");
 }
}
```

3.init:

```
package in.nit.init;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDisp
atcherServletInitializer;

import in.nit.config.AppConfig;

public class AppInt extends
AbstractAnnotationConfigDispatcherServletInitializer {

 @Override
 public Class<?>[] getRootConfigClasses() {

 return null;
 }
}
```

```
@Override
public Class<?>[] getServletConfigClasses() {

 return new Class[] { AppConfig.class };
}

@Override
public String[] getServletMappings() {

 return new String[] { "/" };
}

}
```

#### 4.model:Product

```
package in.nit.model;

import lombok.Data;

@Data
public class Product {

 private Integer proId;
 private String prodCode;
 private Double prodCost;
 private Double prodDiscount;
 private Double prodGst;
}
```

#### 5.Consumer:ProducerRestConsumer:

```
package in.nit.consumer;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
```

```
import org.springframework.web.client.RestTemplate;

import in.nit.model.Product;
@Component
public class ProductRestConsumer {
 @Autowired
 private RestTemplate template;
 public String saveProduct(Product product) {

 String url="http://localhost:3038/IntegrationOfHibernateWithReSTAPI-
ProducerApp99/rest/product/save";

 //Constructing Header Param
 HttpHeaders headers=new HttpHeaders();
 headers.setContentType(MediaType.APPLICATION_JSON);

 //Constructing head+body.
 HttpEntity<Product> request=new
 HttpEntity<Product>(product,headers);

 //Make http call, that gives response
 ResponseEntity<String> resp=template.postForEntity(url,
request,String.class);

 //Return message to UI
 return resp.getBody();
 }
}
```

## 6.IProductService

```
package in.nit.service;

import in.nit.model.Product;

public interface IProductService {
 public String saveProduct(Product product);
}
```

## 7.ProductServiceImpl:

```
package in.nit.service.impl;
```

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

import in.nit.consumer.ProductRestConsumer;
import in.nit.model.Product;
import in.nit.service.IProdcutService;
@Service
public class ProductServiceImpl implements IProdcutService {

 @Autowired
 private ProductRestConsumer consumer;

 @Override
 public String saveProduct(Product product) {

 return consumer.saveProduct(product);
 }
}
```

#### 8.controller:

```
package in.nit.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotationModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import in.nit.model.Product;
import in.nit.service.IProdcutService;

@Controller
public class ProductController {
 @Autowired
 private IProdcutService service;

 @RequestMapping(value = {"/reg", "/"})
 public String showReg() {
 return "Register";
 }
}
```

```
}
```

```
@RequestMapping(value = "/save" ,method = RequestMethod.POST)
public String saveProd(
 @ModelAttribute Product product,//Form Data Input
 Model model//Send data back to UI.
) {

 String msg=service.saveProduct(product);
 model.addAttribute("message",msg);
 return "Register";
}
}
```

9 Ui pages:

Common.jsp:

```
<table>
 <tr>
 <td>Register</td>
 <td>View All</td>
 </tr>
</table>
```

Register.jsp:

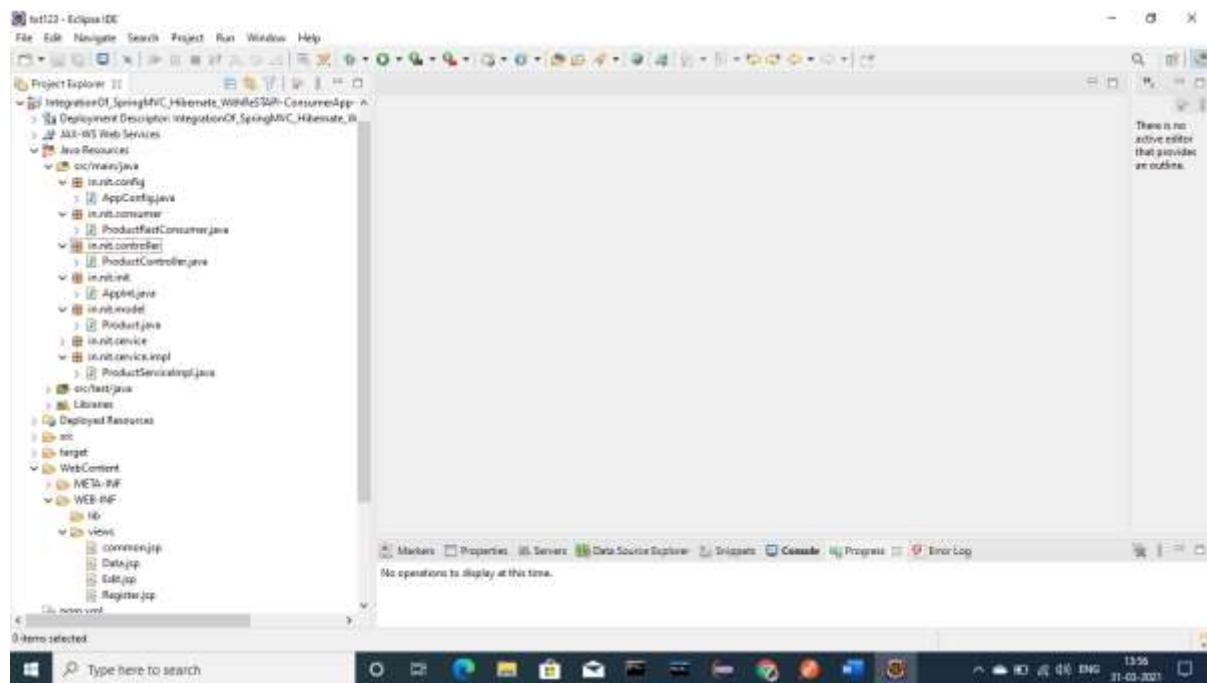
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"
 isELIgnored="false"
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="common.jsp" %>
<h2>Product Register Page</h2>
<form action="save" method="post">
<pre>
```

```
Product Code : <input type="text" name="prodCode"/>
Product Cost : <input type="text" name="prodCost"/>
 <input type="submit" value="Register"/>
</pre>
</form>
${message}
</body>
</html>
```

---

Check save operation output.

Mini Project2: Consumer App full code:



1.pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<artifactId>junit</artifactId>
<version>4.11</version>
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc --
>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>5.2.10.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
<groupId>jstl</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.18</version>
<scope>provided</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
databind -->
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.11.0</version>
</dependency>

</dependencies>
```

2.config:

```
package in.nit.config;

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

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;
import
org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.view.InternalResourceViewResolver;
```

```
@EnableWebMvc
@Configuration
@ComponentScan("in.nit")//Package Name
public class AppConfig {
 @Bean
 public RestTemplate rt() {
 return new RestTemplate();
 }
 public InternalResourceViewResolver ivr() {
 return new InternalResourceViewResolver("/WEB-
INF/views",".jsp");
 }
}
```

3.consumes:

```
package in.nit.consumer;
```

```
import java.util.Arrays;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;
```

```
import in.nit.model.Product;
@Component
public class ProductRestConsumer {
 @Autowired
```

```
private RestTemplate template;
public String saveProduct(Product product) {

 String
url="http://localhost:3032/IntegrationOfHibernateWithReSTAPI-
ProducerApp99/rest/product/save";

 //Constructing Header Param
 HttpHeaders headers=new HttpHeaders();
 headers.setContentType(MediaType.APPLICATION_JSON);

 //Constructing head+body.
 HttpEntity<Product> request=new
HttpEntity<Product>(product,headers);

 //Make http call, that gives response
 ResponseEntity<String> resp=template.postForEntity(url,
request,String.class);

 //Return message to UI
 return resp.getBody();
}

public List<Product> fetchData() {
 String
url="http://localhost:2032/Jersey2AdvHibernateProducerApp/rest/product
/all";
 ResponseEntity<Product[]> resp=template.getForEntity(url,
Product[].class);
 return Arrays.asList(resp.getBody());
}

public void removeData(Integer id) {
 String
url="http://localhost:2032/Jersey2AdvHibernateProducerApp/rest/product
/remove/"+id;
 template.delete(url);
}

public Product getOneProduct(Integer id) {
 String
url="http://localhost:2032/Jersey2AdvHibernateProducerApp/rest/product
/one/"+id;
```

```
 return template.getForObject(url, Product.class);
 }

 public void updateProduct(Product product) {
 String url="http://localhost:2032/Jersey2AdvHibernateProducerApp/rest/product/
/modify";
 //Constructing Header param
 HttpHeaders headers=new HttpHeaders();
 headers.setContentType(MediaType.APPLICATION_JSON);

 //constructing head + body
 HttpEntity<Product> request=new
 HttpEntity<Product>(product, headers);

 template.put(url, request);
 }
}
4.controller:

package in.nit.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotationModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import in.nit.model.Product;
import in.nit.service.IProdcutService;

@Controller
public class ProductController {
 @Autowired
 private IProdcutService service;

 @RequestMapping(value = {"/reg","/"})
 public String showReg() {
 return "Register";
 }
```

```
}
```

```
@RequestMapping(value ="/save" ,method = RequestMethod.POST)
public String saveProd(
 @ModelAttribute Product product,//Form Data Input
 Model model//Send data back to UI.
) {

 String msg=service.saveProduct(product);
 model.addAttribute("message",msg);
 return "Register";
}
```

```
@RequestMapping("/all")
public String viewData(Model model) {
 List<Product> list=service.fetchData();
 model.addAttribute("list", list);
 return "Data";
}
```

```
@RequestMapping("/remove")
public String remove(
 @RequestParam("id")Integer id,
 Model model
)
{
 service.removeData(id);
 model.addAttribute("message", "Product "+id+" Removed");
 List<Product> list=service.fetchData();
 model.addAttribute("list", list);
 return "Data";
}
```

```
@RequestMapping("/edit")
public String showEdit(
 @RequestParam("id")Integer id,
 Model model
)
{
 Product p=service.getOneProduct(id);
 model.addAttribute("ob", p);
 return "Edit";
}
```

```
@RequestMapping(value = "update",method =
RequestMethod.POST)
public String update(
 @ModelAttribute Product product,
 Model model)
{
 service.updateProduct(product);
 model.addAttribute("message", "Product
"+product.getProdId()+" Updated");
 List<Product> list=service.fetchData();
 model.addAttribute("list", list);
 return "Data";
}
}

5.init:

package in.nit.init;

import
org.springframework.web.servlet.support.AbstractAnnotationConfi
gDispatcherServletInitializer;

import in.nit.config.AppConfig;

public class Applnt extends
AbstractAnnotationConfigDispatcherServletInitializer {

 @Override
 public Class<?>[] getRootConfigClasses() {

 return null;
 }

 @Override
 public Class<?>[] getServletConfigClasses() {

 return new Class[] {AppConfig.class};
 }

 @Override
 public String[] getServletMappings() {

 return new String[] {"/"};
 }
}
```

```
}
```

```
}
```

6.model:

```
package in.nit.model;
```

```
import lombok.Data;
```

```
@Data
```

```
public class Product {
```

```
 private Integer proId;
 private String prodCode;
 private Double prodCost;
 private Double prodDiscount;
 private Double prodGst;
```

```
}
```

7.service:

```
package in.nit.service;
```

```
import java.util.List;
```

```
import in.nit.model.Product;
```

```
public interface IProdcutService {
 public String saveProduct(Product product);
 public List<Product> fetchData();
 public void removeData(Integer id);
 public Product getOneProduct(Integer id);
 public void updateProduct(Product product) ;
}
```

8.serviceml:

```
package in.nit.service.impl;
```

```
import java.util.List;
```

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

```
import in.nit.consumer.ProductRestConsumer;
import in.nit.model.Product;
import in.nit.service.IProdcutService;
@Service
public class ProductServiceImpl implements IProdcutService {

 @Autowired
 private ProductRestConsumer consumer;

 @Override
 public String saveProduct(Product product) {
 return consumer.saveProduct(product);
 }

 @Override
 public List<Product> fetchData() {
 return consumer.fetchData();
 }
 @Override
 public void removeData(Integer id) {
 consumer.removeData(id);
 }

 @Override
 public Product getOneProduct(Integer id) {
 return consumer.getOneProduct(id);
 }
 @Override
 public void updateProduct(Product porduct) {
 consumer.updateProduct(porduct);
 }

}
```

9.ui pages:

Common.jsp:

```
<table>
<tr>
 <td>Register</td>
```

```
<td>View All</td>
</tr>
</table>
```

Data.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"
isELIgnored="false"
%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="common.jsp" %>
<h3>Products Data Page</h3>
<table border="1">
<tr>
<th>ID</th>
<th>CODE</th>
<th>COST</th>
<th>DISCOUNT</th>
<th>GST</th>
<th colspan="2">OPERATIONS</th>
</tr>
<c:forEach items="${list}" var="ob">
<tr>
<td>${ob.prodId}</td>
<td>${ob.prodCode}</td>
<td>${ob.prodCost}</td>
<td>${ob.prodDiscount}</td>
<td>${ob.prodGst}</td>
<td>DELETE</td>
<td>EDIT</td>
</tr>
</c:forEach>
</table>
```

```
 ${message }
</body>
</html>
```

### Edit.jsp:

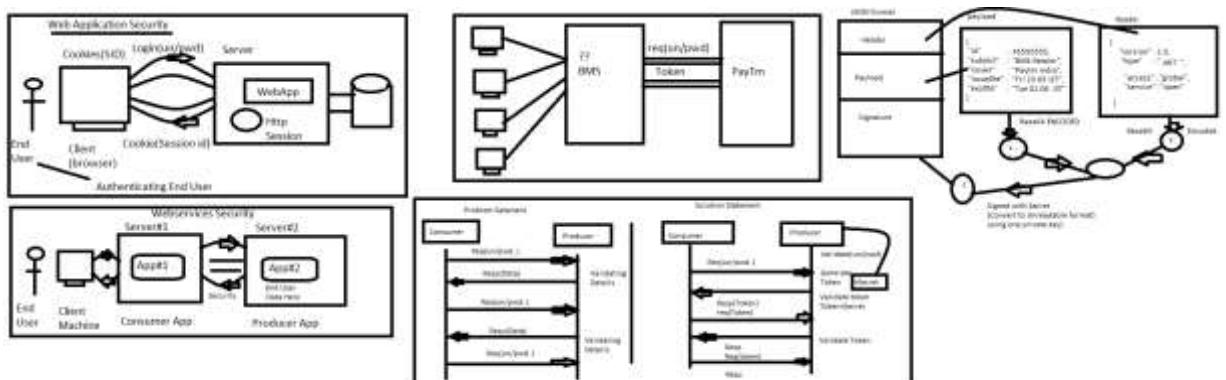
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"
 isELIgnored="false"
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="common.jsp" %>
<h2>Product Edit Page</h2>
<form action="update" method="post">
<pre>
Product Id : <input type="text" name="prodId" value="${ob.prodId}"
readonly="readonly">
Product Code : <input type="text" name="prodCode"
value="${ob.prodCode}"/>
Product Cost : <input type="text" name="prodCost"
value="${ob.prodCost}"/>
 <input type="submit" value="Update"/>
</pre>
</form>
</body>
</html>
```

### Register.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"
 isELIgnored="false"
%>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="common.jsp" %>
<h2>Product Register Page</h2>
<form action="save" method="post">
<pre>
Product Code : <input type="text" name="prodCode"/>
Product Cost : <input type="text" name="prodCost"/>
 <input type="submit" value="Register"/>
</pre>
</form>
${message}
</body>
</html>
```

Use post man tool, you can check outputs...



## Token Based Authorization:

- This concept is used between two different applications. Not between EndUser and App.
- Consider PayTm is supporting multiple clients like BookMyShow, ICIC Bank, Google Pay link, ....etc.

- When Client Application is trying to access Producer Service on behalf of EndUser then it must be Validated(Authorization) before processing request by using concept like Filters.
  - We can define one Filter which gets executed for every Request and it will check Consumer App details(Un/Pwd) for every request. But this process is may not be good(works fine). In this process it has to make network call with DB to validate every time Consumer with all Roles etc... (Validation process is lengthy).
  - Instead of validating Consumer for every request, Just validate one time and provide one unique Number back to Consumer by generating details at Producer. This unique Number(Token) is give to Producer by Consumer for 2<sup>nd</sup> request onwards...If token is valid then Producer will give Service else not. Such process is called as token Based Authorization.

# -----JWT(JSON Web Token)-----

- ➔ JWT – JSON Web Token
  - ➔ It is used to generate one token for Token Based Authorization.
  - ➔ It is Open Source used in any type of Http Supporting Application.  
Ex: java, .net, PHP, Angular .....etc....
  - ➔ Here, we are using JWT for Generating a Token at Producer and Validating Token.
  - ➔ HMAC = HashBased Message Authentication Code ie., in simple words A Unique number is generated and Converted to unreadable format (Encoded) using one Secret key(Private Key) known as Signature.
  - ➔ To Consumer application we have to send only Token not Secret key.
  - ➔ If Consumer sends next request with Token then it must be verified using Secret Key.
  - ➔ Secret Key behaves like a Password to read Token. It can be any Text.
  - ➔ JWT looks like:  
➔ Xxxxxxxxxxxxxx.yyyyyyyyyyyyyy.zzzzzzzzzzzzzzzz  
(Header)                   (Payload)                   (Signature)
  - ➔ Header contains details of JWT, version of JWT, Service Type, Service Mode,...etc.. which can not given by Producer or Consumer.
  - ➔ Payload contains information like
    - Id       = Consumer identity
    - Subject = Consumer name/username
    - Issuer   = Producer Data
    - Issued Date = Date of generation Token

- Exp Date = How Long token is valid.
- Sign = Secure with Password[Base64 Encoded(Header)+Base64 Encoded(PayLoad)].
- Sign can be read back using only secure password (Signkey/Secret key).

\*\*\*\*\*Code#1 Generate Token\*\*\*\*\*

Java JWT(JJWT):

- JJWT is a open source Java API used to work on JWT Tokens CREATION and READING.
- In application you must have added below dependency:

```
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-api -->
```

```
<dependency>
 <groupId>io.jsonwebtoken</groupId>
 <artifactId>jjwt</artifactId>
 <version>0.9.1</version>
</dependency>
```

--Ex--

eyJhbGciOiJIUzI1NiJ9

eyJqdGkiOiI0MDE0MkFBLUJNODgiLCJzdWliOiJCTVMgUmV0YWlsZXIiLCJpc3MiOiJQYXIUbSBJbmQuliwiaWF0ljoxNTkwODM3MDk1LCJleHAiOjE1OTA4MzcOTV9

ksc1iijnP0t6lsM7w9jswc4XD6P\_ItXA3z8Laa1gc4

-----Example Generating one Token-----

In pom.xml file:

```
<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
<dependencies>
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
</dependency>
```

```
<dependency>
 <groupId>io.jsonwebtoken</groupId>
 <artifactId>jjwt</artifactId>
 <version>0.9.1</version>
</dependency>

<dependency>
 <groupId>javax.xml.bind</groupId>
 <artifactId>jaxb-api</artifactId>
 <version>2.3.0</version>
</dependency> </dependencies>
```

-----Test class Code-----

```
package in.nit.test;
```

```
import java.util.Base64;
import java.util.Date;
import java.util.concurrent.TimeUnit;
```

```
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
```

```
public class Test
{
 public static void main(String[] args)
 {
 //Minutes to Mill sec
 System.out.println(TimeUnit.MINUTES.toMillis(1));
 //Hrs to Mill sec
 System.out.println(TimeUnit.HOURS.toMillis(1));
 //Day to Mill sec
 System.out.println(TimeUnit.DAYS.toMillis(1));
 //-----
 String secret="NIT-Service-2021-RA";
```

```
//-----CREATING JWT TOKEN
```

```
String token=
 Jwts.builder()
```

```
.setId("40142AA-BM88") //consumer Id
.setSubject("BMS Retailer") //consumer name
.setIssuer("PayTm Ind.") //token Provider name
.setIssuedAt(new Date(System.currentTimeMillis())) //Token
creation date
.setExpiration(new Date(System.currentTimeMillis() +
TimeUnit.MINUTES.toMillis(5))) //exp Date
.signInWith(SignatureAlgorithm.HS256,
Base64.getEncoder().encode(secret.getBytes()))
.compact();
System.out.println(token);
}
}
Output:
60000
3600000
86400000
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI0MDE0MkFBLUJNODgiLCJzdWliOiJ
CTVMgUmV0YWlsZXIiLCJpc3MiOiJQYXIUbSBJbmQuliwiaWF0ljoxNjE3
MjAyMTA0LCJleHAiOjE2MTcyMDI0MDR9.ye4V_PpXEWIr7_QgCg2Pja
49bI4912O6r6uPjxaAjfk
```

#### -----Code#2 Validate Token-----

Claims: It means reading token data by providing token and Secret. This process also called as Parsing. We need to read Body(Payload details).

```
package in.nit.test;

import java.util.Base64;
import java.util.Date;
import java.util.concurrent.TimeUnit;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.JwtException;
import io.jsonwebtoken.SignatureAlgorithm;

public class ClaimsTest
{
 public static void main(String[] args) {
```

```
//-----
String secret="NIT-Service-2021-RA";

//-----CREATING JWT TOKEN

String token=
 Jwts.builder()
 .setId("40142AA-BM88") //consumer Id
 .setSubject("BMS Retailer") //consumer name
 .setIssuer("PayTm Ind.") //token Provider name
 .setIssuedAt(new Date(System.currentTimeMillis())) //Token
creation date
 .setExpiration(new Date(System.currentTimeMillis() +
TimeUnit.MINUTES.toMillis(5))) //exp Date
 .signWith(SignatureAlgorithm.HS256,
Base64.getEncoder().encode(secret.getBytes()))
 .compact();
System.out.println(token);

Claims c=Jwts.parser()

.setSigningKey(Base64.getEncoder().encode(secret.getBytes()))
 .parseClaimsJws(token)
 .getBody();
System.out.println(c.getId());
System.out.println(c.getSubject());
System.out.println(c.getIssuer());
System.out.println(c.getIssuedAt());
}
}
```

Output:

```
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI0MDE0MkFBLUJNODgiLCJzdWliOiJ
CTVMgUmV0YWlsZXIiLCJpc3MiOiJQYXIUbSBJbmQuIiwiaWF0IjoxNjE3
MjAyMjgxLCJleHAiOjE2MTcyMDI1ODF9.brzqpB9U4nDe6M4A1HluBJp
nLHbwjFdJbiaOeSmbfo
40142AA-BM88
BMS Retailer
PayTm Ind.
Wed Mar 31 20:21:21 IST 2021
```

JWTUtil class:

```
package in.nit.util;

import java.util.Base64;
import java.util.Date;
import java.util.concurrent.TimeUnit;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

public class JWTUtil {
 private static String secret="NIT-RA-20-sec";
 public static String generateToken(String id,String subject) {
 String token=
 Jwts.builder()
 .setId(id)
 .setSubject(subject)
 .setIssuer("Naresh I Technologies")
 .setIssuedAt(new Date(System.currentTimeMillis()))
 .setExpiration(new
Date(System.currentTimeMillis()+TimeUnit.MINUTES.toMillis(5)))

 .signWith(SignatureAlgorithm.HS256,Base64.getEncoder().encode
(secret.getBytes()))
 .compact();
 return token;
 }
 public static Claims getClaims(String token) {
 Claims c=
 Jwts.parser()

 .setSigningKey(Base64.getEncoder().encode(secret.getBytes()))
 .parseClaimsJws(token)
 .getBody();
 return c;
 }
 public static String getSubjects(String token) {

 return getClaims(token).getSubject();
 }

 public static String getId(String token) {
```

```
 return getClaims(token).getId();
 }
}
```

Test1 class:

```
package in.nit.test;

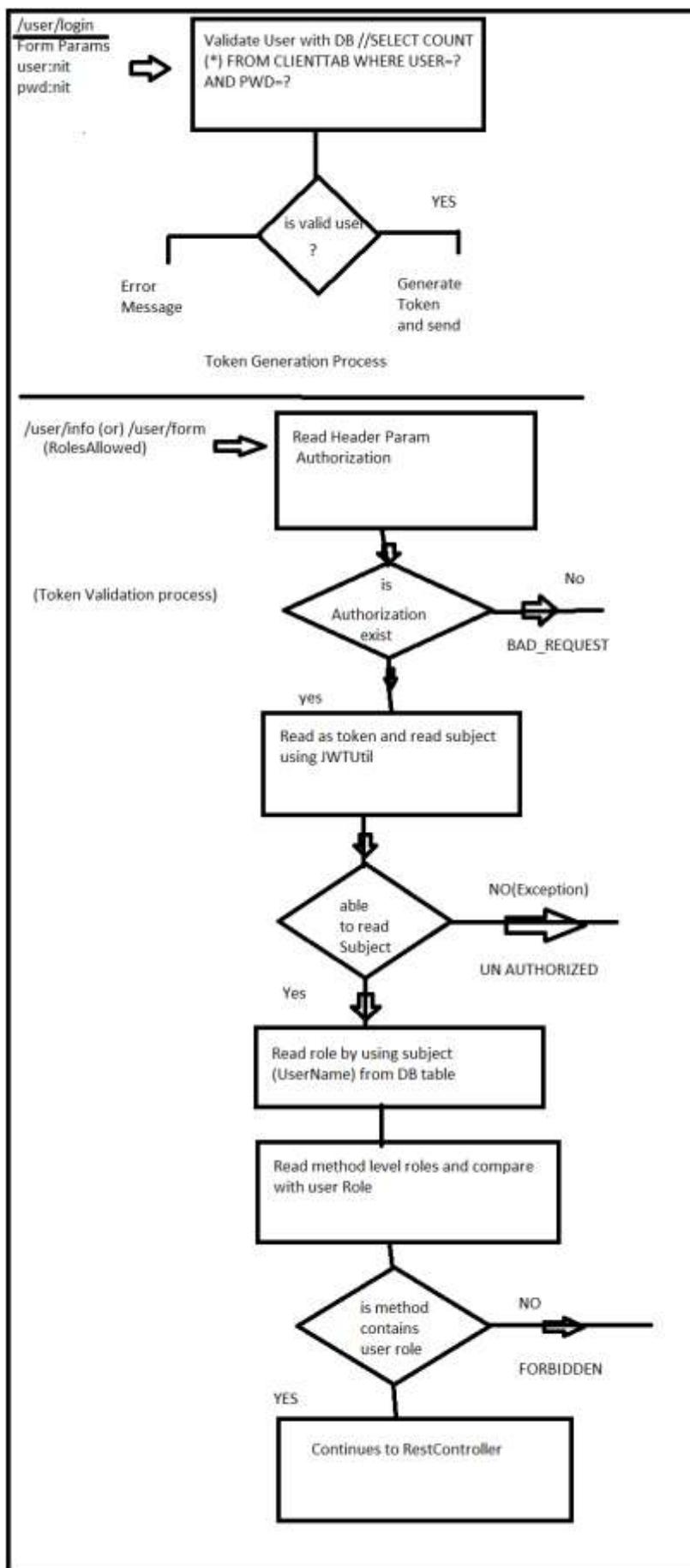
import in.nit.util.JWTUtil;

public class Test1 {
 public static void main(String[] args) {
 String id="qweee";
 String subject="web";
 String token=JWTUtil.generateToken(id, subject);
 System.out.println(token);
 System.out.println(JWTUtil.getSubjects(token));
 System.out.println(JWTUtil.getId(token));
 }
}
```

Output:

```
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJxd2VIZSIsInN1Yil6IndlYilsImlzcyI6Ik
5hcmVzaCBJIFRIY2hub2xvZ2llcyIsImhdCI6MTYxNzIwNDI0MywiZXhwIj
oxNjE3MjA0NTQzfQ.4oqnlyP7yOFVo_OVJw5UkWWsD7L9VNLC9o9Kx
x5NRb4
web
qweee
```

---



Solution1:# Add one method in RestController that can be accessed by every one.

If user/pwd given validate with DB.

If valid user

Generate token and send

Invalid user

Send Error Message

-----Code-----

```
@Path("/user")
public class UserRestController {
//---extra methods---
 @Path("/login")
 @POST
 @PermitAll
 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
 public String checkUserForToken(
 @FormParam("user")String user,
 @FormParam("pwd")String pwd
) {
 String data=null;
 //Validate User and Pwd with DB
 boolean exist=UserValidator.validateUser(user, pwd);
 if(exist) {// If User exist Generate Token
 data=JwtUtil.generateToken("ID: "+user, user);
 }
 else {//No Token Send Error Message
 data="Invalid User(\""+user+"\") details";
 }
 }
}
```

S#2: In Side request Filter, Read Token as header param

If null/Empty(No Token Header)

Reject with Error response (BAD\_REQUEST->No Token/Security Found).

-----code-----

```
if(authList==null || authList.isEmpty()) {
```

```
req.abortWith(
 Response.status(Status.BAD_REQUEST)
 .entity("No Security Details Found....")
 .build()
);
}
```

S#3: If Header Param is not null

Read as token, getSubject using JwtUtil, read roles from DB

-----Code-----

//Read Token from Header

```
String token=authList.get(0);
 //Read Subject from Token
String user=JwtUtil.getSubject(token);
String userRole=UserValidator.getRoleByUser(user);
```

S#4: Read method Level Roles:

-----Code-----

//Read method Level Roles at list

```
String mroles[]=method.getAnnotation(RolesAllowed.class).value();
List<String> methodRoles=Arrays.asList(mroles);
```

S#5: Check contains user role or not?

If contains continue to restController

Else reject with Error Response

-----Code-----

```
if(!methodRoles.contains(userRole)) {
 req.abortWith("//If Token Is not valid....
 Response.status(Status.FORBIDDEN)
 .entity("User can't Access this Method....")
 .build()
);
}
```

JWT(JSON Web Token), Validator, Filter::

```
mysql> use webservices;
Database changed
```

```
mysql> desc clienttab;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| cid | int | YES | | NULL | |
| csecr | varchar(30) | YES | | NULL | |
| crole | varchar(30) | YES | | NULL | |
+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

```
mysql> select * from clienttab;
+-----+-----+
| cid | csecr | crole |
+-----+-----+
| 2 | nit | user |
| 1 | sankar | 2 |
+-----+-----+
2 rows in set (0.00 sec)
```

1.pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-
codec/commons-codec -->
<dependency>
 <groupId>commons-codec</groupId>
 <artifactId>commons-codec</artifactId>
 <version>1.15</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>8.0.20</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
<dependency>
 <groupId>io.jsonwebtoken</groupId>
 <artifactId>jjwt</artifactId>
 <version>0.9.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api
-->
<dependency>
 <groupId>javax.xml.bind</groupId>
 <artifactId>jaxb-api</artifactId>
 <version>2.3.0</version>
</dependency>

</dependencies>
```

## 2.DBConn:

```
package in.nit.db;
```

```
import java.sql.Connection;
import java.sql.DriverManager;

public class DBConn {
 private static Connection conn=null;
 static {
 try {
 Class.forName("com.mysql.jdbc.Driver");

 conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/
webservices","root","root");
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
 public static Connection getConnection() {

 return conn;
 }
}
```

### 3.UserValidator:

```
package in.nit.validator;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

import in.nit.db.DBConn;

public class UserValidator {
 public static boolean validateUser(String un,String pwd) {
 boolean flag=false;
 String sql="SELECT COUNT(CID) FROM CLIENTTAB WHERE
CID=? AND CSECR=?";
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(sql);
 ps.setString(1, un);
 ps.setString(2, pwd);
 ResultSet rs=ps.executeQuery();
 rs.next();
 long count=rs.getLong(1);

```

```
 if(count>0) flag=true; //User Exist
 } catch (Exception e) {
 e.printStackTrace();
 }
 return flag;
}
public static String getRoleByUser(String un) {
 String role=null;
 String sql="SELECT CROLE FROM CLIENTTAB WHERE CID=?";
 try {
 PreparedStatement
ps=DBConn.getConnection().prepareStatement(sql);
 ps.setString(1, un);
 ResultSet rs=ps.executeQuery();
 rs.next();
 role=rs.getString(1);
 } catch (Exception e) {
 e.printStackTrace();
 }
 return role;
}
}
```

#### 4.JwtUtil:

```
package in.nit.util;

import java.util.Base64;
import java.util.Date;
import java.util.concurrent.TimeUnit;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

public class JwtUtil {
private static String secr="Nit-RA-20-SECR";
public static String generateToken(String id,String subject) {
 String token=
 Jwts.builder()
 .setId(id)
 .setSubject(subject)
 .setIssuer("NareshITechnologies")
```

```
.setIssuedAt(new Date(System.currentTimeMillis()))
.setExpiration(new
Date(System.currentTimeMillis() + TimeUnit.MINUTES.toMillis(1)))
.signWith(SignatureAlgorithm.HS256,
Base64.getEncoder().encode(secr.getBytes()))
.compact();
return token;
}
public static Claims getClaims(String token) {
Claims c=
Jwts.parser()

.setSigningKey(Base64.getEncoder().encode(secr.getBytes()))
.parseClaimsJws(token)
.getBody();
return c;
}
public static String getSubject(String token) {

return getClaims(token).getSubject();
}

public static String getId(String token) {

return getClaims(token).getId();
}
}
```

## 5. RestController:

```
package in.nit.controller;

import javax.annotation.security.DenyAll;
import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;
import javax.ws.rs.Consumes;
import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.MediaType;
```

```
import in.nit.util.JwtUtil;
import in.nit.validator.UserValidator;

@Path("/user")
public class UserRestController {
//---extra methods---
 @Path("/login")
 @POST
 @PermitAll
 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
 public String checkUserForToken(
 @FormParam("user")String user,
 @FormParam("pwd")String pwd
) {
 String data=null;
 //Validate User and Pwd with DB
 boolean exist=UserValidator.validateUser(user, pwd);
 if(exist) {// If User exist Generate Token
 data=JwtUtil.generateToken("ID: "+user, user);
 }
 else {//No Token Send Error Message
 data="Invalid User("+user+") details";
 }
 return data;
 }
 @Path("/all")
 @GET
 @PermitAll
 public String common() {

 return "Welcome To All Consumers!";
 }
 @Path("/none")
 @GET
 @DenyAll
 public String noMsg() {

 return "You can not see this!!";
 }
 @GET
 @Path("/info")
```

```
@RolesAllowed({"user"})
public String welcomeToService() {
 return "Welcome to user!! ";
}
@GET
@Path("/form")
@RolesAllowed({"2"})
public String welcomeToForm() {

 return " Welcome to 2!!!";
}
}
```

## 6. Security Filter:

```
package in.nit.filter;

import java.io.IOException;
import java.lang.reflect.Method;
import java.util.Arrays;
import java.util.List;

import javax.annotation.security.DenyAll;
import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.container.ResourceInfo;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.Status;

import in.nit.util.JwtUtil;
import in.nit.validator.UserValidator;

public class SecureJwtFilter implements ContainerRequestFilter {
 @Context
 private HttpHeaders headers;
 @Context
 private ResourceInfo resource;
 @Override
```

```
public void filter(ContainerRequestContext req) throws
IOException {
 Method method=resource.getResourceMethod();

 if(method.isAnnotationPresent(PermitAll.class)) {
 return;// Continue to Rest Controller.
 }
 if(method.isAnnotationPresent(DenyAll.class)) {
 req.abortWith(
 Response.status(Status.FORBIDDEN)
 .entity("No Access")
 .build()
);
 }
 if(method.isAnnotationPresent(RolesAllowed.class)) {
 //read Token as header Param
 List<String>
authList=headers.getRequestHeader("Authorization");
 if(authList==null || authList.isEmpty()) {
 req.abortWith(
 Response.status(Status.BAD_REQUEST)
 .entity("No Security Details Found....")
 .build()
);
 }
 try {
 //Read Token from Header
 String token=authList.get(0);
 //Read Subject from Token
 String user=JwtUtil.getSubject(token);
 String userRole=UserValidator.getRoleByUser(user);
 //Read method Level Roles at list
 String
mroles[]=method.getAnnotation(RolesAllowed.class).value();
 List<String> methodRoles=Arrays.asList(mroles);
 if(!methodRoles.contains(userRole)) {
 req.abortWith("//If Token Is not valid...."

Response.status(Status.FORBIDDEN)
 .entity("User can't Access this
Method....")
 .build()
);
 }
}
```

```
 }
 } catch (Exception e) {
 req.abortWith(
 Response.status(Status.UNAUTHORIZED)
 .entity("Invalid Token Found....")
 .build()
);
 e.printStackTrace();
 }
}

}
```

## 7. AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;
import org.glassfish.jersey.server.ResourceConfig;
import in.nit.filter.SecureJwtFilter;
import org.glassfish.jersey.server.ResourceConfig;
import in.nit.filter.SecureJwtFilter;
@ApplicationPath("/rest")
public class AppConfig extends ResourceConfig {
 public AppConfig() {
 packages("in.nit");
 register(SecureJwtFilter.class);
 }
}
```

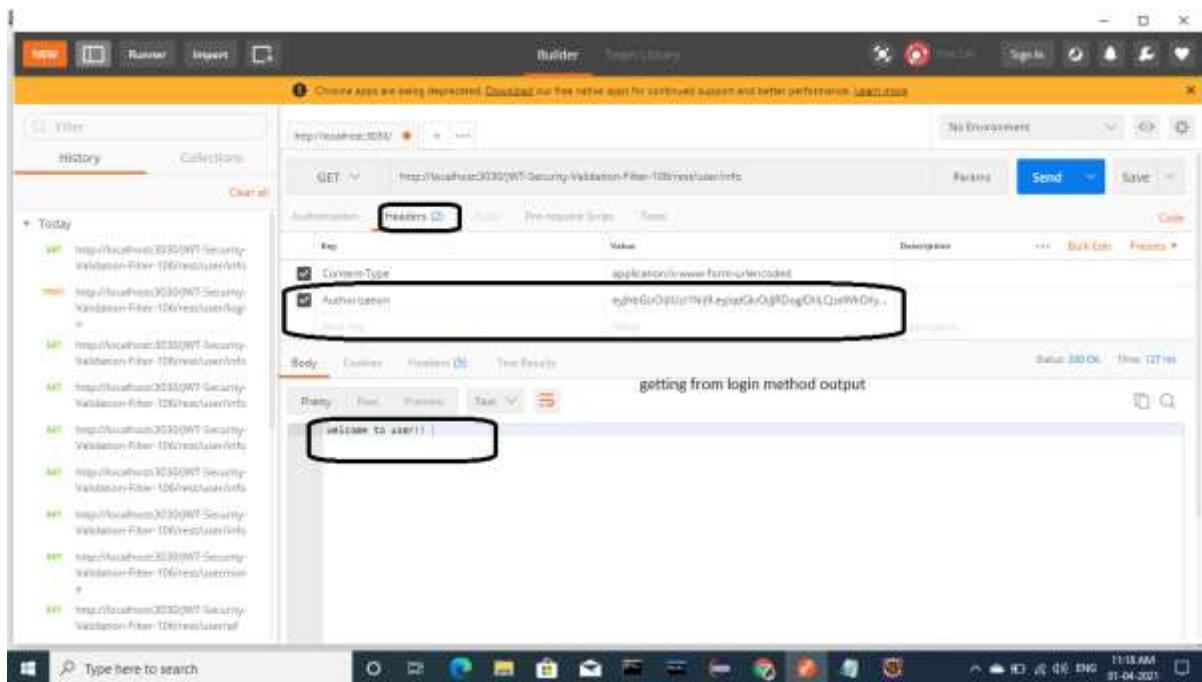
Output: Screens:

Postman Screens:

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:3030/WS-Security-Validation-Filter-10/test/user/login`. The request body is set to `application/x-www-form-urlencoded` and contains two key-value pairs: `uid` with value `2` and `pwd` with value `44`. The response status is `200 OK` with a time of `116 ms`.

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:3030/WS-Security-Validation-Filter-10/test/user/all`. The response status is `200 OK` with a time of `44 ms`. The response body displays the message `Welcome To All Consumers!`



## Headers in http(Adv. Java Concept)

Http (request/response) will have two parts, those are: Head and Body.

Head: This area contains data in key = value format. Also called as Header Params.

These params will execute “ A command or provides meta data of request/response”.

Example for Commands:

Clear cache(pragma), Go to URL after some time (Refresh), check info (if-match), etc....

Example for Meta Data:

Content-Type=text,  
Content-length=20kb,  
Date=-----  
Etc.....

Body:

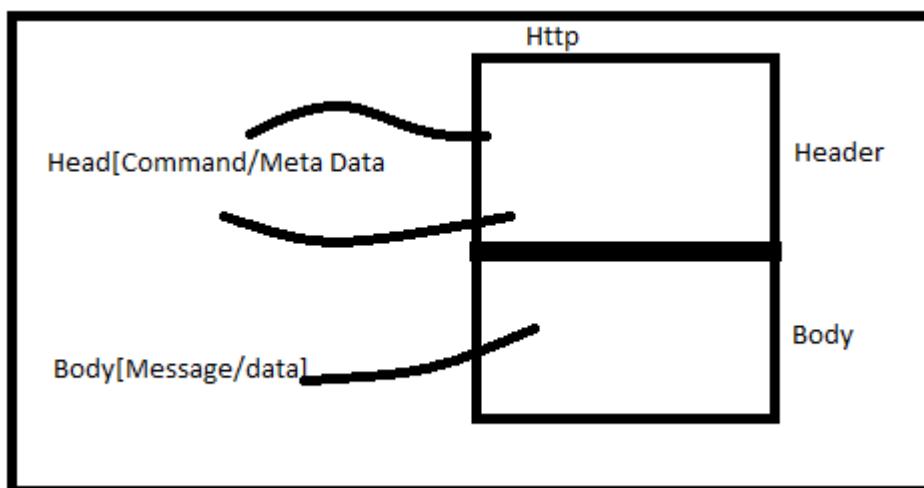
Http Body contains only data(message).

Header Parameters are two types: Those are:

1. Predefined.
2. Customer(Programmer defined).

Example for predefined Header Parameters:

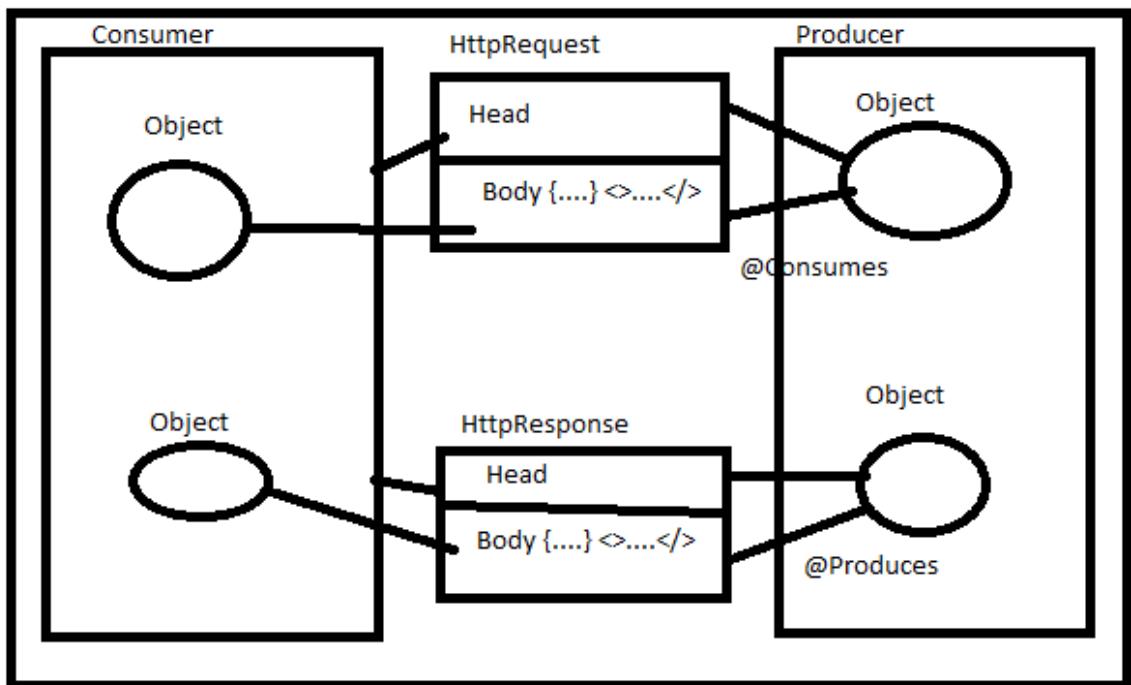
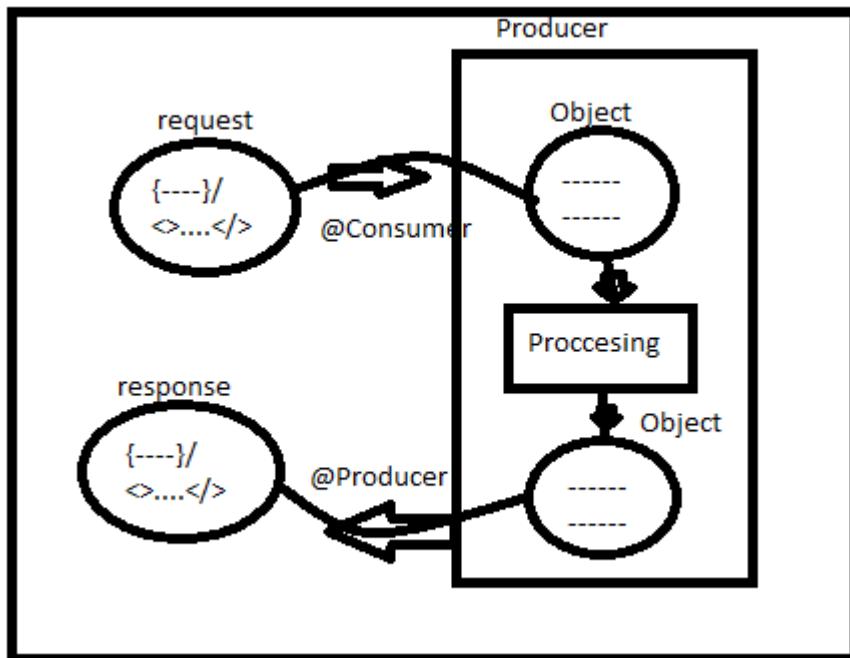
Refer below link for the list of Header Parameters.  
[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_header\\_fields](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields)



## MediaType Annotations:

@Consumes and @Produces are known as MedaiType Annotations. These are used to convert Global format Data to Object Object and Object to Global format.

Here Global formats are: XML,JSON



#### @Consumes:

It converts Global Data, taken from HttpServletRequest Body, into Object format. This Object is given as input to method(Parameter).

**@Produces:** It converts Object which is method return value(output) to Global format (XML/JSON) and placed into HttpServletResponse Body.

```
Example: @Consumes("application/json")
 @Produces("application/json")
 Public Model show(Product p){

 }
```

Here @Consumes reads JSON data from HttpRequest Body. This JSON data will be converted to Product class object and give as input to show(....) method.

Show(...) method executes method Body by taking Product Object as input as input and returns finally model class as Object.

Model class Object will be converted to JSON and placed into HttpResponse Body.

Here, @Consumes("application/json") can be written as  
@Consumes(MediaType.APPLICATION\_JSON)

MediaType is a class given from "javax.ws.rs.core" package and APPLICATION\_JSON is a static property which has internal value ="application/json".

@Consumes("application/xml" is equals to  
@Consumes(MediaType.APPLICATION\_XML)

Every class Object can be converted to JSON, can be converted to any class Object.

Every class Object cannot be converted to XML format. Only JAXB(Java Architecture for XML Binding) class Object can be converted to XML.

To convert normal class to JAXB class, apply @XmlRootElement on top of class.

@Consumes:

It is used to convert HttpRequest Data (JSON/XML) into Object format and give to method parameter (input to method).

Along with data, we must specify one Header "Content-type:-----"

In HttpRequest Head part, else Provider throw HTTP-415  
(UnSupported MediaType)

i.e.,expected input(ex.) JSON, but received one is XML(ex.) text.

## @Context Data in ReST:

This annotation is used to fetch Objects created at server memory from Producer classes. It is applicable for special type of classes/interfaces (Not applicable for all classes/interfaces).

Example:

```
HttpServletRequest
ServletContext ,
ServletConfig,
HttpHeaders,
UriInfo,
ResourceInfo etc....
```

Syntax:

```
@Context className ObjName
@Context interface ObjName
```

It will load Object data from server, to read in method.

## Role Management in ReST with SecurityFilter:

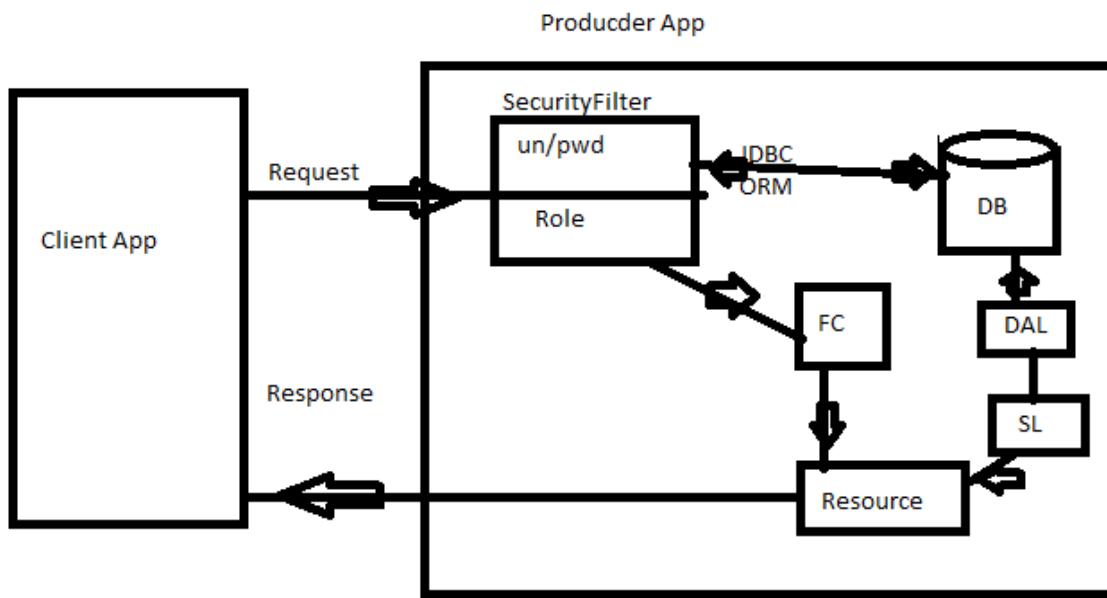
ReST supports SecurityFilter implementation for all Resource classes ( and their methods). It will check username,password along with Role for requested client.

If everything is valid then Request will be send to FrontController (ServletContainer). It will Execute Resource method based on Path which may communicate to database using Service layer and Data Access Layer. At Last, returns Response with Entity and Status.

Annotations used for Role management are:

1. @PermitAll:-- Allows every request.

2. @DenyAll:---- No request is allowed
3. @RolesAllowed:----- Only few requests are allowed.



## Securing ReST Endpoints:

1. Define one class with any Name  
 Example: SecurityFilter which implements interface "ContainerRequestFilter" given by Jersey2.x.
2. Override method filter(.....)
3. Define logic in below way:

If-method-has-annotation:

→PermitAll : No Checking & Continue same  
 →DenyAll : No checking & Reject(403:FORBIDDEN)  
 →RolesAllowed : Check un,pwd and Role.  
     If empty : 400 (BAD\_REQUEST)  
     If Invalid : 401(AUAUTHORIZED)  
     Else : continue request.

4. Define Resources and Config classes for application.
5. Apply Annotation @PermitAll/@DenyAll/@RolesAllowed over methods.
6. Enable Filter class using
  - a. Add @Produces on top of Filter class, provide package name in Config class.
  - b. Use register(T.class) in Config class along with package(....).

## WADL [Web Application Description Language]

This document will be generated by ReST Service producer Application and given to Client Application.

**EndPoints:** It is an information of one service(web service Producer class) which contains details like “URL, Http Method Type, Input and Output MediaType, Parameters, Roles, Security Headers,...etc.

**WADL:** It is a XML File which Provides details of endpoint of a service

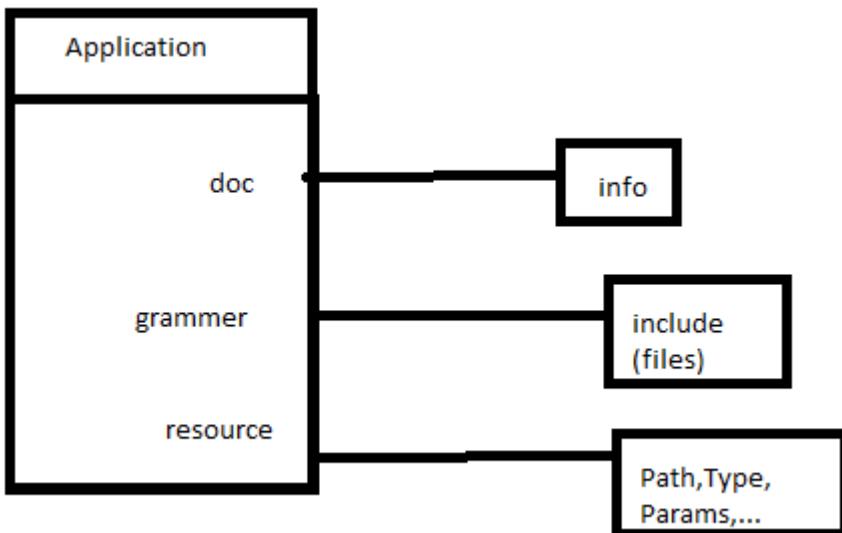
WADL will be generated by Service Producer which contains URL (Path), Input and Output, MediaTypes, Parameters details,MethodTypes(GET/POST), ReturnType Details, but not the logic.

Based on WADL, Client can do coding and makes the request.

**API:** Application Programming Interface, It is a document which provides information of Java code(Class/ENUM/@annotation/interface), Basic API type is HTML.

Generating of WADL[Creation]:

1. Define one Jersey Producer application with FC and Resource (Producer class).
  2. Start application and test using PostMan.
  3. Enter URL upto FC and append/application.wadl, then press enter.
  4. Root is application having details od doc, grammer, resources.
  5. `http:<IP>:<PORT>/<AppName>/<FCUrl>/application.wadl`.
- Example: `http:localhost:3032:/Producer/rest/application.wadl`.



WADL Contains Root <application> tag, it has 3 child.

- 1.<doc/>:-It contains information of document generator. That is “who Generated this document?” and their links (URL).
- 2.<grammer/>:- It includes files like XSD (XML Schema Design )which gives XML format in case of XML MediaType.
3. <resource/>:-It provides details like
  - a) base and path
  - b)method id,name
  - c)request and response.

Here, base = common URL of Producer  
 Path =Path at class level and method level,  
 Method id =method name in code,  
 Method Name =method Type in (GET/POST).  
 Request =Input Params and MediaTypes for Consumer  
 Response =Output(returnType)MediaTypes for Produces.

Overview of WADL document:

```

<application.....>
<doc.....>....</>
<grammers>
<include href="....xsd.xsd">
</include>
</grammers>

<resource base="URLUptoFrontController">
<resource path="ClassPath">

```

```
<resource path=" method path">
 <method id="methodName" name="Type"/>
 <request>param/input Type</request>
 <request>output Type</request>
 <method>
</resource>
</resource>

</application>
```

---

## Interceptors:

- Model Similar to Filter.
- Used to manipulate entities (input and output Streams).
- Interceptors are:
  1. ReaderInterceptor
  2. WriterInterceptor

| Interceptors                                                                                                                                                                  | Filters                                                                                                                                                                                         |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>1.Used to manipulate entities( input , output streams).</p> <p>2. two types:</p> <p>a.ReaderInterceptor<br/>b.WriterInterceptor</p> <p>ex: Encoding an entity response</p> | <p>1. Used to manipulate request and response params (headers,URLs etc...).</p> <p>2.two types</p> <p>a) ContainerRequestFilter<br/>b)ContainerResponseFilter</p> <p>ex: Logging, Security.</p> |

### Example of Interceptor:

#### 1.pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

## 2.Filters:

```
package in.nit.filters;

import java.io.IOException;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;

public class MyRequestFilter implements ContainerRequestFilter{

 @Override
 public void filter(ContainerRequestContext requestContext) throws
 IOException {
 System.out.println("FROM REQUEST FILTER....");
 }
}
```

```
}
```

```
package in.nit.filters;
```

```
import java.io.IOException;
```

```
import javax.ws.rs.container.ContainerRequestContext;
```

```
import javax.ws.rs.container.ContainerResponseContext;
```

```
import javax.ws.rs.container.ContainerResponseFilter;
```

```
public class MyResponseFilter implements ContainerResponseFilter{
```

```
 @Override
```

```
 public void filter(ContainerRequestContext requestContext,
```

```
ContainerResponseContext responseContext)
```

```
 throws IOException {
```

```
 System.out.println("FROM RESPONSE FILTER");
```

```
 }
```

```
}
```

### 3. Interceptors:

```
package in.nit.interceptor;
```

```
import java.io.BufferedReader;
```

```
import java.io.ByteArrayInputStream;
```

```
import java.io.IOException;
```

```
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
```

```
import java.util.stream.Collectors;
```

```
import javax.ws.rs.WebApplicationException;
```

```
import javax.ws.rs.ext.Provider;
```

```
import javax.ws.rs.ext.ReaderInterceptor;
```

```
import javax.ws.rs.ext.ReaderInterceptorContext;
```

```
@Provider
```

```
public class RequestReaderInterceptor implements
```

```
ReaderInterceptor {
```

```
 @Override
```

```
public Object aroundReadFrom(ReaderInterceptorContext context)
 throws IOException, WebApplicationException {
 InputStream is = context.getInputStream();
 String body = new BufferedReader(new
InputStreamReader(is)).lines()
 .collect(Collectors.joining("\n"));

 context.setInputStream(new ByteArrayInputStream(
 (body + " message added in server reader
interceptor").getBytes()));
 System.out.println(" reader interceptor.....");
 return context.proceed();
}
}

package in.nit.interceptor;

import java.io.IOException;

import javax.ws.rs.WebApplicationException;
import javax.ws.rs.ext.Provider;
import javax.ws.rs.ext.WriterInterceptor;
import javax.ws.rs.ext.WriterInterceptorContext;

@Provider
public class RequestWriterInterceptor implements WriterInterceptor {

 @Override
 public void aroundWriteTo(WriterInterceptorContext context)
 throws IOException, WebApplicationException {
 context.getOutputStream()
 .write(("Message added in the writer interceptor in the client
side\n").getBytes());
 System.out.println("writer interceptor");
 context.proceed();
 }
}
```

#### 4.controller:

```
package in.nit.controller;

import javax.ws.rs.GET;
```

```
import javax.ws.rs.Path;

@Path("/msg")
public class MessageRestController {
 @GET
 public String show() {

 System.out.println(" From Rest Controller");
 return "Hello From Advanced Config!";
 }
}

package in.nit.config;

import javax.ws.rs.ApplicationPath;
import org.glassfish.jersey.server.ResourceConfig;

import in.nit.filters.MyRequestFilter;
import in.nit.filters.MyResponseFilter;
import in.nit.interceptor.RequestReaderInterceptor;
import in.nit.interceptor.RequestWriterInterceptor;
@ApplicationPath("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC:
ServletContainer
 //Constructor
 public AppConfig() {
 packages("in.nit");//init-param

 //Activate Filters
 register(MyRequestFilter.class);
 register(MyResponseFilter.class);

 //Activate Interceptors
 register(RequestReaderInterceptor.class);
 register(RequestWriterInterceptor.class);
 }
}
```

---

Output:

[http://localhost:3032/Jersey\\_Filter\\_And\\_Interceptor-107/rest/msg](http://localhost:3032/Jersey_Filter_And_Interceptor-107/rest/msg)

Message added in the writer interceptor in the client side Hello From Advanced Config!

---

## Resource Scopes:

| Scope            | Annotation                  | Annotation full class name                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------|-----------------------------|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Request scope    | @RequestScoped<br>(or none) | org.glassfish.jersey.process.internal.RequestScoped | Default lifecycle (applied when no annotation is present). In this scope the resource instance is created for each new request and used for processing of this request. If the resource is used more than one time in the request processing, always the same instance will be used. This can happen when a resource is a sub resource and is returned more times during the matching. In this situation only one instance will serve the requests. |
| Per-lookup scope | @PerLookup                  | org.glassfish.hk2.api.PerLookup                     | In this scope the resource instance is created every time it is needed for the processing even it handles the same request.                                                                                                                                                                                                                                                                                                                         |
| Singleton        | @Singleton                  | javax.inject.Singleton                              | In this scope there is only one instance per jax-rs application. Singleton resource can be either annotated with @Singleton and its class can be registered using the instance of Application. You can also create singletons by registering singleton instances into Application.                                                                                                                                                                  |

## Bean Validations:

Is not working properly so use <version>2.33</version>

Use this version <version>2.30</version>

### 1.pom.xml:

```
<properties>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>
```

```
<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
```

[https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->](https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet-->)

```
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>

</dependencies>
```

## 2.web.xml:

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
 <display-name>Archetype Created Web Application</display-name>
 <servlet>
 <servlet-name>sample</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>in.nit.controller</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>sample</servlet-name>
 <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

## 3. Controller:

```
package in.nit.controller;
```

```
import javax.validation.constraints.Email;
import javax.validation.constraints.NotNull;
import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

@Path("/p")
public class ProducerController {

 @Path("/u")
 @POST
 public String message(
 @NotNull
 @FormParam("name")String name,
 @Email
 @FormParam("mail")String mail
) {
 return "hello: "+name+" "+mail;
 }
}
```

### Output:

The screenshot shows a REST client interface with the following details:

- Request URL:** POST [http://localhost:3033/jersey\\_form\\_param/Validation-108/test/p/u](http://localhost:3033/jersey_form_param/Validation-108/test/p/u)
- Method:** POST
- Body:** form-data
- Parameters:**
  - name: sankar
  - mail: sankar@gmail.com
- Response:** Status 200 OK | Time: 17 ms
- Response Body:** Hello: sankar sankar@gmail.com

## Jersey MVC:

### 1.pom.xml:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
 </dependency>
 <!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
 <dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
 </dependency>
 <dependency>
 <groupId>org.glassfish.jersey.ext</groupId>
 <artifactId>jersey-mvc-jsp</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

### 2.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.server.ResourceConfig;
@ApplicationPath("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC: ServletContainer
 //Constructor
 public AppConfig() {
 packages("in.nit");//init-param

 }
}
```

### 3.Controller:

```
package in.nit.controller;
```

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import org.glassfish.jersey.server.mvc.Template;
import org.glassfish.jersey.server.mvc.Viewable;

@Path("/msg")
public class MessageRestController {
 @GET
 @Produces(MediaType.TEXT_HTML)
 public Viewable get() {
 return new Viewable("/index.jsp", "MessageRestController");
 }
 @GET
 @Template(name="index1.jsp")
 public String get1() {
 return "MessageRestController";
 }
}
```

#### 4. Index.jsp:

```
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
```

#### 5. Index1.jsp:

```
<html>
<body>
<h2>Hello World!</h2>
raju
</body>
</html>
```

Output:  
Hello World!

---

The `@FormDataParam("file")` annotation is used to mention file parameter in the controller class. The `@Consumer(MediaType.MULTIPART_FORM_DATA)` is used to provide information of the file upload.

#### 1.pom.xml:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
```

```
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
container-servlet -->
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 -->
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<dependency>
 <groupId>org.glassfish.jersey.media</groupId>
 <artifactId>jersey-media-multipart</artifactId>
 <version>2.30</version>
</dependency>
</dependencies>
```

## 2.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;

import org.glassfish.jersey.media.multipart.MultiPartFeature;
import org.glassfish.jersey.server.ResourceConfig;
@ApplicationPath("/rest")//URL Pattern
public class AppConfig extends ResourceConfig { //FC:
ServletContainer
 //Constructor
public AppConfig() {
 packages("in.nit");//init-param
 register(MultiPartFeature.class);
}
}
```

## 3.controller:

```
package in.nit.controller;

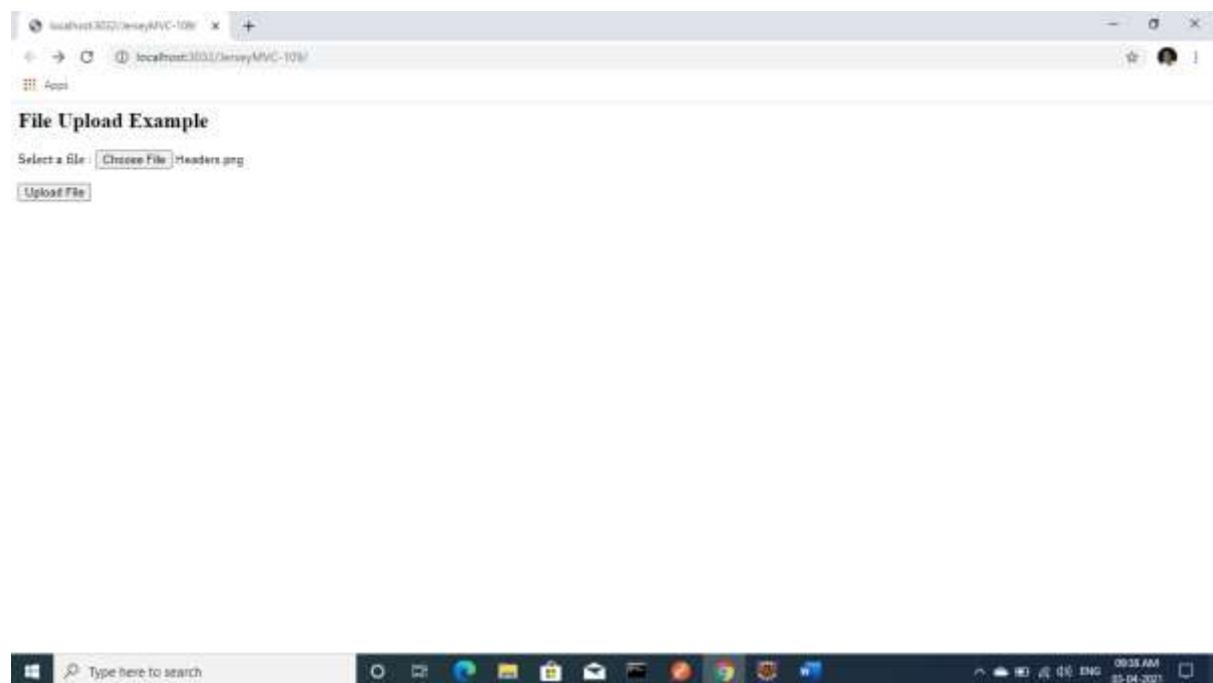
import javax.ws.rs.Path;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import org.glassfish.jersey.media.multipart.FormDataContentDisposition;
import org.glassfish.jersey.media.multipart.FormDataParam;
@Path("/files")
public class FileUploadController {
 @POST
 @Path("/upload")
 @Consumes(MediaType.MULTIPART_FORM_DATA)
 public Response uploadFile(
 @FormDataParam("file") InputStream uploadedInputStream,
 @FormDataParam("file") FormDataContentDisposition fileDetail)
 {
 String fileLocation = "e://" + fileDetail.getFileName();
 //saving file
 try {
 FileOutputStream out = new FileOutputStream(new
File(fileLocation));
 int read = 0;
 byte[] bytes = new byte[1024];
 out = new FileOutputStream(new File(fileLocation));
 while ((read = uploadedInputStream.read(bytes)) != -1) {
 out.write(bytes, 0, read);
 }
 out.flush();
 out.close();
 } catch (IOException e) {e.printStackTrace();}
 String output = "File successfully uploaded to : " + fileLocation;
 return Response.status(200).entity(output).build();
 }
}
```

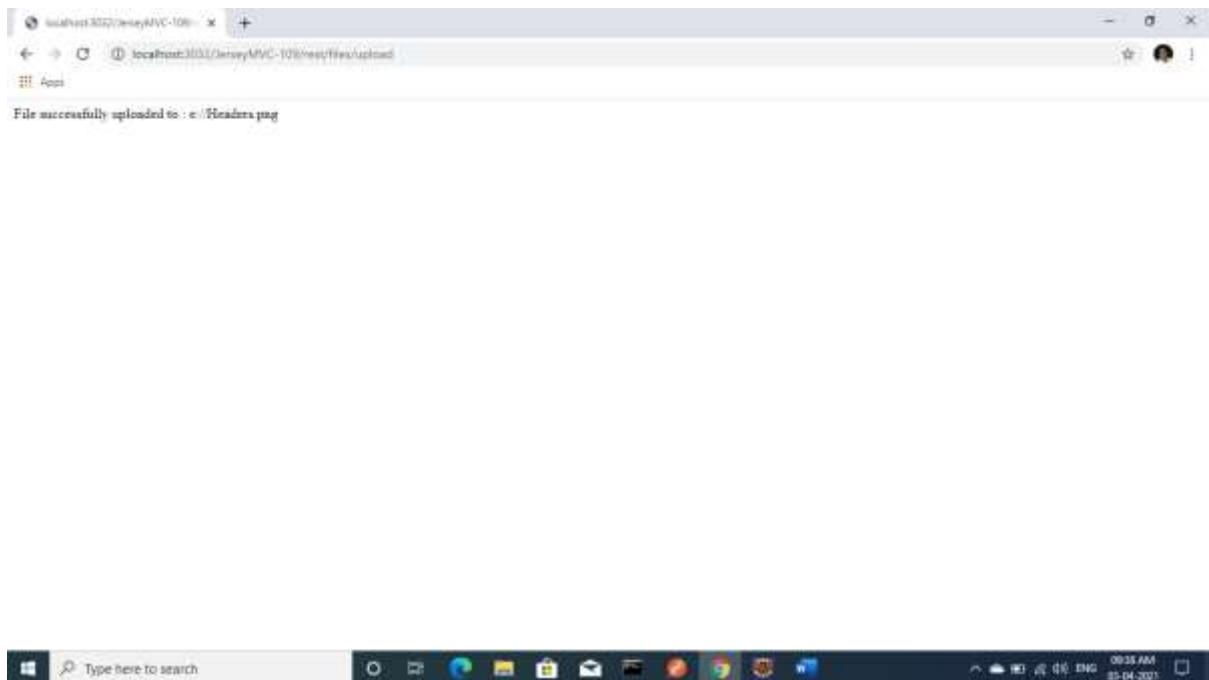
#### 4. Index.html:

```
<h2>File Upload Example</h2>
<form action="rest/files/upload" method="post" enctype="multipart/form-data">
<p>
Select a file : <input type="file" name="file" size="45" />
</p>
<input type="submit" value="Upload File" />
</form>
```

Output:



For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>



## Jersey Test Framework(Junit4.x):

1.pom.xml:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>

 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
 <scope>test</scope>
 </dependency>
 <!--
 https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-
 container-servlet -->
```

[https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet -->](https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet)

```
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
>
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<dependency>
 <groupId>org.glassfish.jersey.test-
framework.providers</groupId>
 <artifactId>jersey-test-framework-provider-
grizzly2</artifactId>
 <version>2.30</version>
</dependency>
</dependencies>
```

## 2.AppConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;
import org.glassfish.jersey.server.ResourceConfig;
@ApplicationPath("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC:
ServletContainer
 //Constructor
public AppConfig() {
 packages("in.nit");//init-param
}
}
```

## 3.Controller: Src/main/java

```
package in.nit.controller;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
```

```
@Path("/s")
public class Controller {
 @GET
 public static int findMax(int arr[]){
 int max=arr[0];//arr[0] instead of 0
 for(int i=1;i<arr.length;i++){
 if(max<arr[i])
 max=arr[i];
 }
 return max;
 }
}
```

#### 4.Junit Test class:

Src/main/test:

```
package in.nit.test;
import static org.junit.Assert.assertEquals;

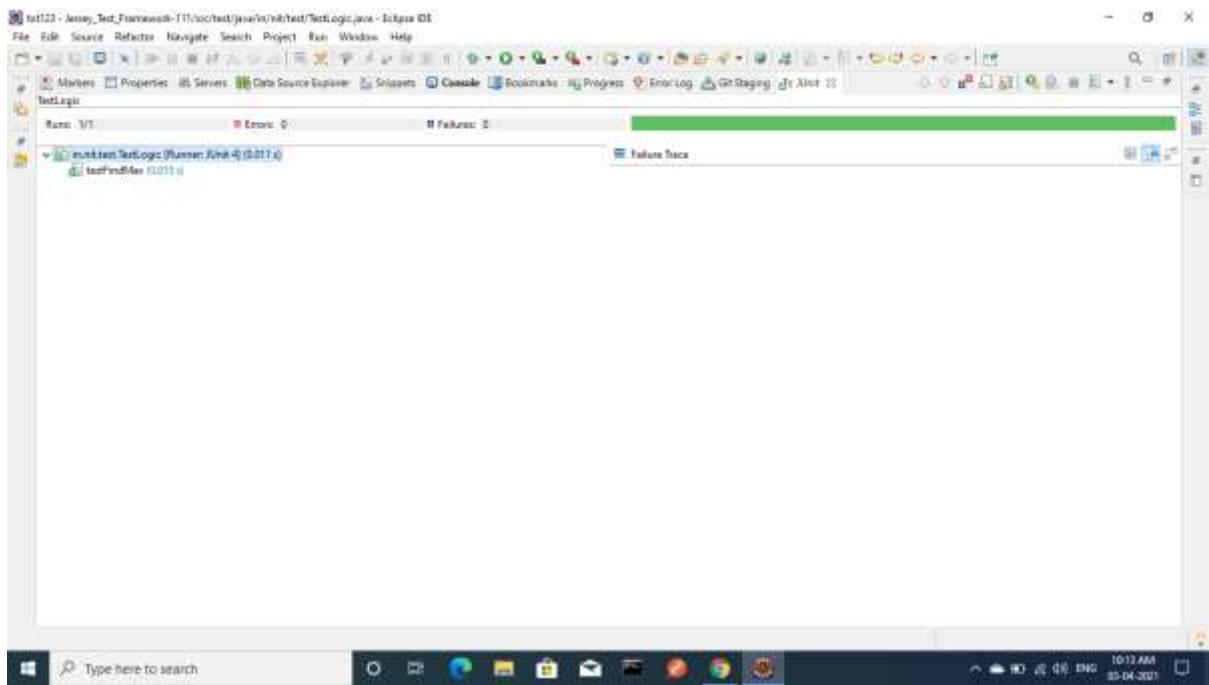
import org.junit.Test;

import in.nit.controller.Controller;

public class TestLogic {

 @Test
 public void testFindMax(){
 assertEquals(4,Controller.findMax(new int[]{1,3,4,2}));
 assertEquals(-1,Controller.findMax(new int[]{-12,-1,-3,-4,-2}));
 }
}

// right click on TestLogic class -> Run As -> 1Junit Test.
```



## Jersey logging Log4j:

1.pom.xml:

```
<properties>
 <failOnMissingWebXml>false</failOnMissingWebXml>
 <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
 <maven.compiler.source>15</maven.compiler.source>
 <maven.compiler.target>15</maven.compiler.target>
</properties>

<dependencies>
 <!-- https://mvnrepository.com/artifact/log4j/log4j -->
 <dependency>
 <groupId>log4j</groupId>
 <artifactId>log4j</artifactId>
 <version>1.2.17</version>
 </dependency>

 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.11</version>
```

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>

```
<scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.containers/jersey-container-servlet --
<dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 <version>2.30</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.inject/jersey-hk2 --
<dependency>
 <groupId>org.glassfish.jersey.inject</groupId>
 <artifactId>jersey-hk2</artifactId>
 <version>2.30</version>
</dependency>
<dependency>
 <groupId>org.glassfish.jersey.test-framework.providers</groupId>
 <artifactId>jersey-test-framework-provider-grizzly2</artifactId>
 <version>2.30</version>
 </dependency>
</dependencies>
```

## 2.appConfig:

```
package in.nit.config;

import javax.ws.rs.ApplicationPath;
import org.glassfish.jersey.server.ResourceConfig;
@Path("/rest")//URL Pattern
public class AppConfig extends ResourceConfig {//FC: ServletContainer
 //Constructor
 public AppConfig() {
 packages("in.nit");//init-param
 }
}
```

## 3.controller:

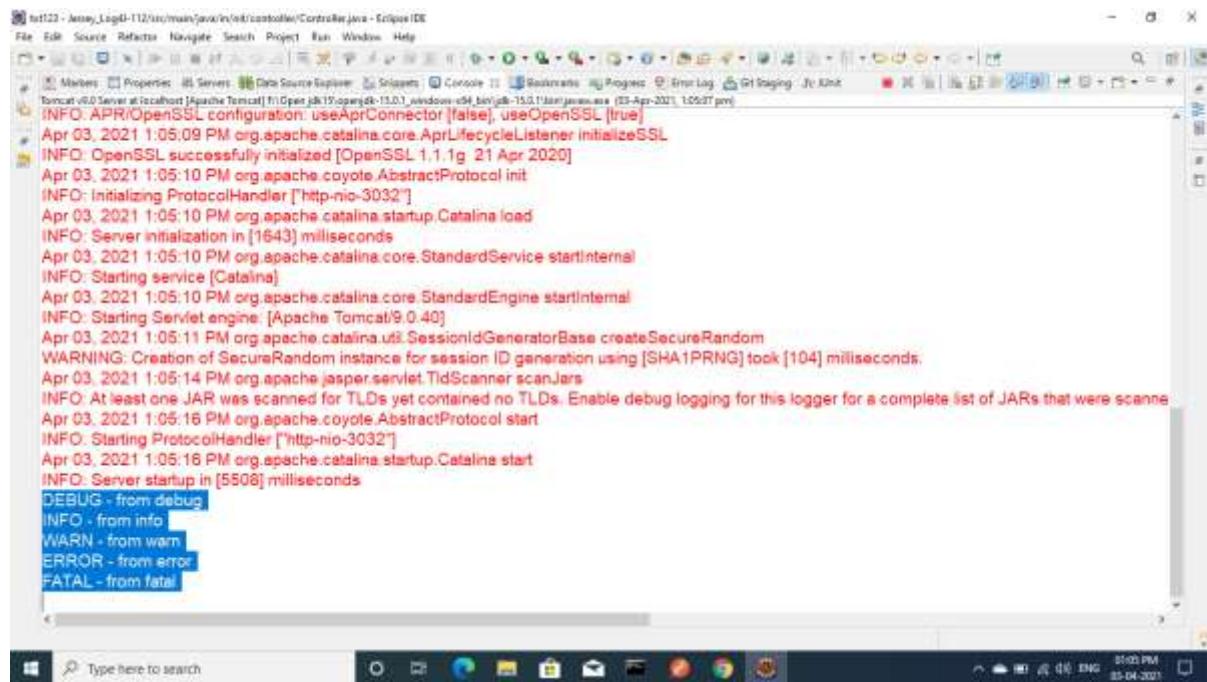
```
package in.nit.controller;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

import org.apache.log4j.Appender;
import org.apache.log4j.ConsoleAppender;
import org.apache.log4j.Layout;
import org.apache.log4j.Logger;
import org.apache.log4j.SimpleLayout;

@Path("/s")
public class Controller {
 private static Logger log = Logger.getLogger(Controller.class);
 @GET
 public static String message() {//must static method
 Layout layout=new SimpleLayout();
 Appender ap=new ConsoleAppender(layout);
 log.addAppender(ap);
 log.debug("from debug");
 log.info("from info");
 log.warn("from warn");
 log.error("from error");
 log.fatal("from fatal");
 return "welcome to ReST API";
 }
}
```

Output:

For Any Queries: [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
For Projects Github: <https://github.com/karrasankar158/Webservice>



The screenshot shows the Eclipse IDE interface with the title "tomcat - Jersey\_LagU-T12/src/main/java/in/kti/controller/Controller.java - Eclipse IDE". The central area displays the Tomcat server logs. The log output includes:

```
INFO: APR/OpenSSL configuration: useAprConnector [false], useOpenSSL [true]
Apr 03, 2021 1:05:09 PM org.apache.catalina.core.AprLifecycleListener initializeSSL
INFO: OpenSSL successfully initialized [OpenSSL 1.1.1g 21 Apr 2020]
Apr 03, 2021 1:05:10 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-nio-3032"]
Apr 03, 2021 1:05:10 PM org.apache.catalina.startup.Catalina load
INFO: Server initialization in [1643] milliseconds
Apr 03, 2021 1:05:10 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Catalina]
Apr 03, 2021 1:05:10 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/9.0.40]
Apr 03, 2021 1:05:11 PM org.apache.catalina.util.SessionIdGeneratorBase createSecureRandom
WARNING: Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [104] milliseconds.
Apr 03, 2021 1:05:14 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned.
Apr 03, 2021 1:05:16 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-3032"]
Apr 03, 2021 1:05:16 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in [5508] milliseconds
DEBUG - from debug
INFO - from info
WARN - from warn
ERROR - from error
FATAL - from fatal
```

The status bar at the bottom right shows the date and time: 31/03 PM 25-04-2021.

\*\*\*\*\*All The Best\*\*\*\*\*

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>

**For Any Queries:** [karrasankar@gmail.com](mailto:karrasankar@gmail.com)  
**For Projects Github:** <https://github.com/karrasankar158/Webservice>