CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Web Services
420-941-VA

Teacher: Samir Chebbine **Lab 6** **Nov 13, 2024**

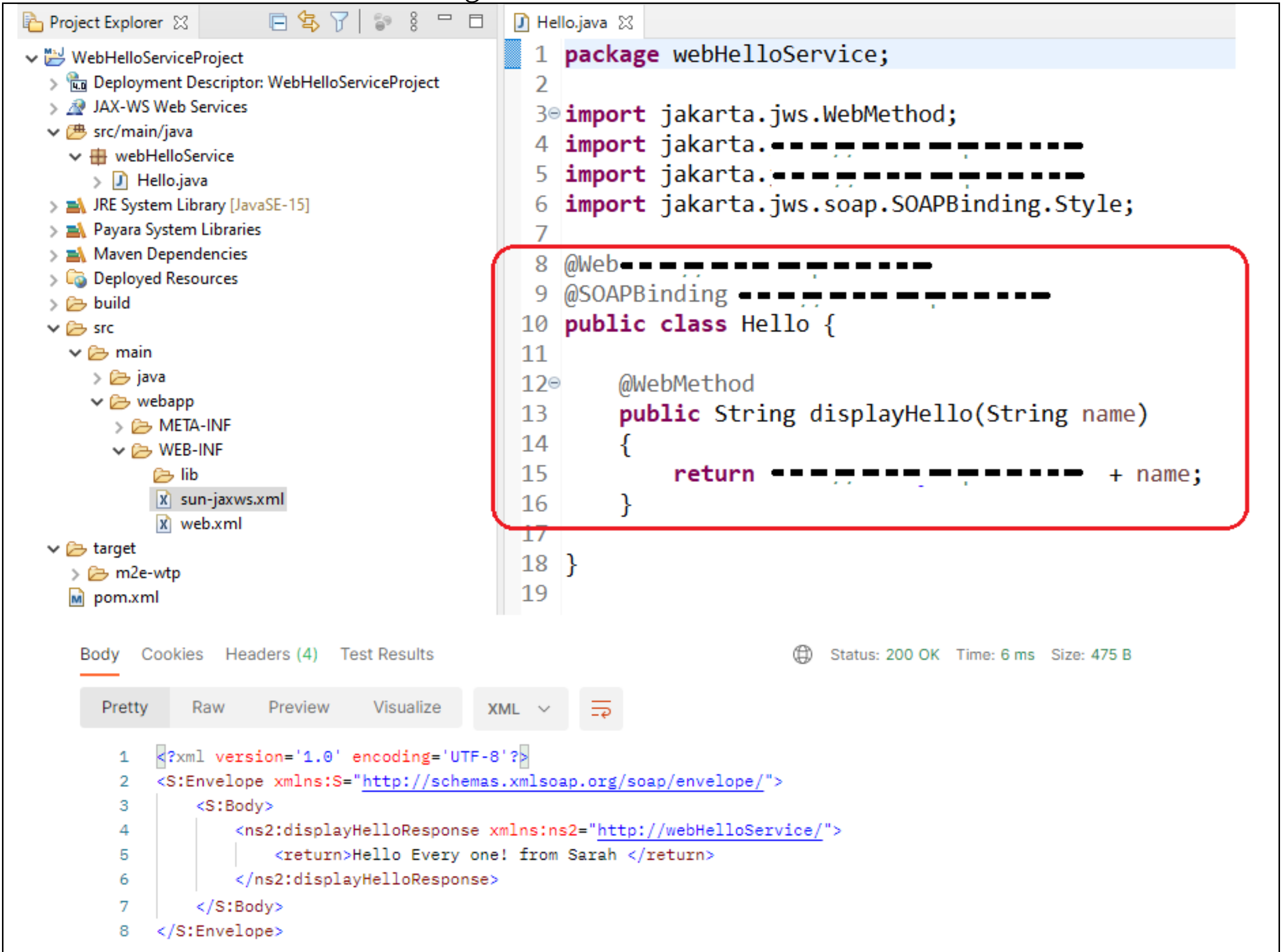
Lab 6: Glassfish SOAP Web Services & Spring Boot REST Resources

Complete all these following programs in class**.** All *missing coding statements* are presented during class time and in Presentation 6.

Create and Submit a Word file ***Lab6WebServicesYourName.doc*** which contains Answers of theory questions if any and output screenshots for every Java EE Project. Submit the Java projects too and submit the whole Lab 6 as compressed zip file

1. SOAP-Based Application

a) Create a new Dynamic Web project called **WebHelloServiceProject** and convert it into Maven Project.
b) Deploy **WebHelloServiceProject** within GalssFish Server.
c) Using Postman, display screenshots testing each SOAP request included in my YouTube Video Lab 6 as shown in Figure 1.



```
WebHelloServiceProject
  Deployment Descriptor: WebHelloServiceProject
  JAX-WS Web Services
  src/main/java
    webHelloService
      Hello.java
  JRE System Library [JavaSE-15]
  Payara System Libraries
  Maven Dependencies
  Deployed Resources
  build
  src
    main
      java
      webapp
        META-INF
        WEB-INF
          lib
          sun-jaxws.xml
          web.xml
  target
    m2e-wtp
  pom.xml
```

```java
package webHelloService;

import jakarta.jws.WebMethod;
import jakarta.----------------
import jakarta.----------------
import jakarta.jws.soap.SOAPBinding.Style;

@Web--------------------
@SOAPBinding ---------------------
public class Hello {

    @WebMethod
    public String displayHello(String name)
    {
        return ----------------------  + name;
    }

}
```

```xml
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:displayHelloResponse xmlns:ns2="http://webHelloService/">
            <return>Hello Every one! from Sarah </return>
        </ns2:displayHelloResponse>
    </S:Body>
</S:Envelope>
```

Figure 1

2. Maven Dynamic Web Project: WebBillingServiceProject

a) Create a new Dynamic Web project called **WebBillingServiceProject and convert it into Maven Project**.

b) Add **Maven Project dependencies** in **pom.xml**. Create new package called **webBillingService**.

c) Specify Web Service Endpoint (ws/WebBilling) in sun-jaxws.xml and deploy **WebBillingServiceProject** within GalssFish Server.

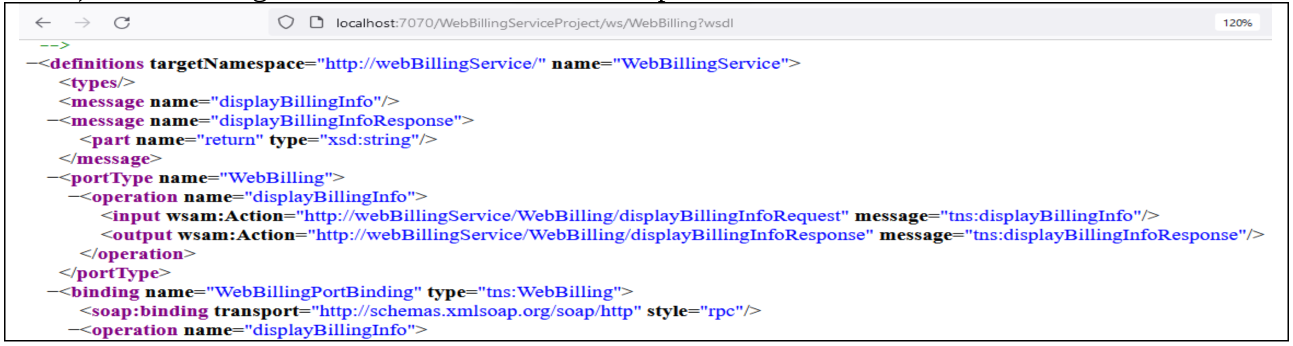
d) You need to develop a **Java class** called `Billing`, which takes `client_ID`, `client_LName`, `and client_FName`, `product_Name`, `prd_Price=0`, `prd_Qty` as **private** `non static` members. The variables called `Fed_Tax`, `Prv_Tax` as **public** and `static` data members. The `Billing` class contains the following method members:
(Notice that it is same class as in Lab 3)

e) Create a new SOAP Service class **WebBilling.java**

1. Add a method **displayBillingInfo()** to instantiate a Java data structure **Array List** `of object of Billing class type` to be referenced by (`BillingList`). Add every component of Array List Billing object using the implemented setter methods (`setClient_ID()`, `setClient_LName()`,`setClient_FName ()`,`setproduct_Name()`, `setPrd_Price()`, `setPrd_Qty()`) to values read from input file `Billing.in` of Lab 3.

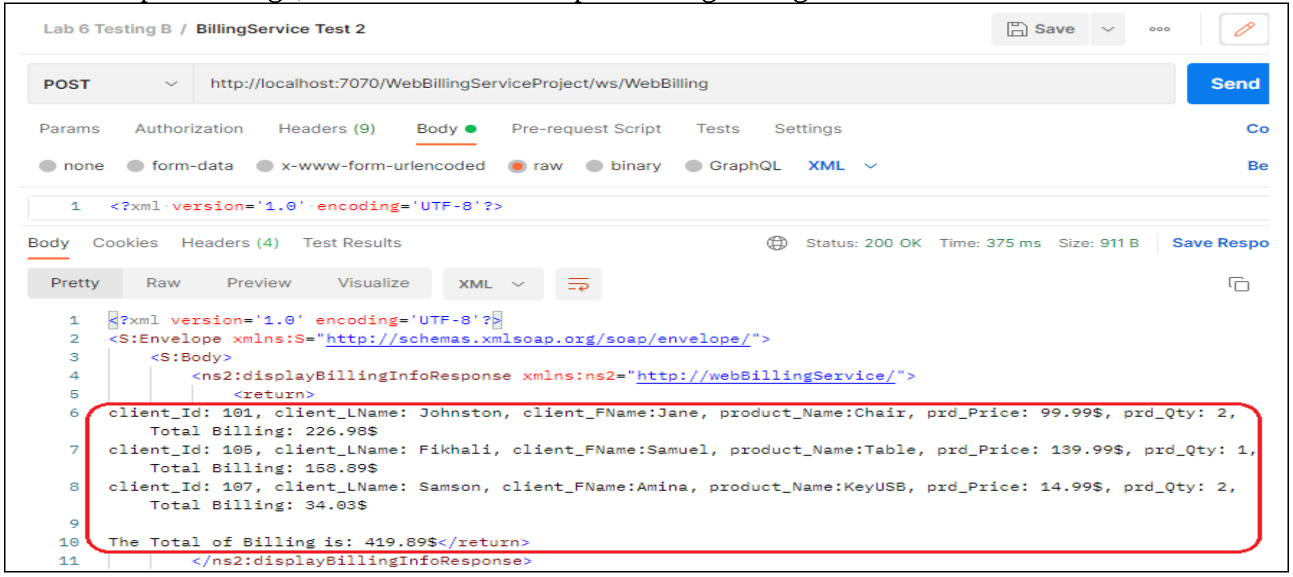
2. Skip through Array `List of object` (`BillingList`) and return the content of populated Array List.

f) Check the generated WSDL service description as shown hereafter.



```
localhost:7070/WebBillingServiceProject/ws/WebBilling?wsdl
-->
-<definitions targetNamespace="http://webBillingService/" name="WebBillingService">
   <types/>
   <message name="displayBillingInfo"/>
  -<message name="displayBillingInfoResponse">
     <part name="return" type="xsd:string"/>
   </message>
  -<portType name="WebBilling">
    -<operation name="displayBillingInfo">
       <input wsam:Action="http://webBillingService/WebBilling/displayBillingInfoRequest" message="tns:displayBillingInfo"/>
       <output wsam:Action="http://webBillingService/WebBilling/displayBillingInfoResponse" message="tns:displayBillingInfoResponse"/>
     </operation>
   </portType>
  -<binding name="WebBillingPortBinding" type="tns:WebBilling">
     <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    -<operation name="displayBillingInfo">
```

g) Using appropriate SOAP XML body message related to **displayBillingInfo()** in SOAP input message, check the SOAP output message using Postman as shown hereafter.



POST http://localhost:7070/WebBillingServiceProject/ws/WebBilling

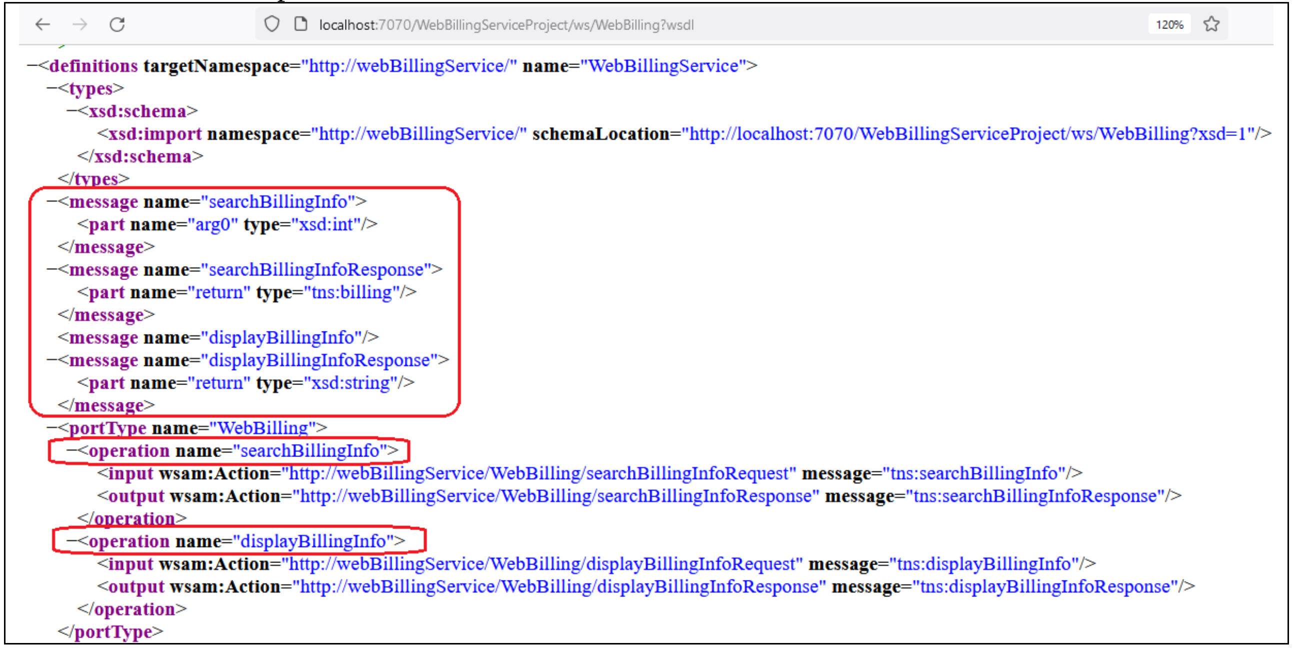
```
<?xml version='1.0' encoding='UTF-8'?>
```

Status: 200 OK Time: 375 ms Size: 911 B

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:displayBillingInfoResponse xmlns:ns2="http://webBillingService/">
            <return>
client_Id: 101, client_LName: Johnston, client_FName:Jane, product_Name:Chair, prd_Price: 99.99$, prd_Qty: 2,
    Total Billing: 226.98$
client_Id: 105, client_LName: Fikhali, client_FName:Samuel, product_Name:Table, prd_Price: 139.99$, prd_Qty: 1,
    Total Billing: 158.89$
client_Id: 107, client_LName: Samson, client_FName:Amina, product_Name:KeyUSB, prd_Price: 14.99$, prd_Qty: 2,
    Total Billing: 34.03$

The Total of Billing is: 419.89$</return>
        </ns2:displayBillingInfoResponse>
```

h) Add a method **searchBillingInfo(int client_id)** in SOAP Service class **WebBilling.java** that returns the result of Array List search.

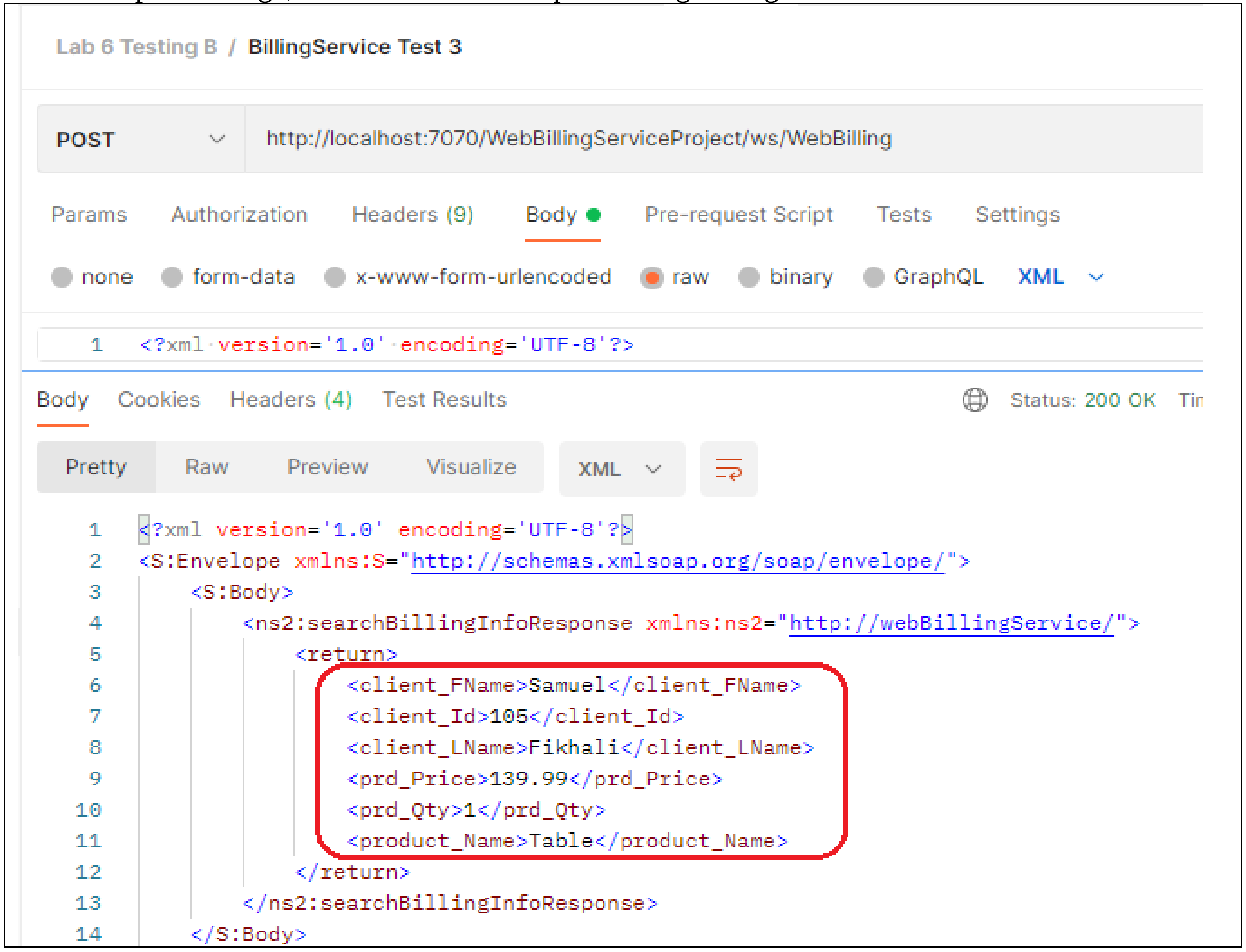
i) Check the new generated WSDL service description which includes the description of two SOAP operations as shown hereafter.



localhost:7070/WebBillingServiceProject/ws/WebBilling?wsdl

```
-<definitions targetNamespace="http://webBillingService/" name="WebBillingService">
  -<types>
    -<xsd:schema>
       <xsd:import namespace="http://webBillingService/" schemaLocation="http://localhost:7070/WebBillingServiceProject/ws/WebBilling?xsd=1"/>
     </xsd:schema>
   </types>
  -<message name="searchBillingInfo">
     <part name="arg0" type="xsd:int"/>
   </message>
  -<message name="searchBillingInfoResponse">
     <part name="return" type="tns:billing"/>
   </message>
   <message name="displayBillingInfo"/>
  -<message name="displayBillingInfoResponse">
     <part name="return" type="xsd:string"/>
   </message>
  -<portType name="WebBilling">
    -<operation name="searchBillingInfo">
       <input wsam:Action="http://webBillingService/WebBilling/searchBillingInfoRequest" message="tns:searchBillingInfo"/>
       <output wsam:Action="http://webBillingService/WebBilling/searchBillingInfoResponse" message="tns:searchBillingInfoResponse"/>
     </operation>
    -<operation name="displayBillingInfo">
       <input wsam:Action="http://webBillingService/WebBilling/displayBillingInfoRequest" message="tns:displayBillingInfo"/>
       <output wsam:Action="http://webBillingService/WebBilling/displayBillingInfoResponse" message="tns:displayBillingInfoResponse"/>
     </operation>
   </portType>
```

j) Using appropriate SOAP XML body message related to **searchBillingInfo()** in SOAP input message, check the SOAP output message using Postman as shown hereafter.



Lab 6 Testing B / BillingService Test 3

POST http://localhost:7070/WebBillingServiceProject/ws/WebBilling

Body: raw, XML

```
<?xml version='1.0' encoding='UTF-8'?>
```

Status: 200 OK

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:searchBillingInfoResponse xmlns:ns2="http://webBillingService/">
            <return>
                <client_FName>Samuel</client_FName>
                <client_Id>105</client_Id>
                <client_LName>Fikhali</client_LName>
                <prd_Price>139.99</prd_Price>
                <prd_Qty>1</prd_Qty>
                <product_Name>Table</product_Name>
            </return>
        </ns2:searchBillingInfoResponse>
    </S:Body>
```

3. Maven Dynamic Web Project: WebCourseServiceProject

a) Create a new Dynamic Web project called **WebCourseServiceProject and convert it into Maven Project**.

b) Add **Maven Project dependencies** in **pom.xml**. Create new package called **webCourseService**.

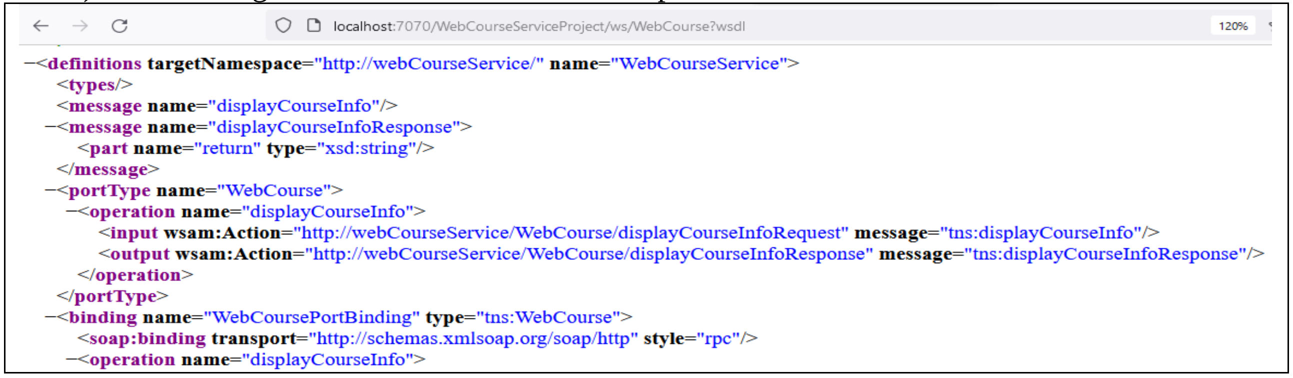
c) Specify Web Service Endpoint (ws/WebCourse) in sun-jaxws.xml and deploy **WebCourseServiceProject** within GalssFish Server.

d) You need to develop a **Java class** called **`Course`** that *represents* a template of the fields used in defining the columns of a given table *Course* which takes `course_no`, `course_name`, `max_enrl` as **private** `non static` data members. `credits` as **public** and `static` data member. The `Course` class contains the following method members:
(Notice that it is same class as in Lab 1)

e) Create a new SOAP Service class **WebCourse.java**

1. Add a method **displayCourseInfo()** to instantiate a Java data structure **Array List** `of object of Course class type` to be referenced by (`CourseList`). Add every component of Array List course object to values read from input file `Course.in` of Lab 3.
2. Skip through `Array List of object` (`CourseList`) and return the content of populated Array List.

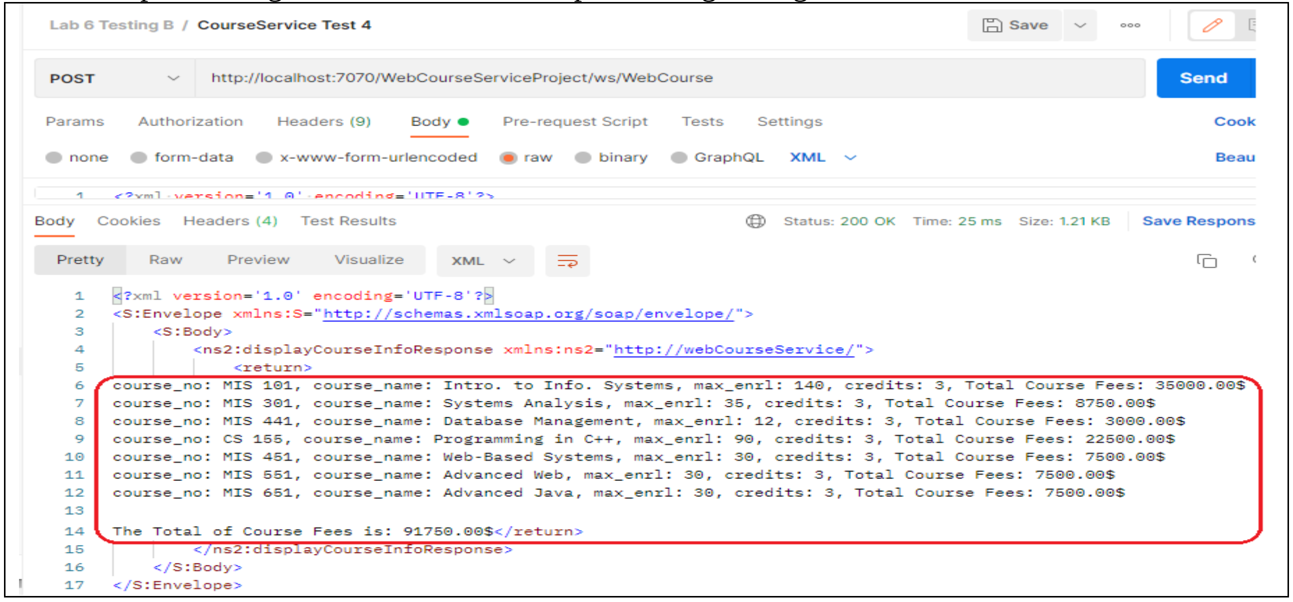
k) Check the generated WSDL service description as shown hereafter.



localhost:7070/WebCourseServiceProject/ws/WebCourse?wsdl

```
−<definitions targetNamespace="http://webCourseService/" name="WebCourseService">
   <types/>
   <message name="displayCourseInfo"/>
  −<message name="displayCourseInfoResponse">
     <part name="return" type="xsd:string"/>
   </message>
  −<portType name="WebCourse">
    −<operation name="displayCourseInfo">
       <input wsam:Action="http://webCourseService/WebCourse/displayCourseInfoRequest" message="tns:displayCourseInfo"/>
       <output wsam:Action="http://webCourseService/WebCourse/displayCourseInfoResponse" message="tns:displayCourseInfoResponse"/>
     </operation>
   </portType>
  −<binding name="WebCoursePortBinding" type="tns:WebCourse">
     <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    −<operation name="displayCourseInfo">
```

l) Using appropriate SOAP XML body message related to **displayCourseInfo()** in SOAP input message, check the SOAP output message using Postman as shown hereafter.



Lab 6 Testing B / CourseService Test 4

POST http://localhost:7070/WebCourseServiceProject/ws/WebCourse

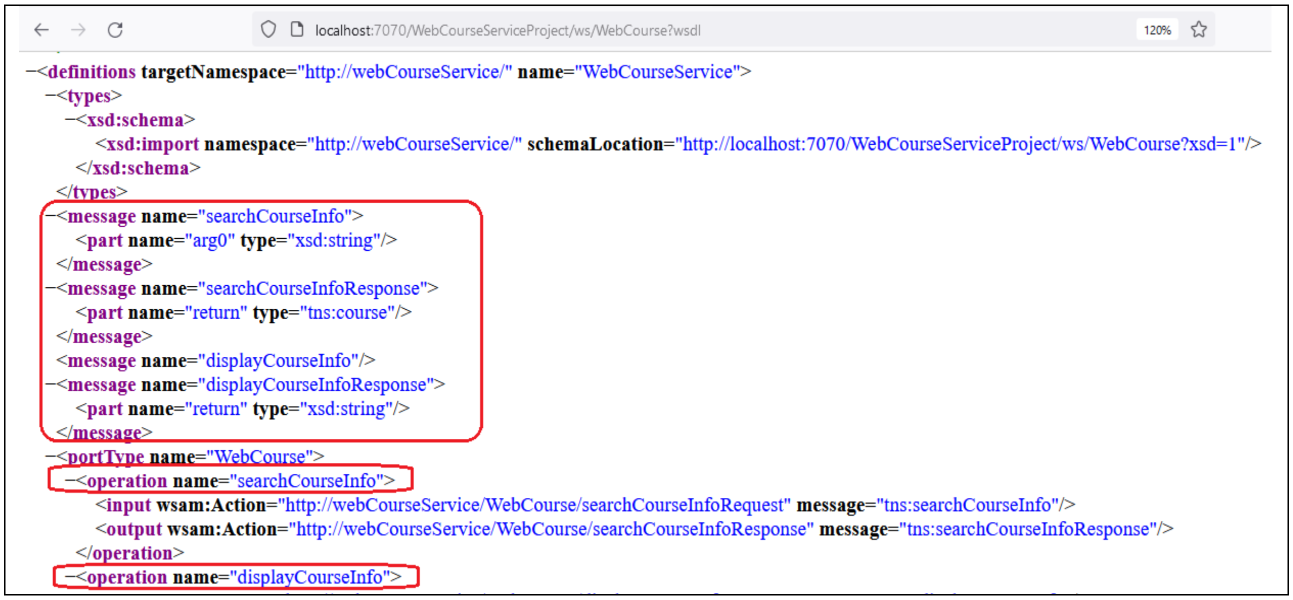
Body: raw XML

Status: 200 OK Time: 25 ms Size: 1.21 KB

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:displayCourseInfoResponse xmlns:ns2="http://webCourseService/">
            <return>
course_no: MIS 101, course_name: Intro. to Info. Systems, max_enrl: 140, credits: 3, Total Course Fees: 35000.00$
course_no: MIS 301, course_name: Systems Analysis, max_enrl: 35, credits: 3, Total Course Fees: 8750.00$
course_no: MIS 441, course_name: Database Management, max_enrl: 12, credits: 3, Total Course Fees: 3000.00$
course_no: CS 155, course_name: Programming in C++, max_enrl: 90, credits: 3, Total Course Fees: 22500.00$
course_no: MIS 451, course_name: Web-Based Systems, max_enrl: 30, credits: 3, Total Course Fees: 7500.00$
course_no: MIS 551, course_name: Advanced Web, max_enrl: 30, credits: 3, Total Course Fees: 7500.00$
course_no: MIS 651, course_name: Advanced Java, max_enrl: 30, credits: 3, Total Course Fees: 7500.00$

The Total of Course Fees is: 91750.00$</return>
        </ns2:displayCourseInfoResponse>
    </S:Body>
</S:Envelope>
```

m) Add a method **searchCourseInfo(String course_no)** that returns the result of Array List search.

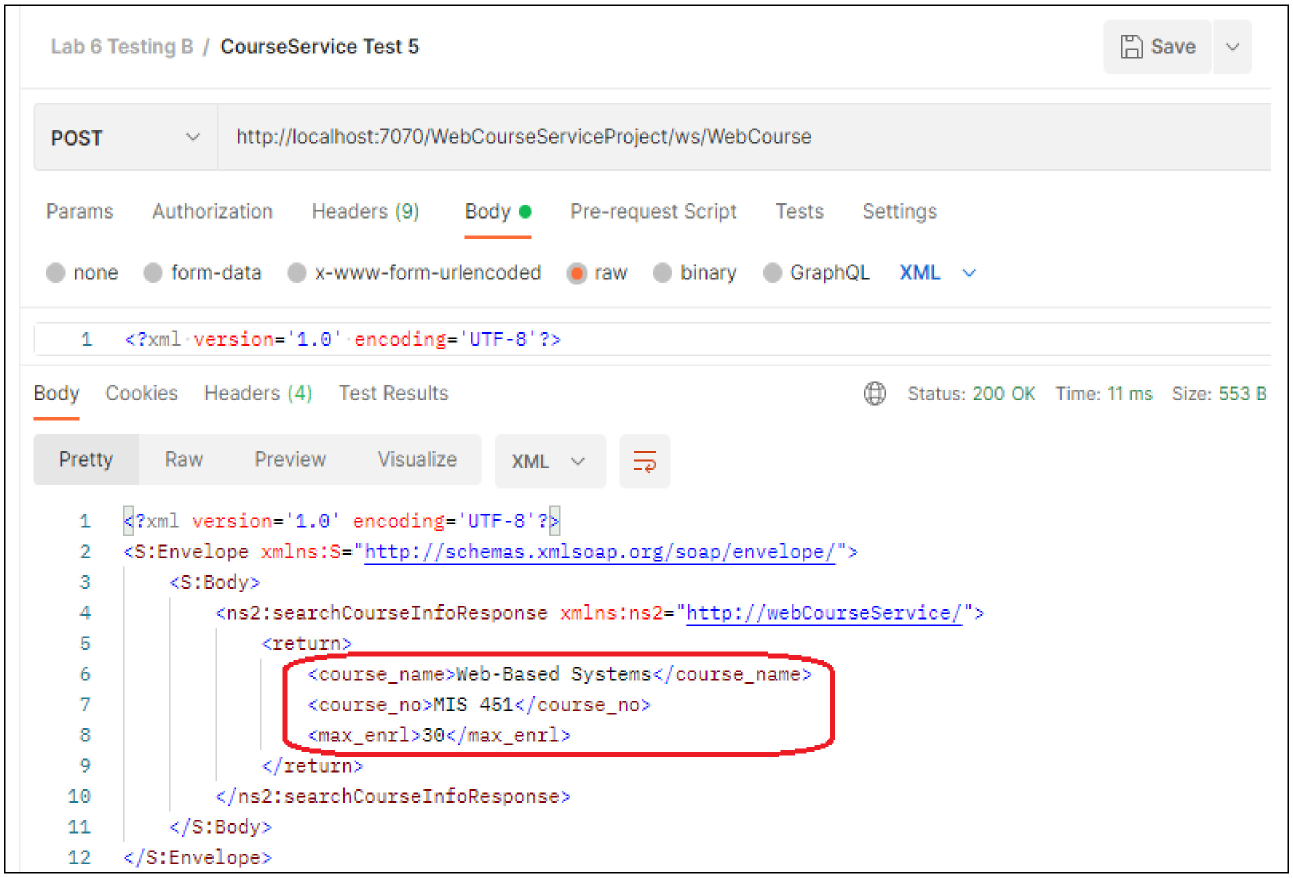
n) Check the new generated WSDL service description which includes the description of two SOAP operations as shown hereafter.



localhost:7070/WebCourseServiceProject/ws/WebCourse?wsdl

```
-<definitions targetNamespace="http://webCourseService/" name="WebCourseService">
  -<types>
    -<xsd:schema>
        <xsd:import namespace="http://webCourseService/" schemaLocation="http://localhost:7070/WebCourseServiceProject/ws/WebCourse?xsd=1"/>
      </xsd:schema>
    </types>
  -<message name="searchCourseInfo">
      <part name="arg0" type="xsd:string"/>
    </message>
  -<message name="searchCourseInfoResponse">
      <part name="return" type="tns:course"/>
    </message>
    <message name="displayCourseInfo"/>
  -<message name="displayCourseInfoResponse">
      <part name="return" type="xsd:string"/>
    </message>
  -<portType name="WebCourse">
    -<operation name="searchCourseInfo">
        <input wsam:Action="http://webCourseService/WebCourse/searchCourseInfoRequest" message="tns:searchCourseInfo"/>
        <output wsam:Action="http://webCourseService/WebCourse/searchCourseInfoResponse" message="tns:searchCourseInfoResponse"/>
      </operation>
    -<operation name="displayCourseInfo">
```

o) Using appropriate SOAP XML body message related to **searchCourseInfo()** in SOAP input message, check the SOAP output message using Postman as shown hereafter.



Lab 6 Testing B / CourseService Test 5

POST http://localhost:7070/WebCourseServiceProject/ws/WebCourse

Body: raw, XML

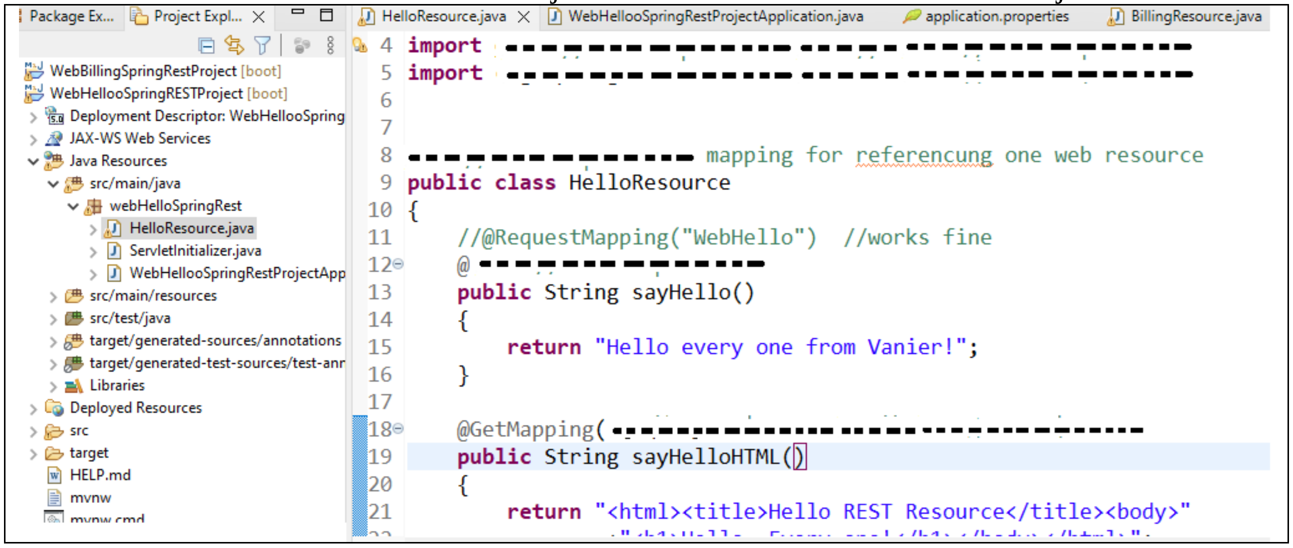
```
<?xml version='1.0' encoding='UTF-8'?>
```

Status: 200 OK Time: 11 ms Size: 553 B

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:searchCourseInfoResponse xmlns:ns2="http://webCourseService/">
            <return>
                <course_name>Web-Based Systems</course_name>
                <course_no>MIS 451</course_no>
                <max_enrl>30</max_enrl>
            </return>
        </ns2:searchCourseInfoResponse>
    </S:Body>
</S:Envelope>
```

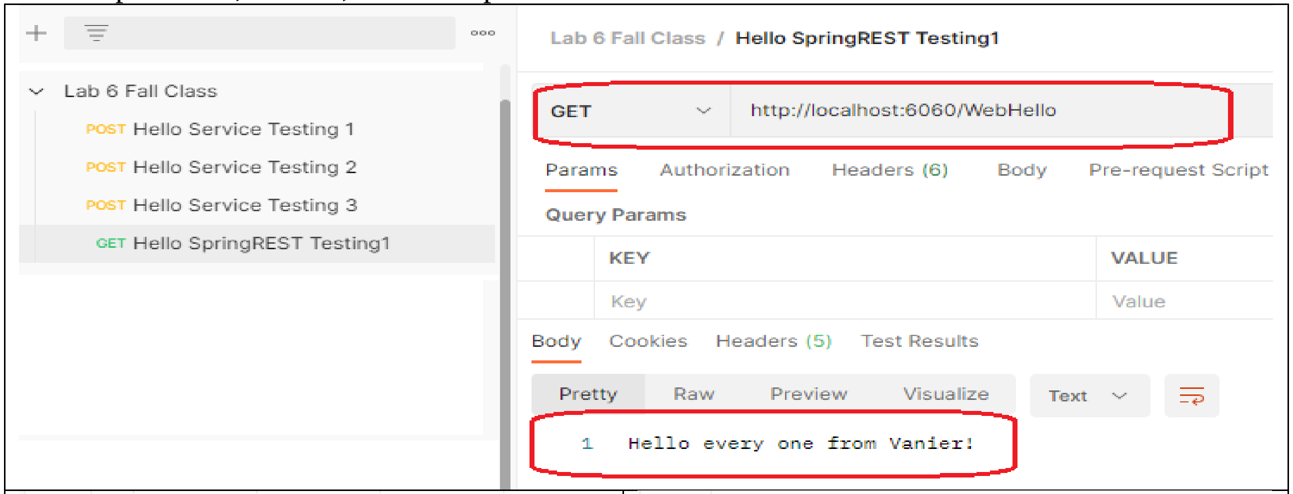
4. Creating REST Projects Using Spring Boot Framework

a) Complete all these following programs in class. All *missing coding statements* were presented during class time. Your WebHelloSpringRESTProject should reflect all options used in Lab 3 WebHelloRESTProject and Lab5 WebHelloFormRESTProject.



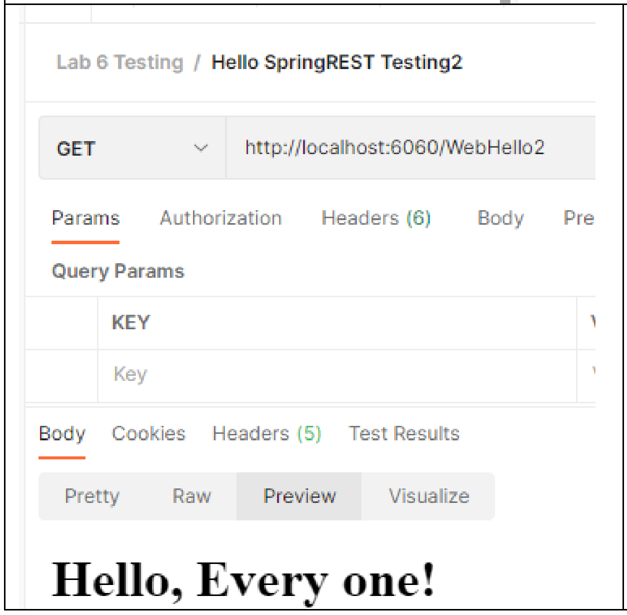
```
import ---------------------------------------
import ---------------------------------------


---------------- mapping for referencung one web resource
public class HelloResource
{
    //@RequestMapping("WebHello")  //works fine
    @ ---------------
    public String sayHello()
    {
        return "Hello every one from Vanier!";
    }

    @GetMapping( --------------------------------
    public String sayHelloHTML()
    {
        return "<html><title>Hello REST Resource</title><body>"
```

b) Using Postman, display screenshots testing to each media type included in Lab6 related to plain text, HTML, JSON output as shown hereafter.



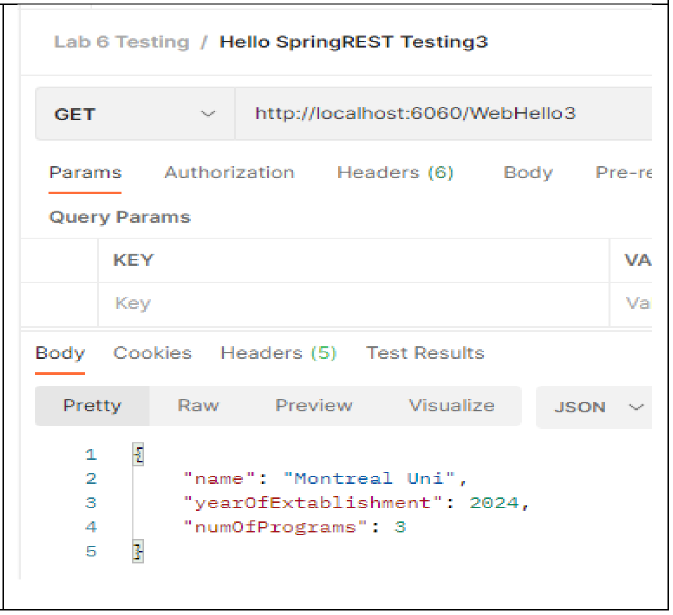
GET http://localhost:6060/WebHello

```
Hello every one from Vanier!
```



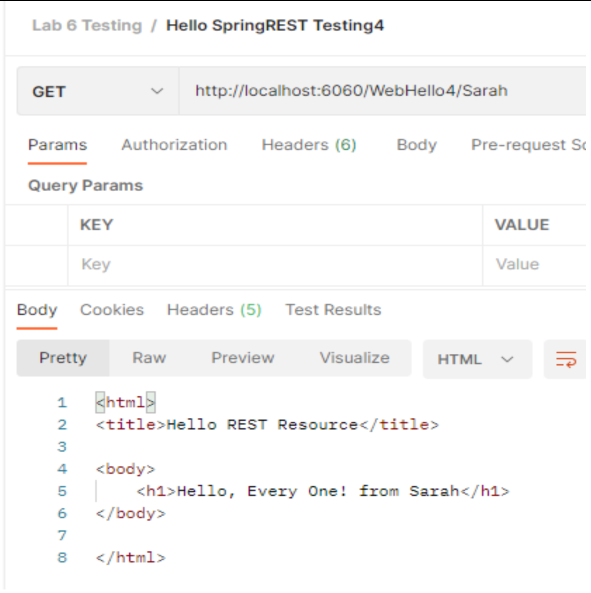
GET http://localhost:6060/WebHello2

Hello, Every one!



GET http://localhost:6060/WebHello3

```
{
    "name": "Montreal Uni",
    "yearOfExtablishment": 2024,
    "numOfPrograms": 3
}
```

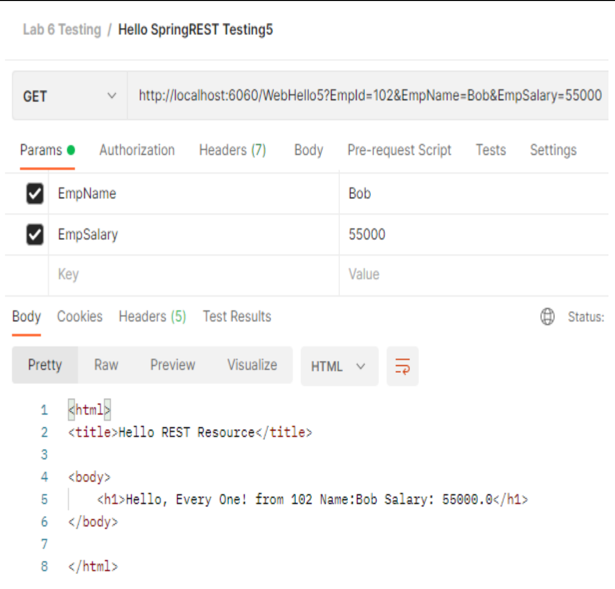

Lab 6 Testing / Hello SpringREST Testing4

GET http://localhost:6060/WebHello4/Sarah

KEY	VALUE
Key	Value

```
<html>
<title>Hello REST Resource</title>

<body>
    <h1>Hello, Every One! from Sarah</h1>
</body>

</html>
```

Path Parameter

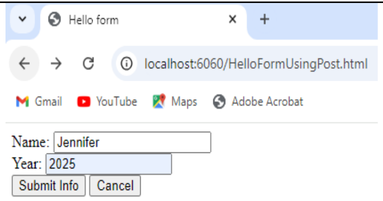

Lab 6 Testing / Hello SpringREST Testing5

GET http://localhost:6060/WebHello5?EmpId=102&EmpName=Bob&EmpSalary=55000

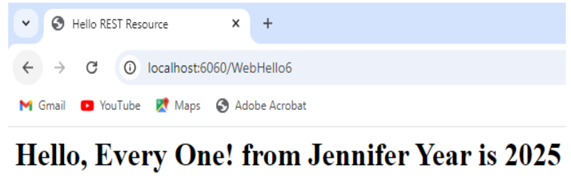
✓	EmpName	Bob
✓	EmpSalary	55000
	Key	Value

```
<html>
<title>Hello REST Resource</title>

<body>
    <h1>Hello, Every One! from 102 Name:Bob Salary: 55000.0</h1>
</body>

</html>
```

Query Parameters

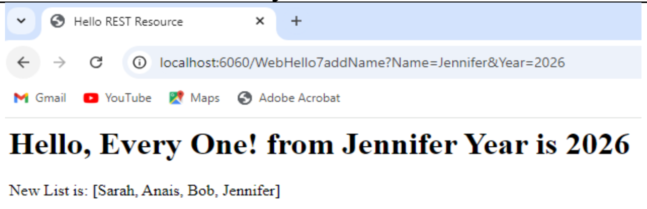


Name: Jennifer
Year: 2025



Hello, Every One! from Jennifer Year is 2025

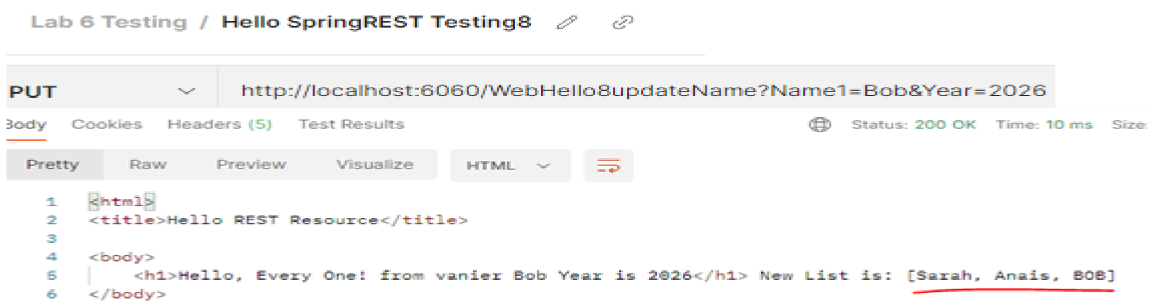
Form processing using POST method in HTTP request



Hello, Every One! from Jennifer Year is 2026

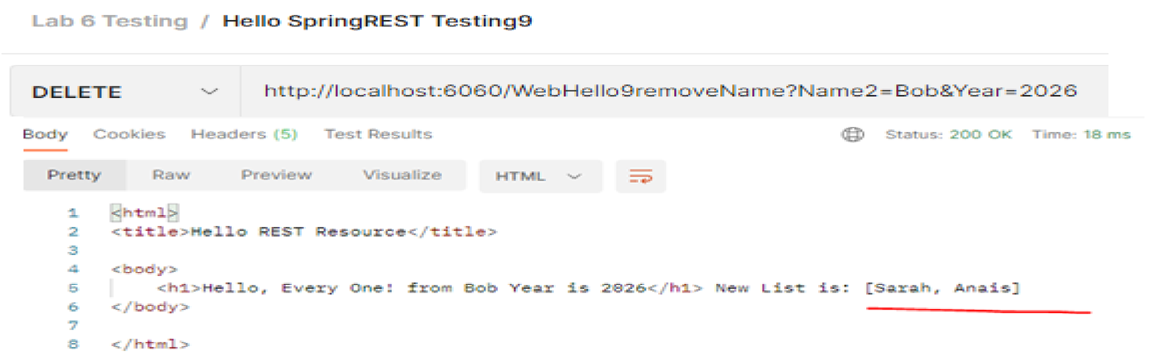
New List is: [Sarah, Anais, Bob, Jennifer]

Form processing using GET method in HTTP request to add new Name into ArrayList


Lab 6 Testing / Hello SpringREST Testing8

PUT http://localhost:6060/WebHello8updateName?Name1=Bob&Year=2026

```
<html>
<title>Hello REST Resource</title>

<body>
    <h1>Hello, Every One! from vanier Bob Year is 2026</h1> New List is: [Sarah, Anais, BOB]
</body>
```


Lab 6 Testing / Hello SpringREST Testing9

DELETE http://localhost:6060/WebHello9removeName?Name2=Bob&Year=2026

```
<html>
<title>Hello REST Resource</title>

<body>
    <h1>Hello, Every One! from Bob Year is 2026</h1> New List is: [Sarah, Anais]
</body>

</html>
```

Form processing using PUT & DELETE methods in HTTP request