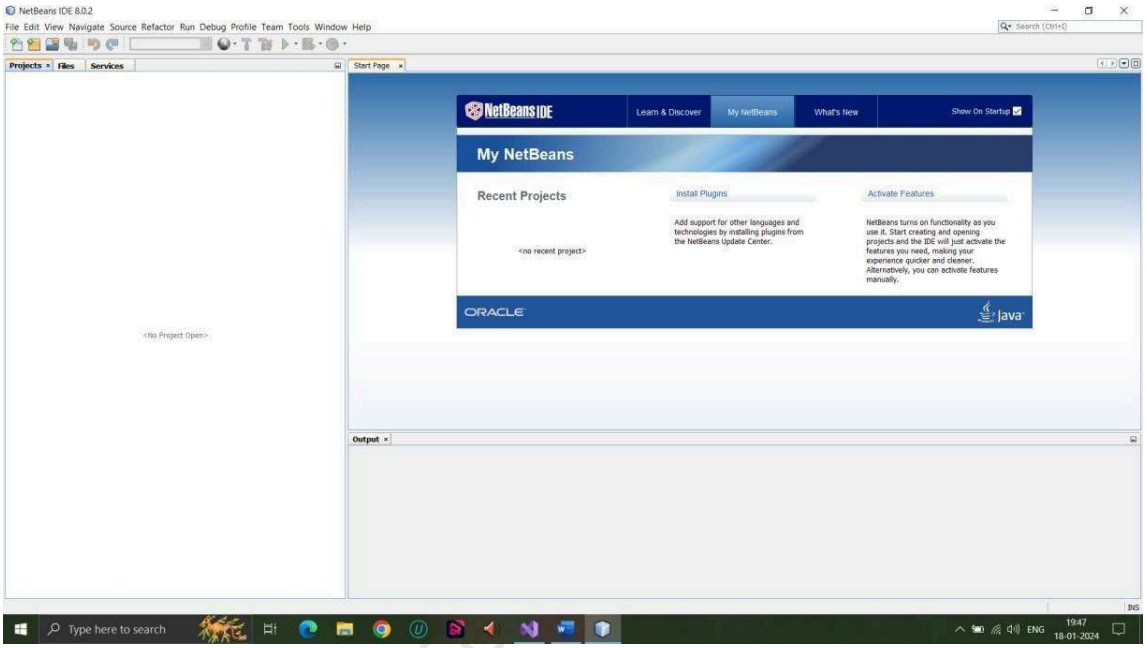
Cloud Computing and Web Services Practical

Practical - 1

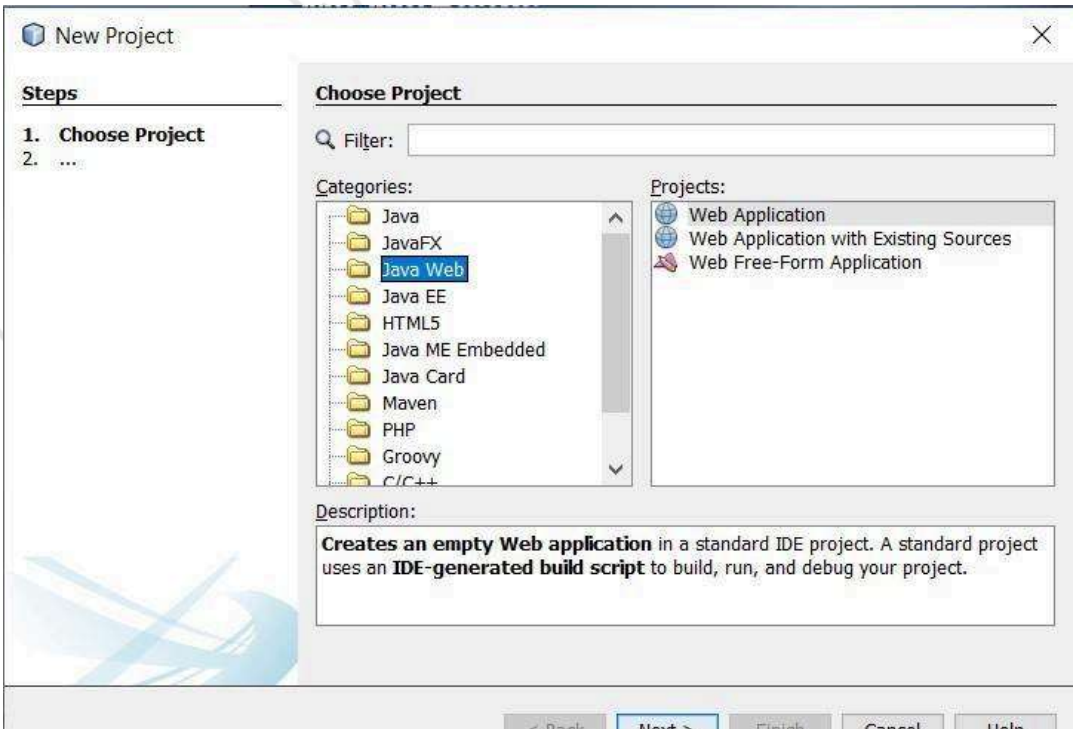
Aim: Define a simple services like Converting Rs into Dollar and Call it from different platform like JAVA

Steps:

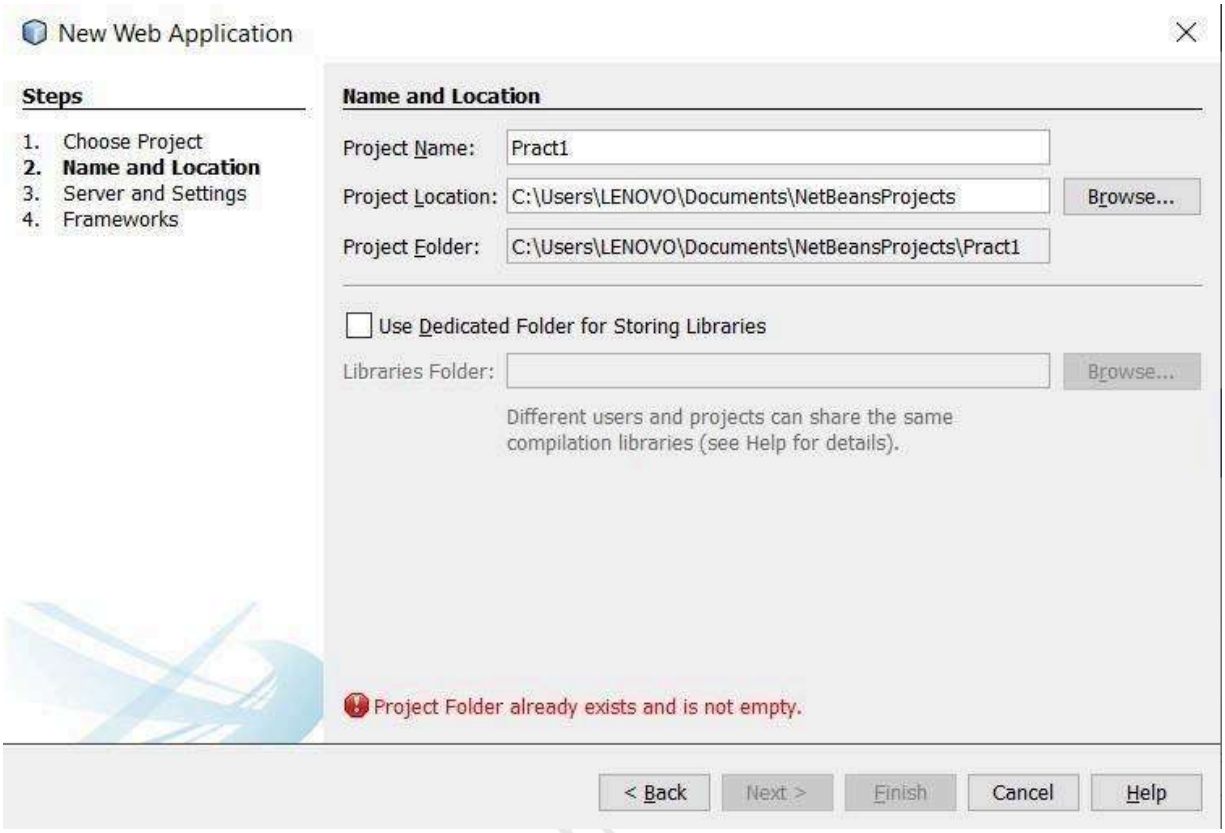
1] Open the NetBeans, and you will get the following screen. Close the start page.



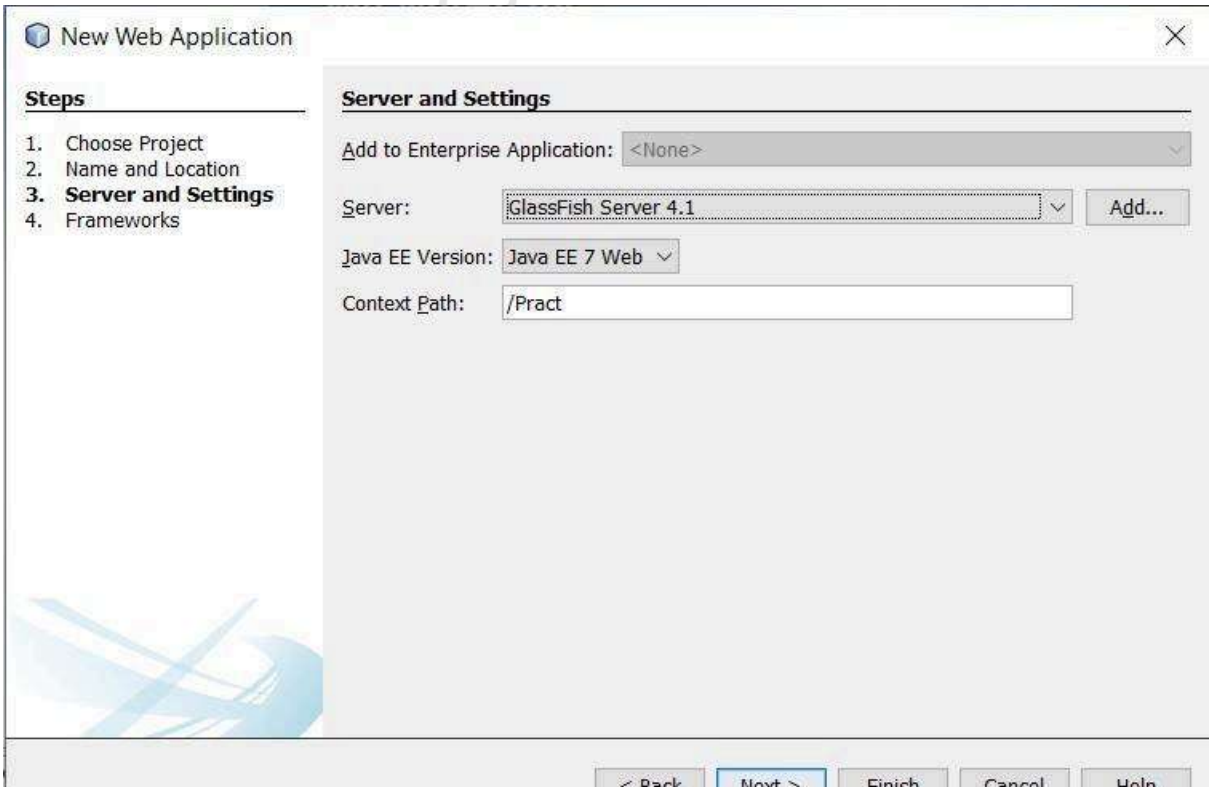
2] Now click on the file tab and click on new project you will get the following screen :



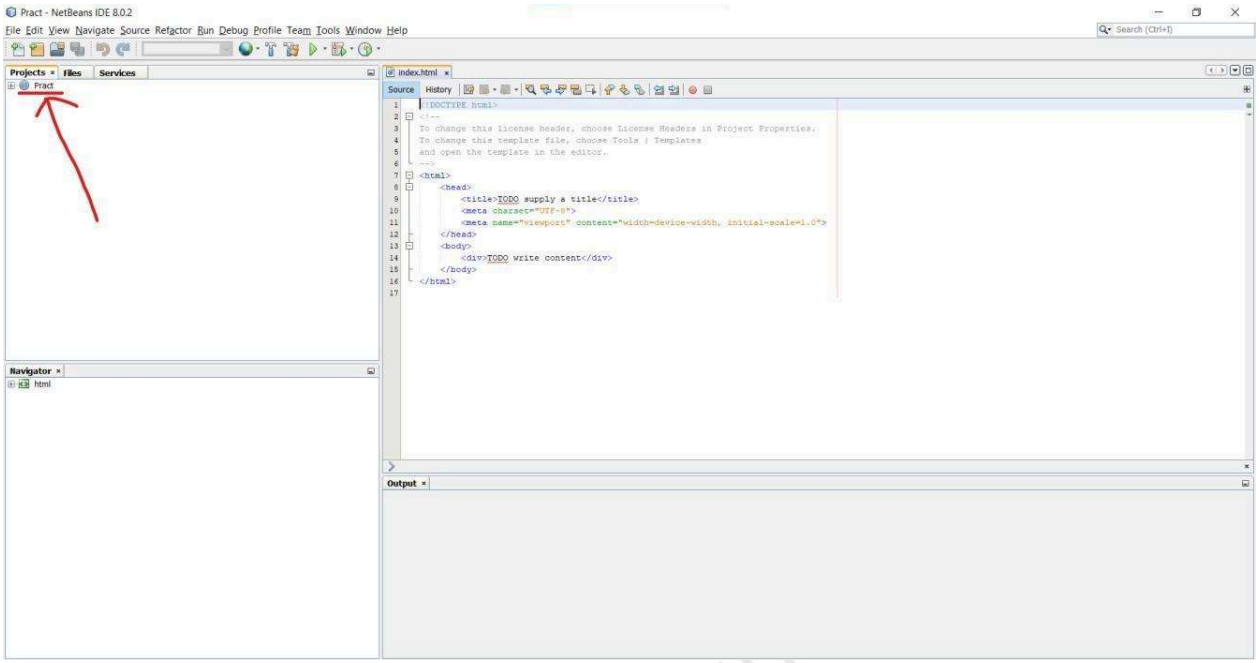
3] In Categories select Java Web and in Projects , Select Web Application. After selecting click onnext . You will get the following window:



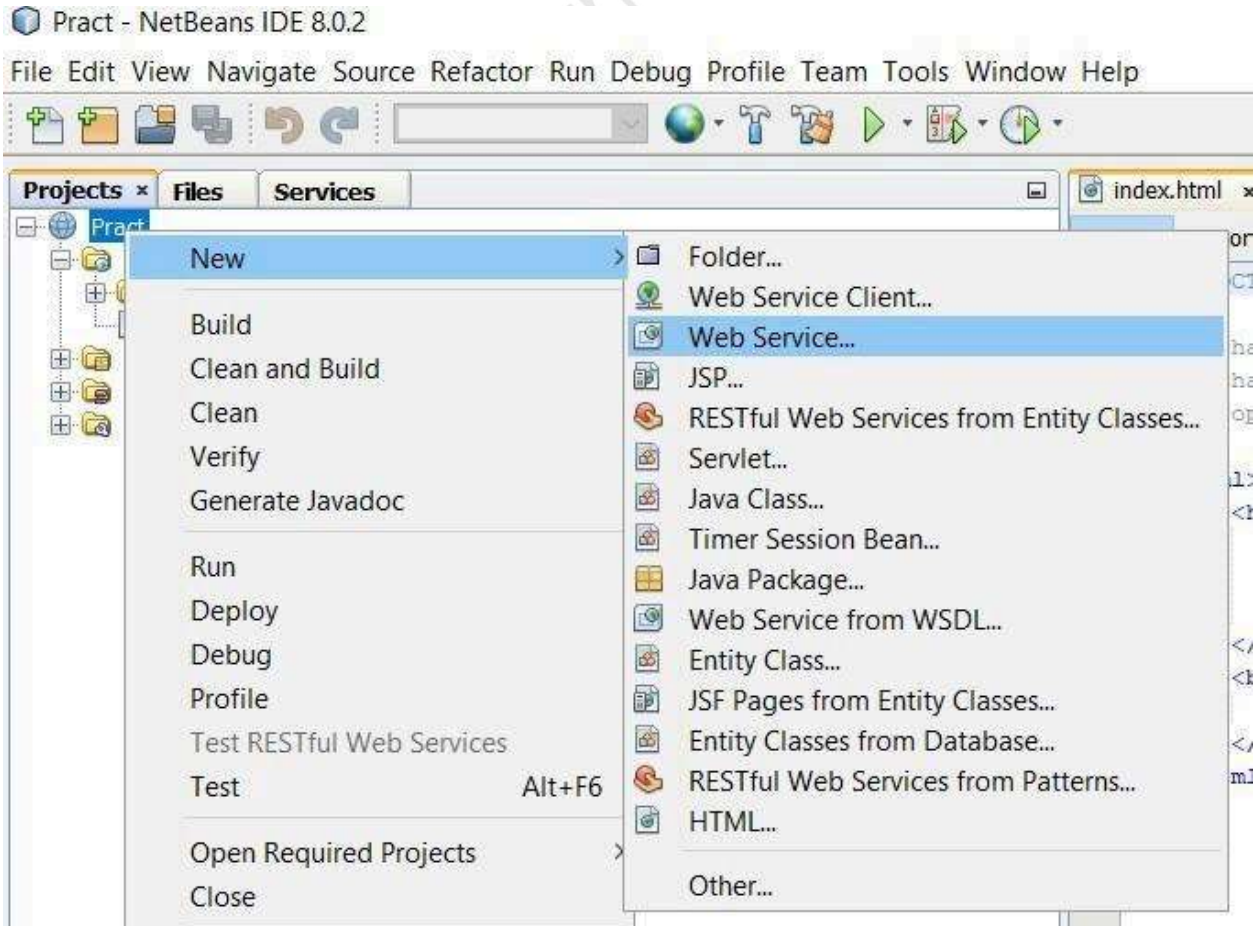
4] Now Give name to the Project Name: , and click on next . you will get the following windowthen . again click on finish:



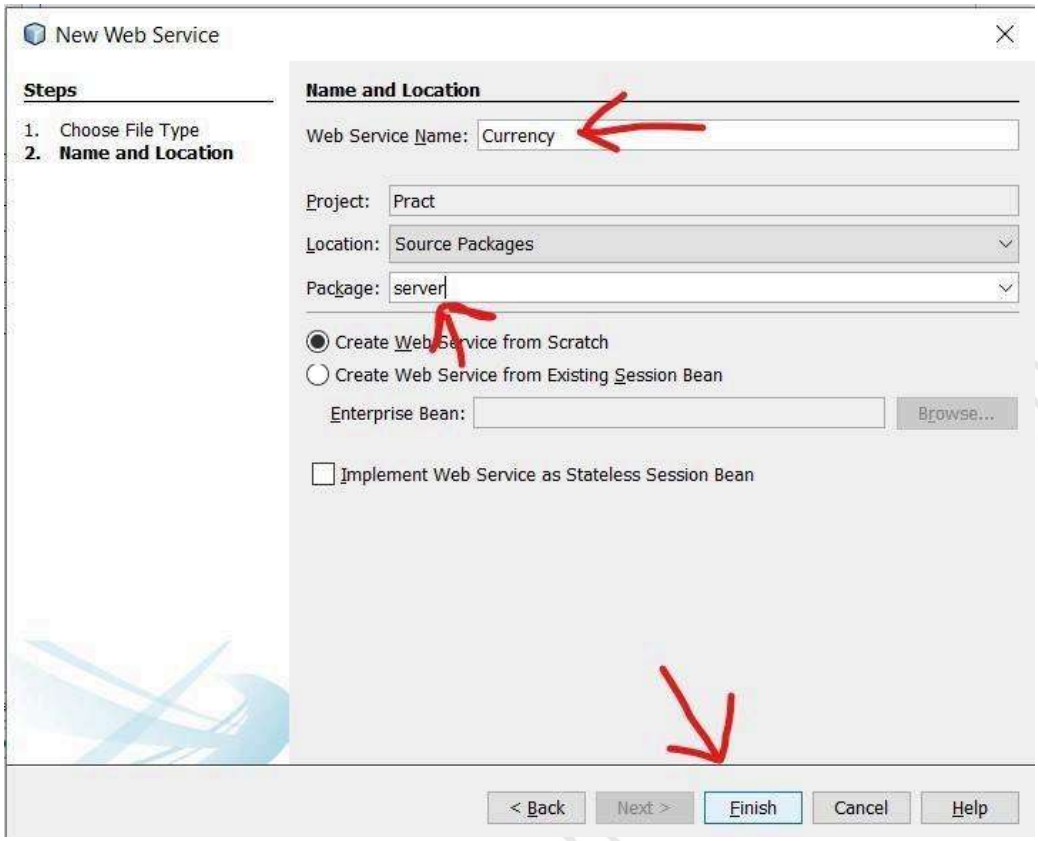
5] You will get the following screen now carefully see in projects section your recently created project appears double click to expand it:



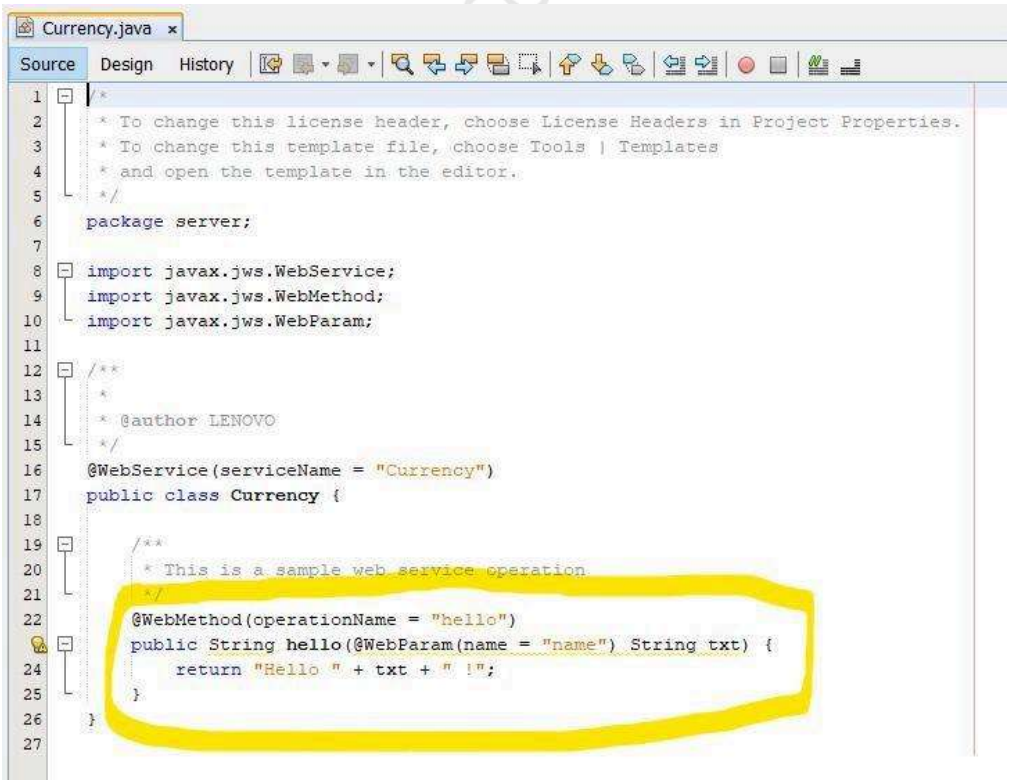
6] Now Right click on the project and select new and then select Web Service ,As shown Below:



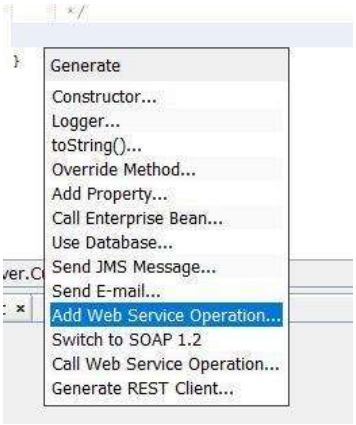
7] after clicking Web Service following window should appear , now give name to web serviceand give package as “server”. As shown in the image:



After clicking finish you should get the following window erase the mentioned code :

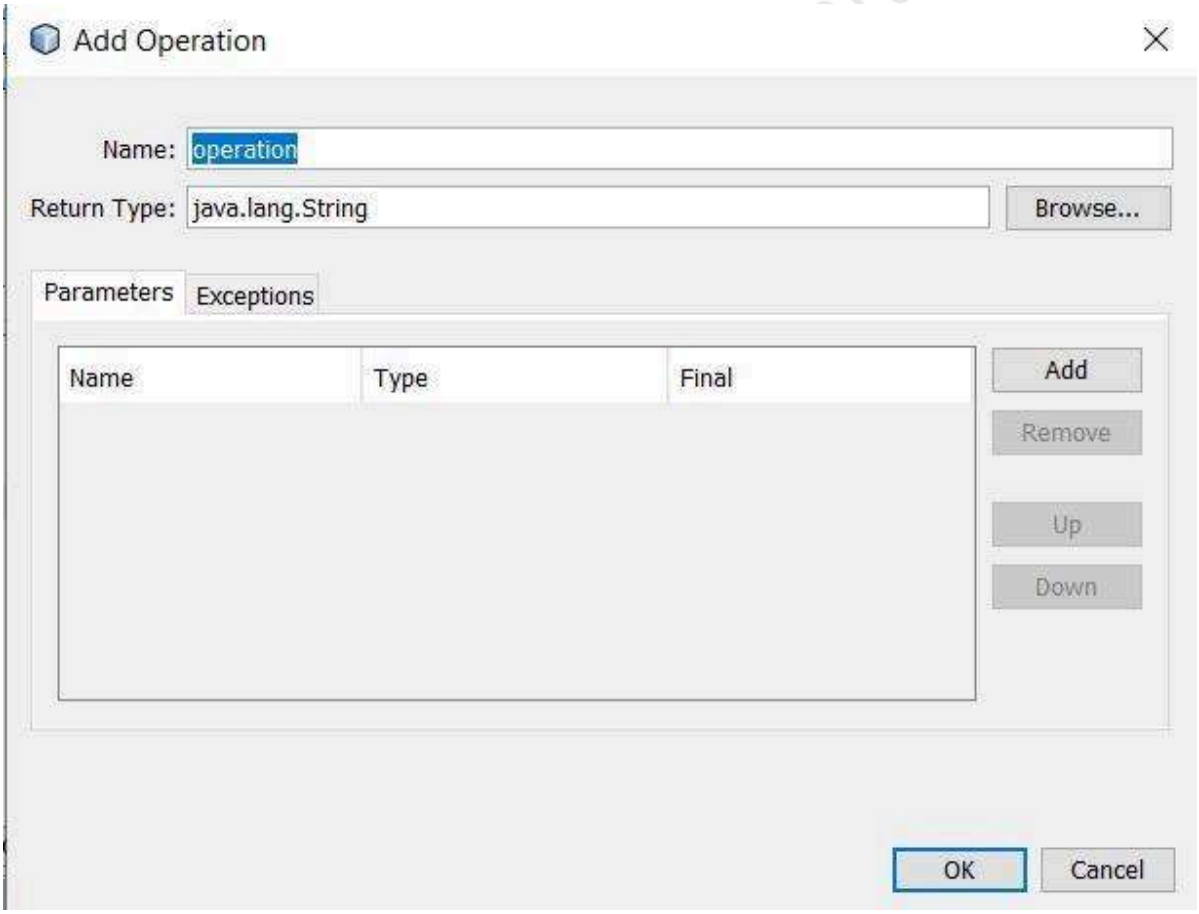


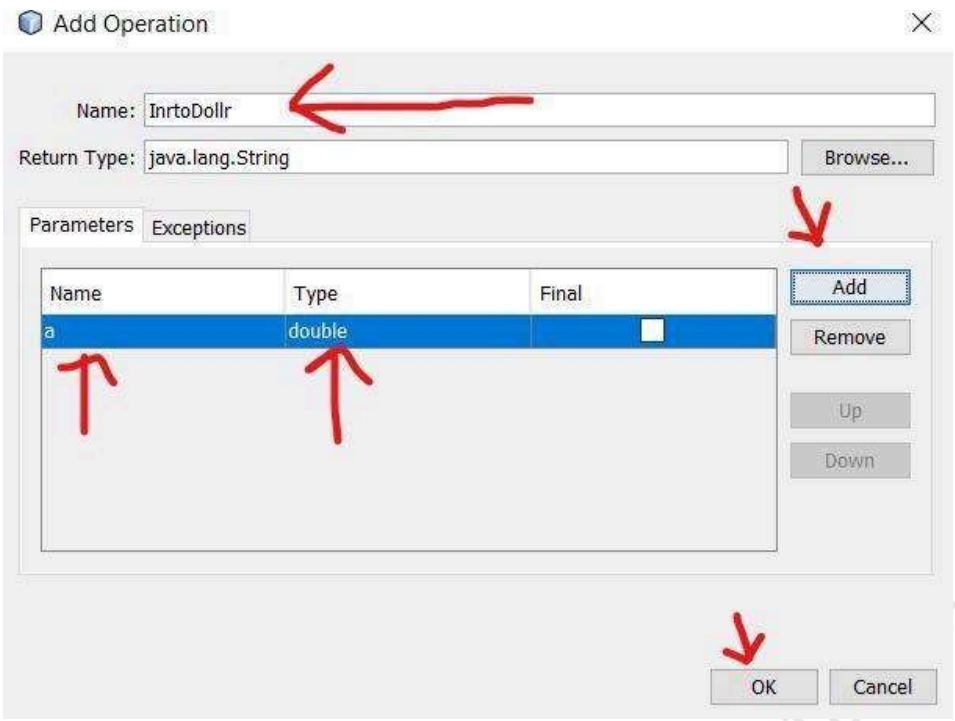
8] Now right click any where and click on insert code and select Add Web Service Operation :



9] the following window would appear :

Just give name to the method or operation and click on add button to add parameters to the method as here we are converting dollar to rupees we should need only one parameter .

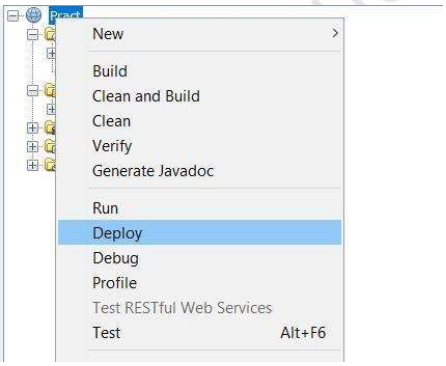


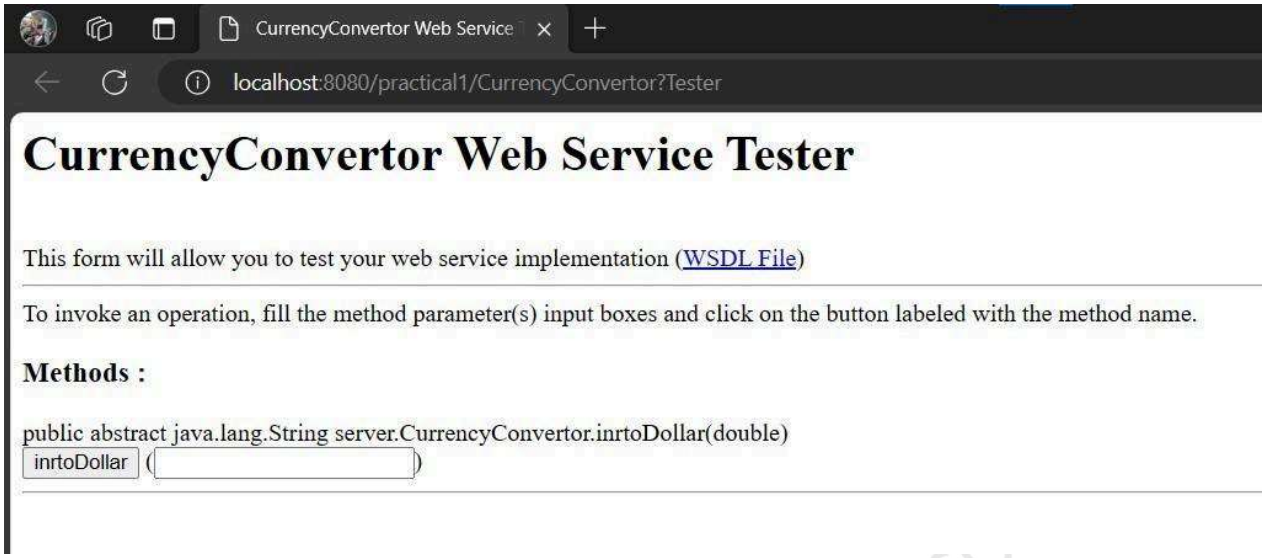


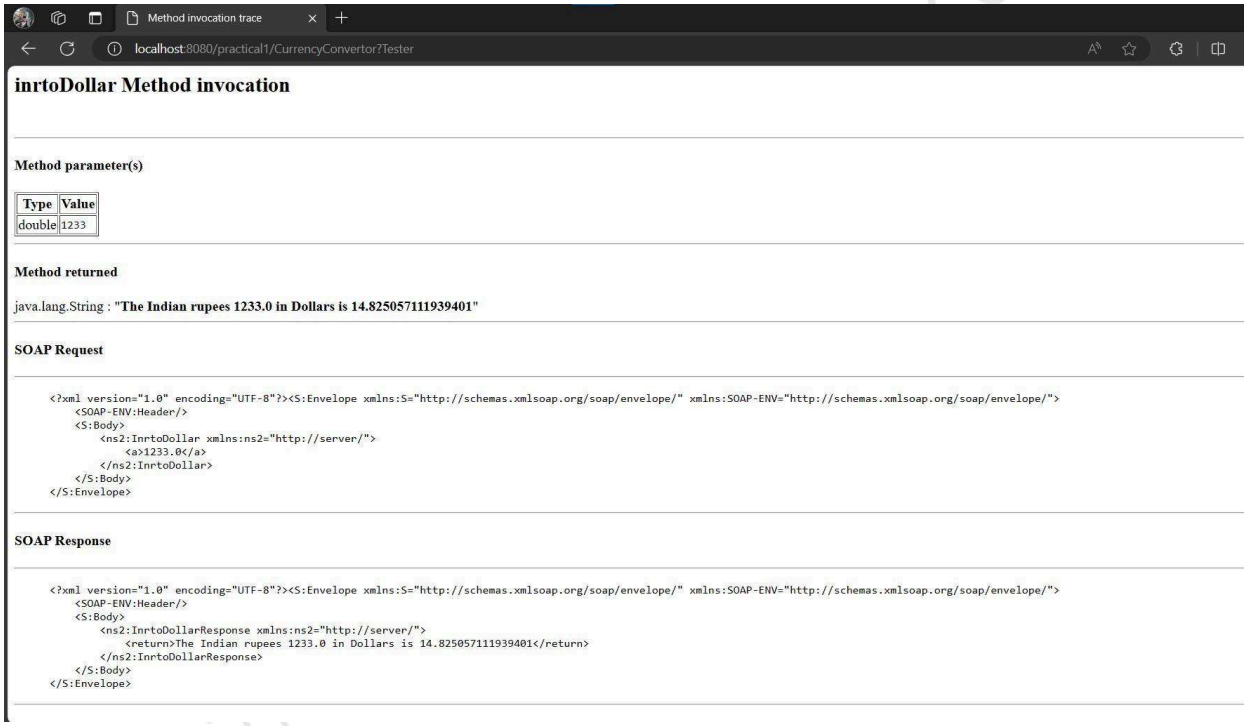
11] After clicking ok code will autogenerate , make changes In that code as mentioned below:

```
@WebMethod(operationName = "InrtoDollar")
public String InrtoDollar(@WebParam(name = "a") double a) {
    //TODO write your implementation code here:
    return "The Indian rupees "+a+" in Dollars is "+(a/83.17);
}
```

12] now our web service is ready now right click on project and click on deploy :



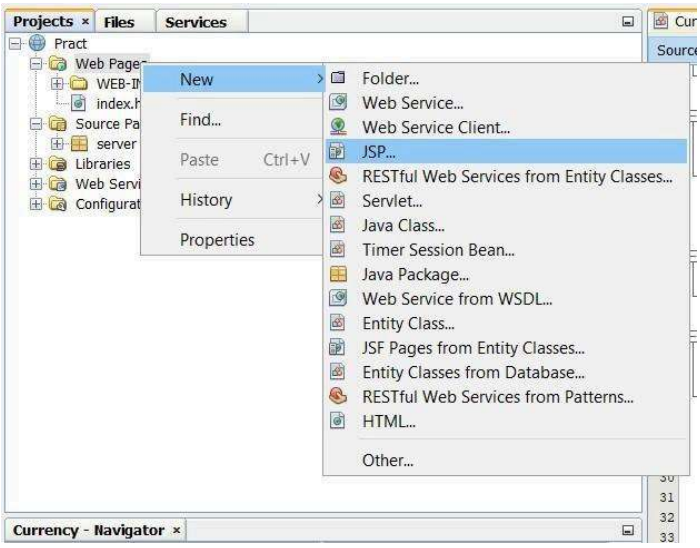




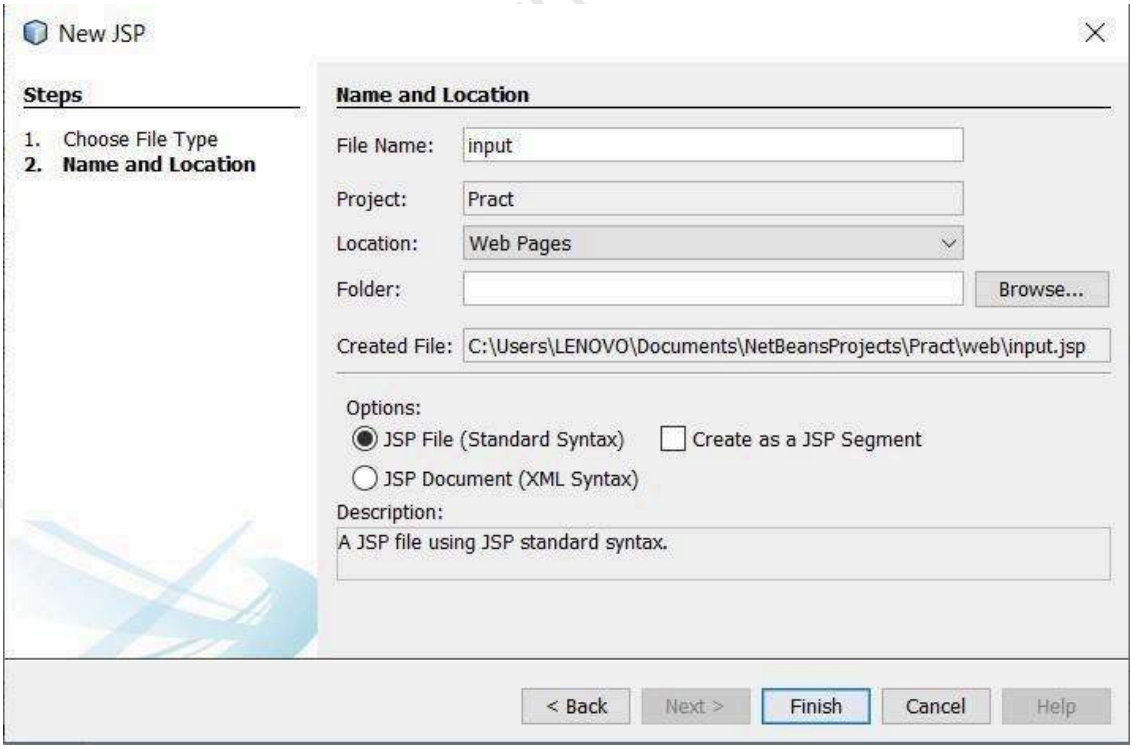
So this is how we created our web service and deployed it.

Creating a java Client using jsp

14] Now our web service is successfully deployed. Right click on web pages and select new and select jsp as shown below:



15] give name and click on finish as shown below do it 2 times on for input and one foroutput:

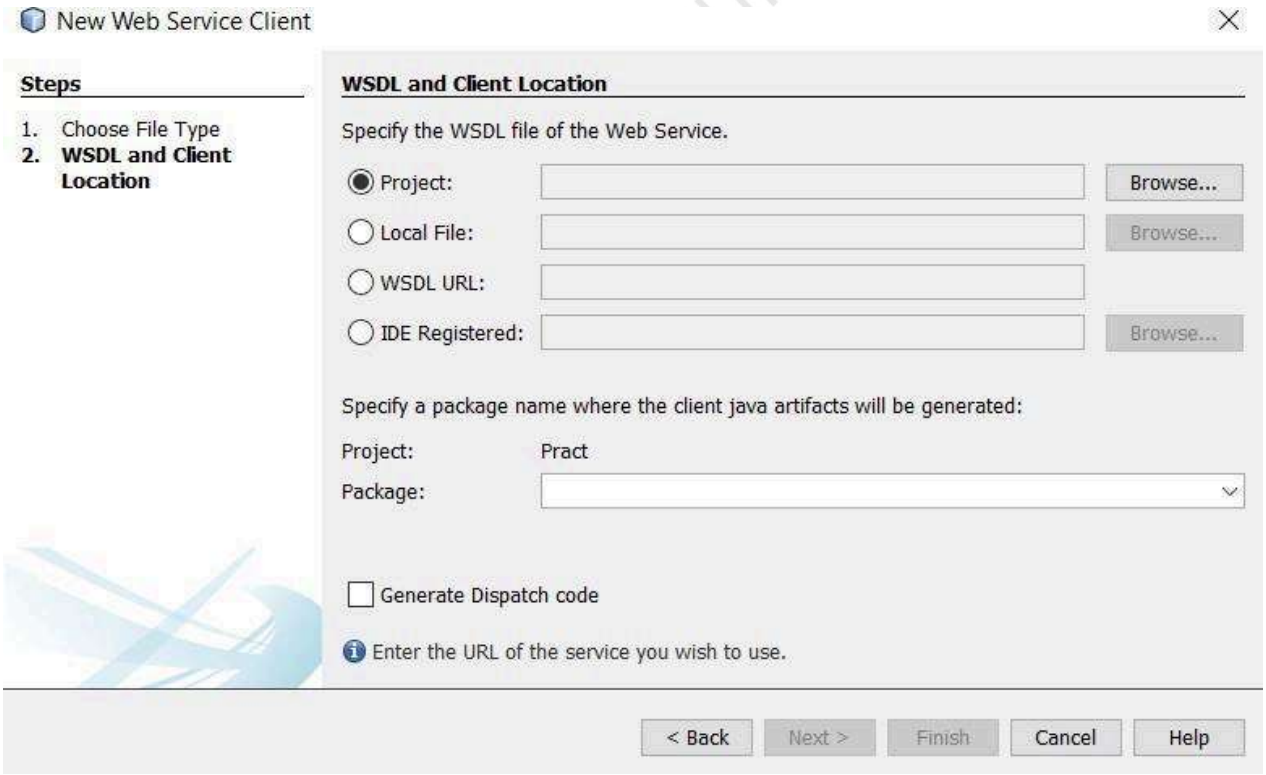


16] In input.jsp create a form for taking user input as shown below:

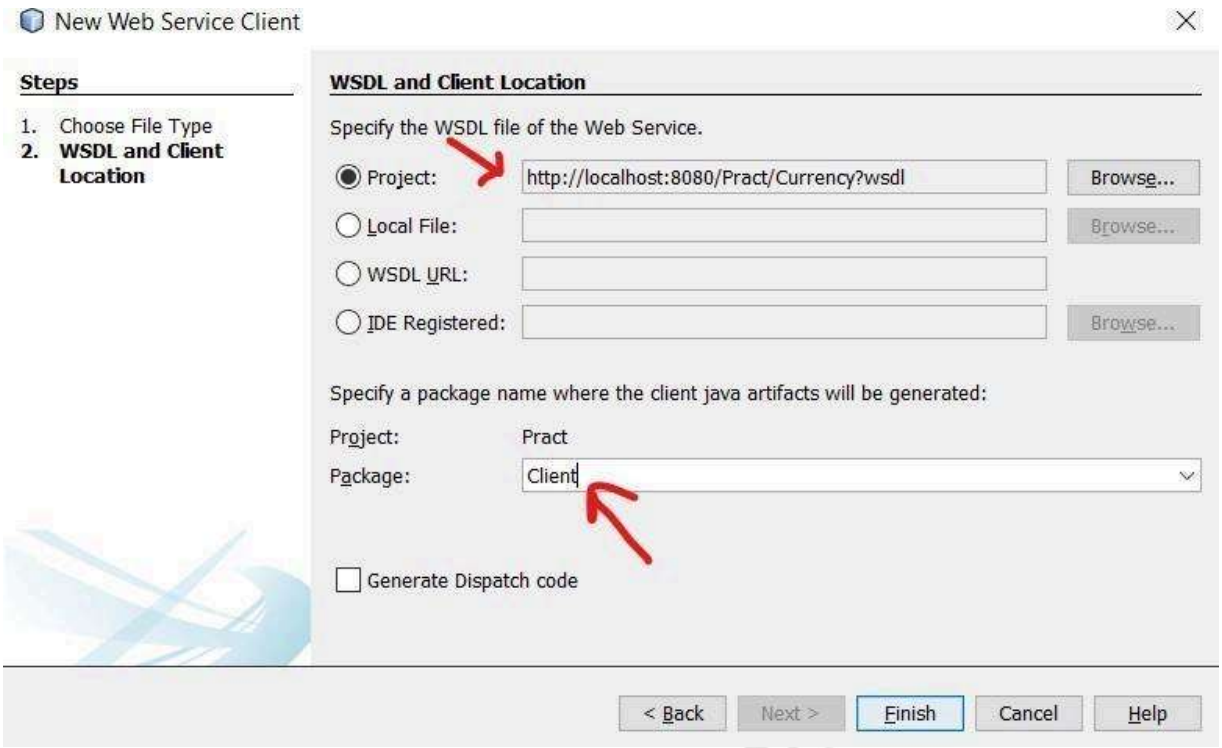
```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="output.jsp">
            <pre>
                Enter the currency in rupees : <input type="text" name="t1">
                <input type="submit">       <input type="reset">
            </pre>
        </form>
    </body>
</html>
```

In this code set action = the jsp file where u want output and give name to textbox input.

17] now we have to create web service client, for same right click on project and select new then select web service client you will get the following screen:



At this step click on browse and select your project and click ok after clicking ok you will get wsdl url as shown below:



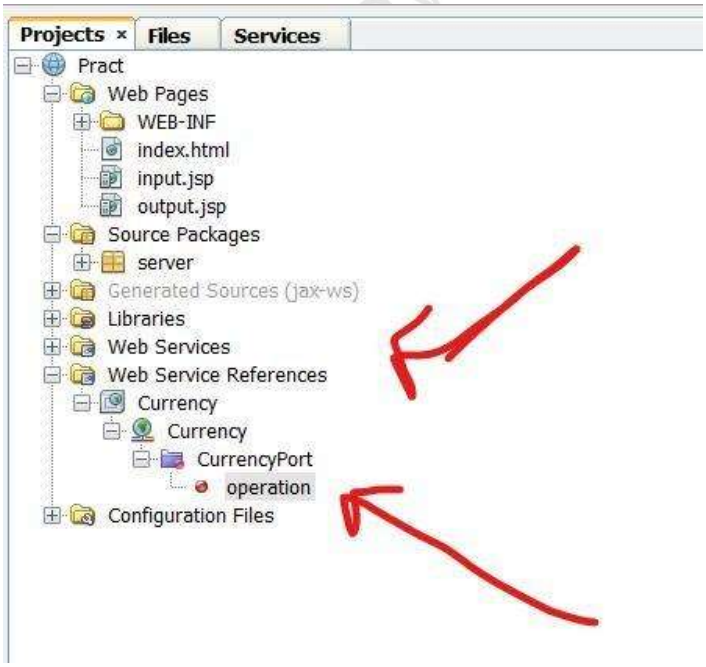
At this stage copy the url and paste it in notepad for future.

Give package name as "Client".

Now click on finish.

18] now in project section you can se a new folder is been created named "web servicerefrence".

double click to expand it you should get the following :



Hold the operation or your operation name and drag it into the output.jsp in body tag as shown : You will get the following auto generated code:

```
<body>
        <%-- start web service invocation --%><hr/>
<%
try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
     // TODO initialize WS operation arguments here
    double a = 0.0d;
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println("Result = "+result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>

</body>
```

19] now make changes as shown below in the code:

```
        <%-- start web service invocation --%><hr/>
<%
try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
     // TODO initialize WS operation arguments here
    double a = Double.parseDouble(request.getParameter("t1"));
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println(result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>
```

20] now Deploy our project again and right click on input.jsp and click run file you will redirect to a browser page as shown :

Enter any numerical value in text box and click on submit you will get the following output:

Python client:

1] for consuming java web services in python we should repeat the above steps till step np.19 .

Note: for getting output the project or web service should be deployed .

After doing all of the steps write the following code in python idle:

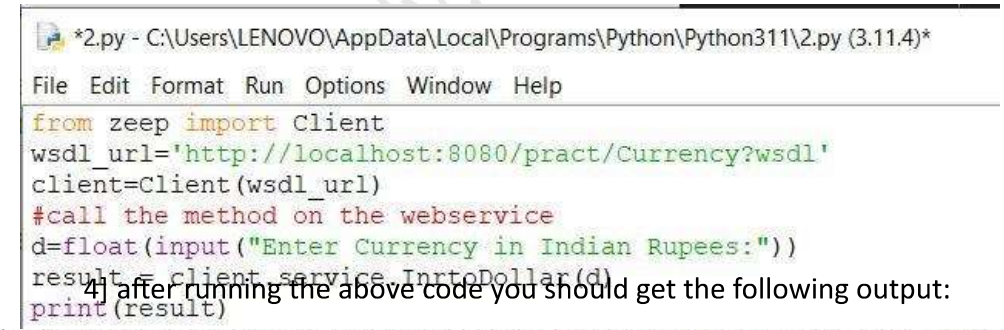
Code :

```
from zeep import Client
wsdl_url='http://localhost:8080/pract/Currency?wsdl' client=Client(wsdl_url)
#call the method on the webservice
d=float(input("Enter Currency in Indian Rupees:"))
result = client.service.InrtoDollar(d)
print(result)
```

2] replace wsdl_url with the url which was copied at the time of web service client creation .

In step no.17 . and replace the InrtoDollarwith your method or operation name. and pass parameter to it

3] final code should look like following:



4] after running the above code you should get the following output:

```
= RESTART: C:\Users\LENOVO\AppData\Local\Programs\Python\Python311\2.py
Enter Currency in Indian Rupees:12000
The Indian rupees 12000.0 in Dollars is 144.28279427678248
```

Practical - 2

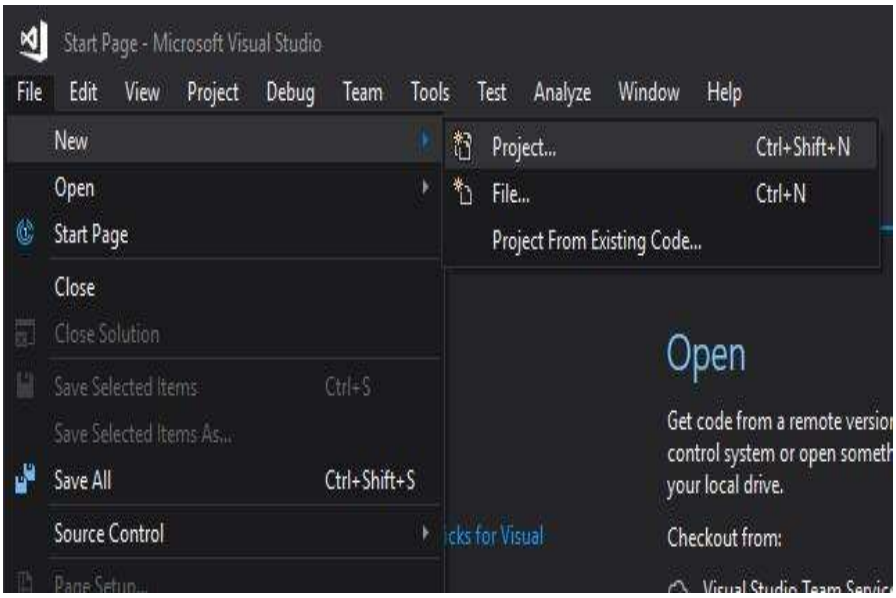
Aim: Write a program to implement a simple SOAP WebService

Requirement:

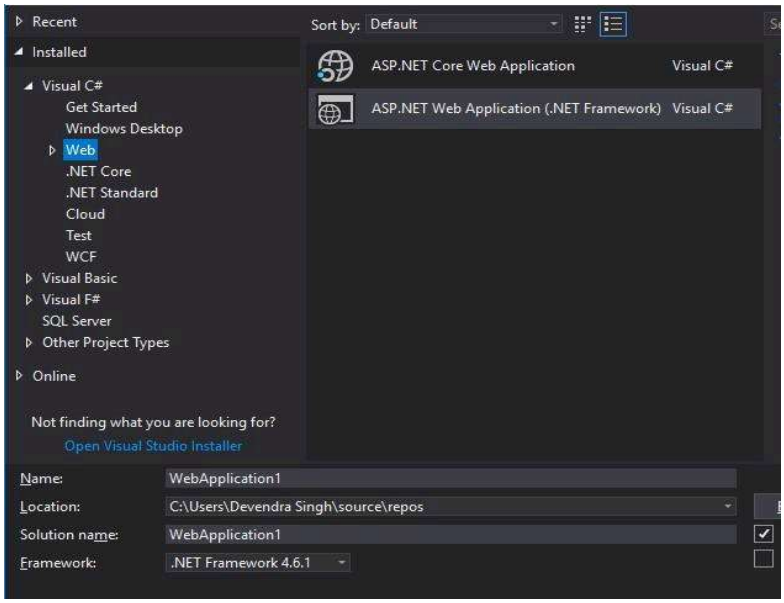
1. Visual Studio Community 2017
2. Version : 15.8 or latest

In this practical we are creating a Web Service in Visual Studio and then we will consume it in NetBeans.

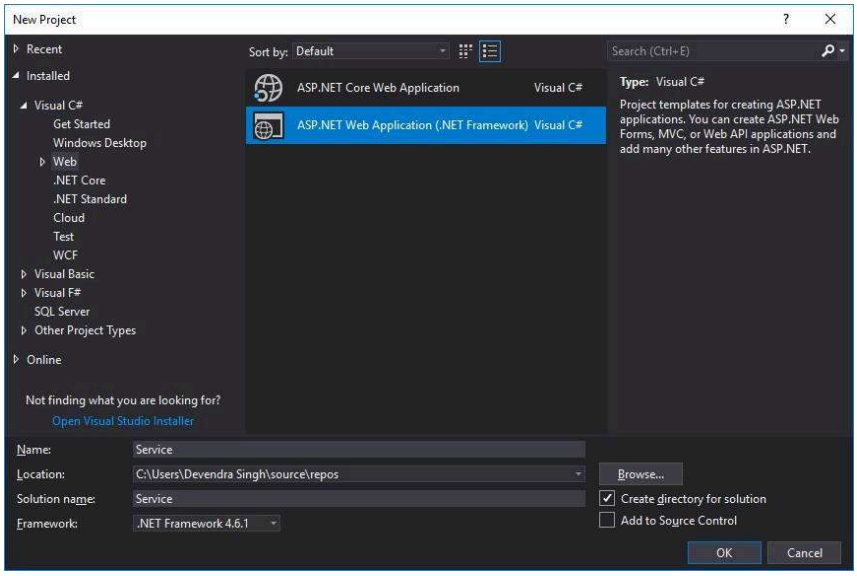
1. Open Visual Studio IDE and click on File.
   File -> New -> Project



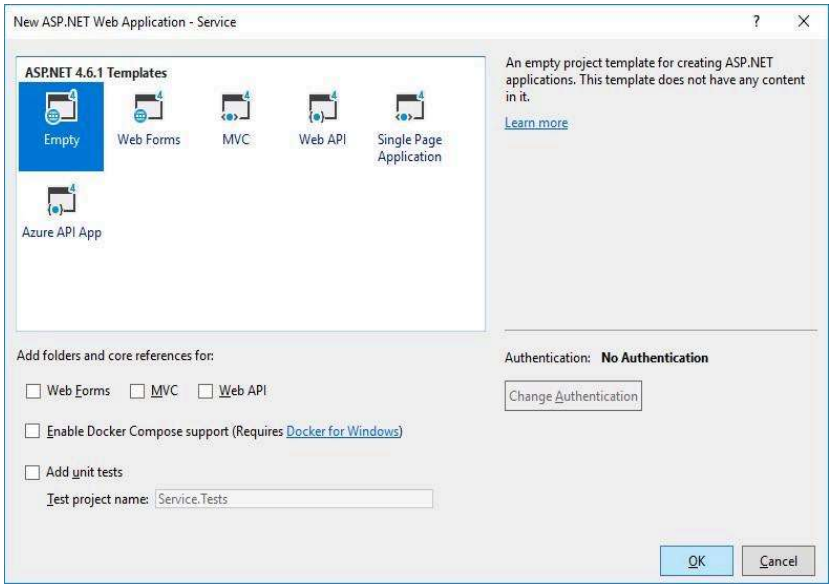
2. Click on Web.



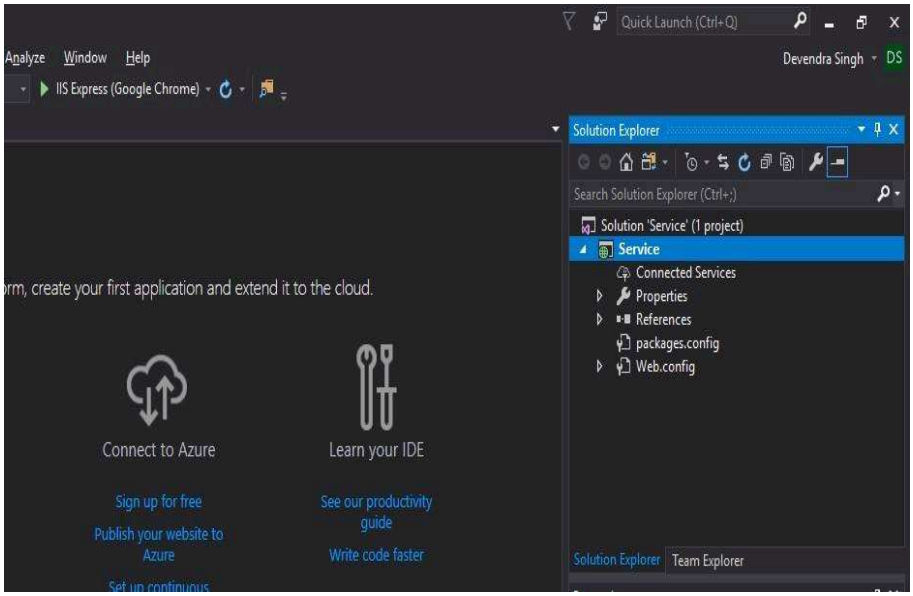
3. Select ASP.NET Web Application and give Name as Service. After that click on OK button.



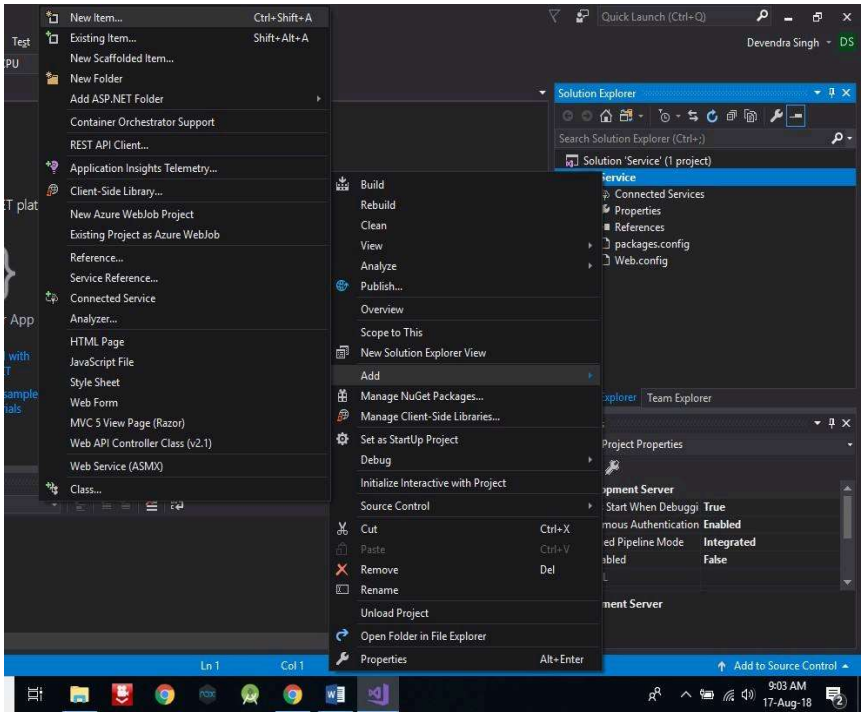
4. Select Empty and click on OK button.



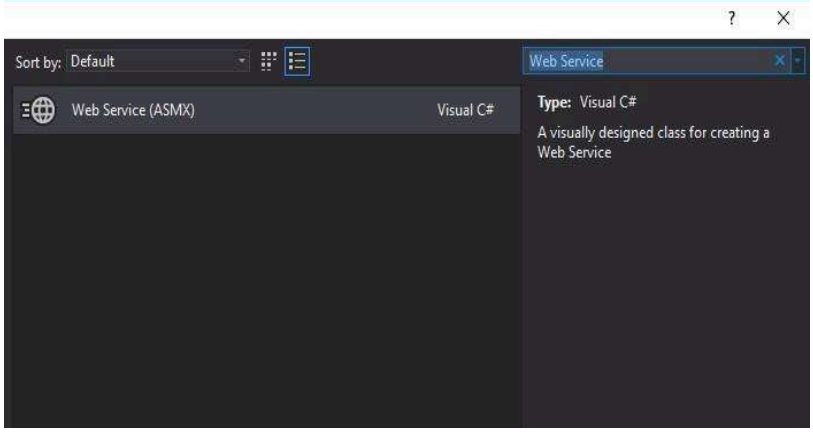
5. Now you can see, on right side in Solution Explorer Service project is created.



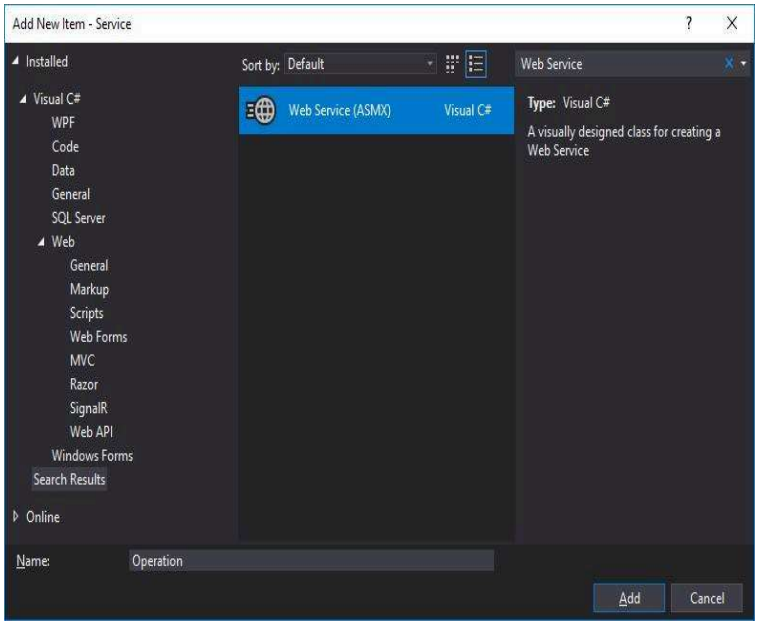
6. Right click on Service -> Add -> New Item.



7. Search for Web Service.

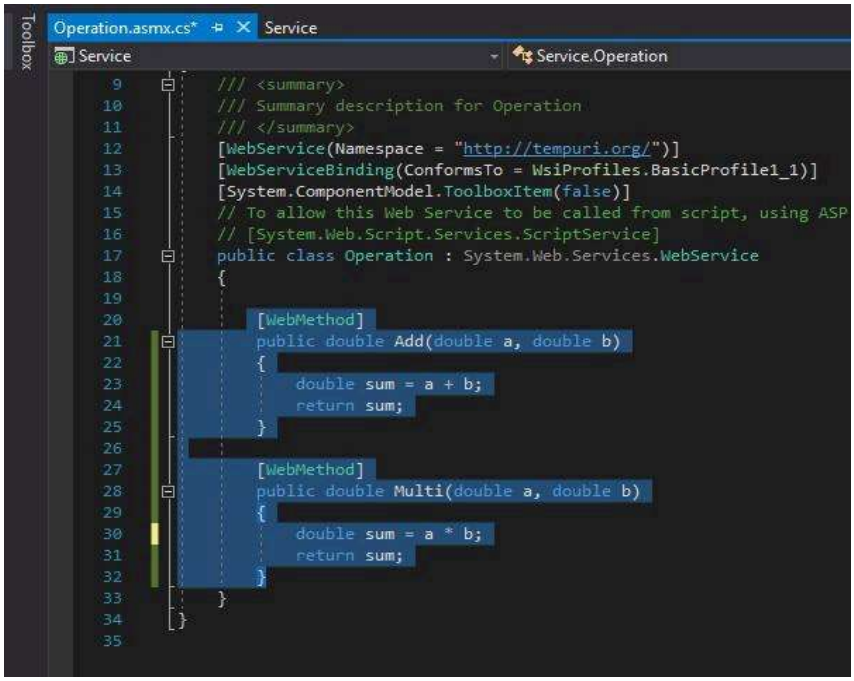


8. Select Web Service and give Name as Operation. After then click on Add button.



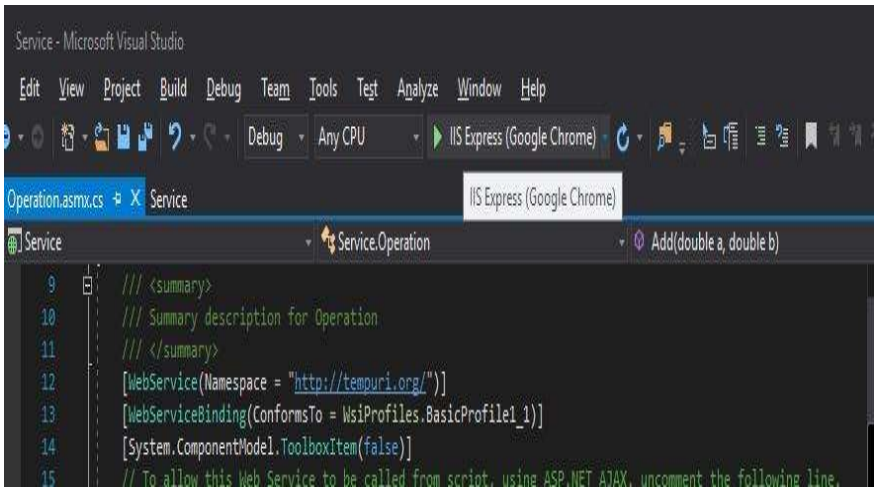
9. After click on Add button, Operaion.asmx.cs file will be automatically open otherwise open it from Solution Explorer and Add the following code into Class Operation.

```
    [WebMethod]
public double Add(double a, double b)
    {
        double sum = a +
        b;          return
        sum;
    }

    [WebMethod]
    public double Multi(double a, double b)
    {
       double m = a * b;
       return m;
}
```

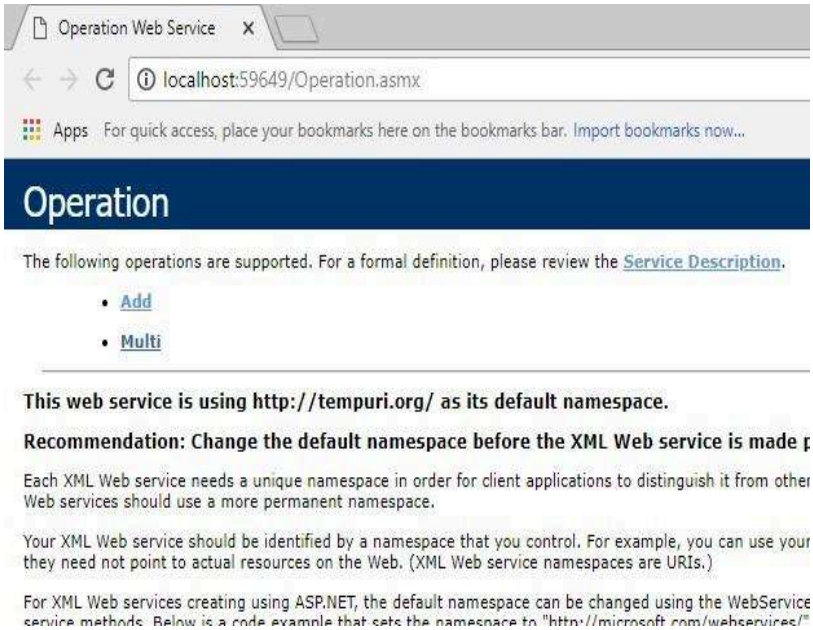


After that press Ctrl+S to save the methods. Actually we are creating two methods for Web Service. One is for addition of two numbers and second one is for multiplication of two numbers.

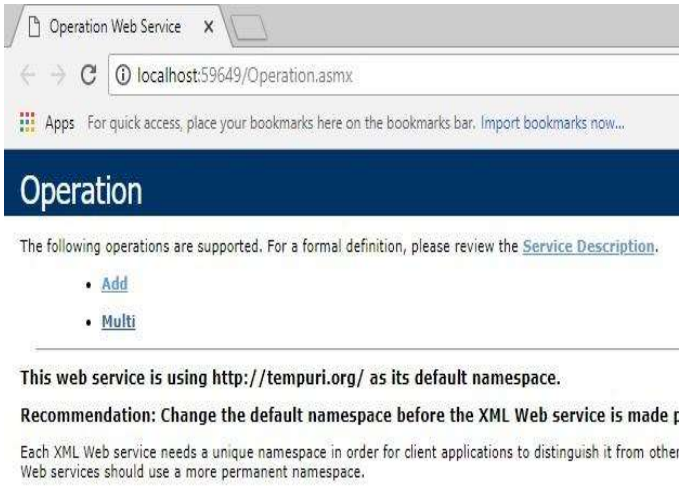
10. Now run the project by click on Green arrow button below the Window menu.



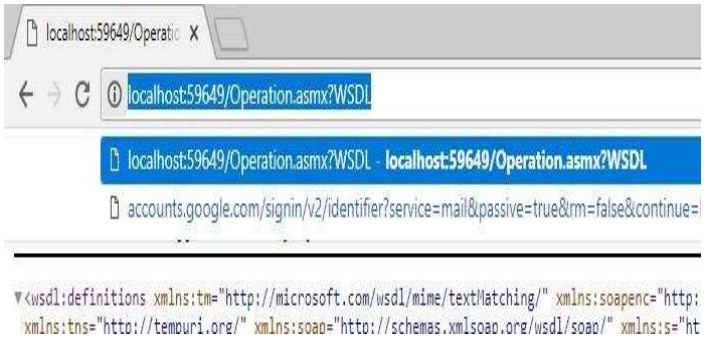
11. A window will open in browser and that is our web service.



12. You can check services by click on Add or Multi option. But we don't need this.
13. Now click on Service Description option.



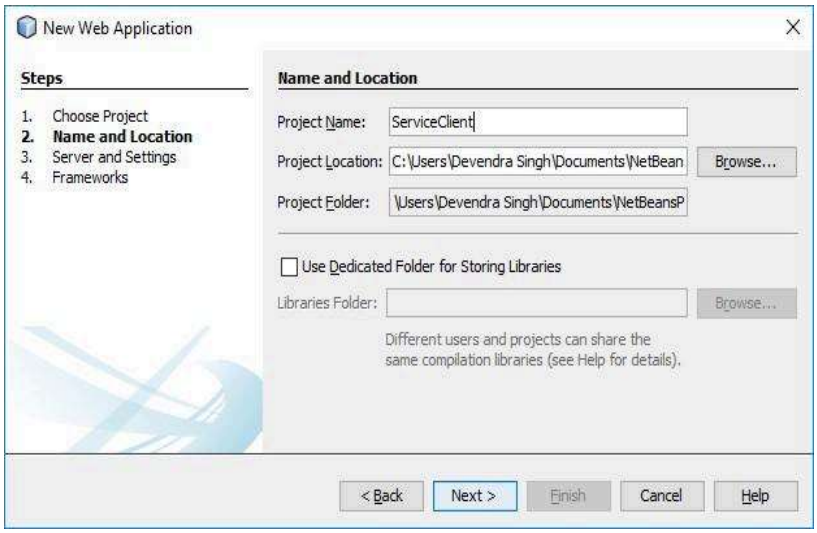
14. Now a new window will open. Select the link and copy it. We will use this link in NetBeans to consume these services.



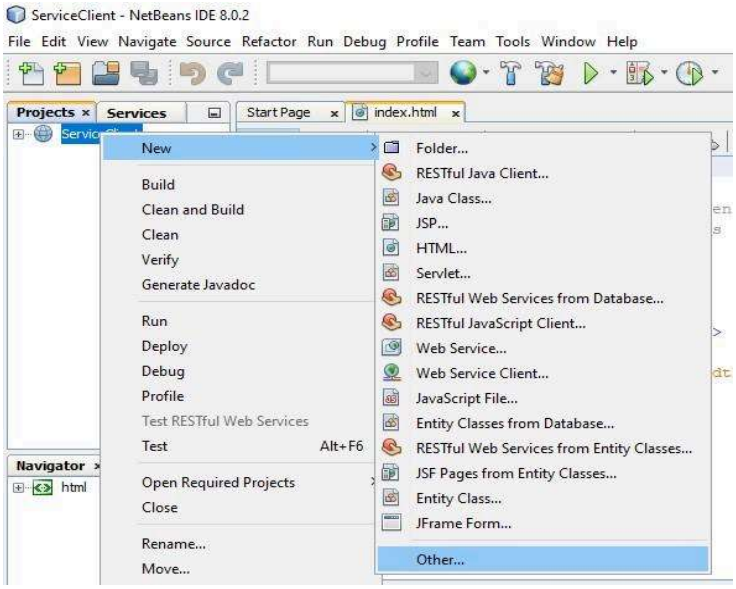
15. Don't close Visual Studio and browser, just minimize it otherwise server will stop. But save the link anywhere, so that we can use it later.

16. Now open NetBeans.

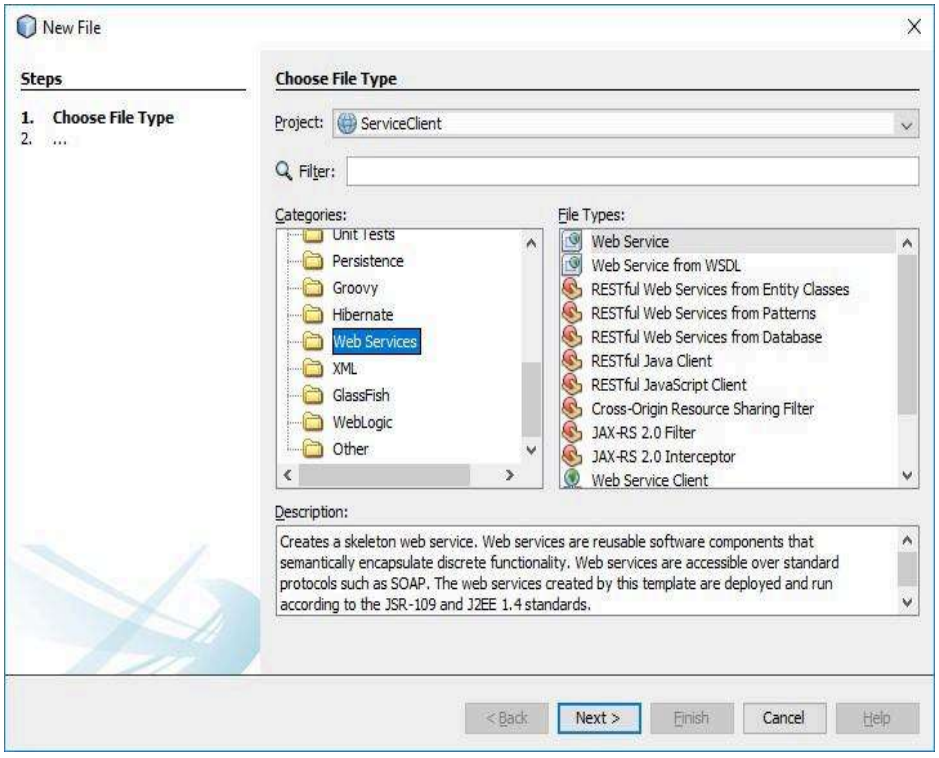
17. Create a Web Application with name ServiceClient. Next -> Finish.



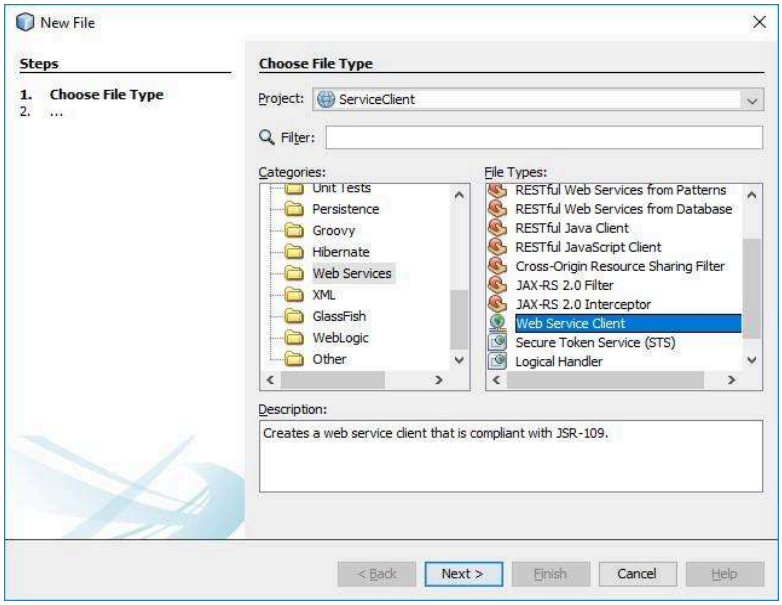
18. Now create a Web Service Client.
Right click on ServiceClient -> New -> Other.



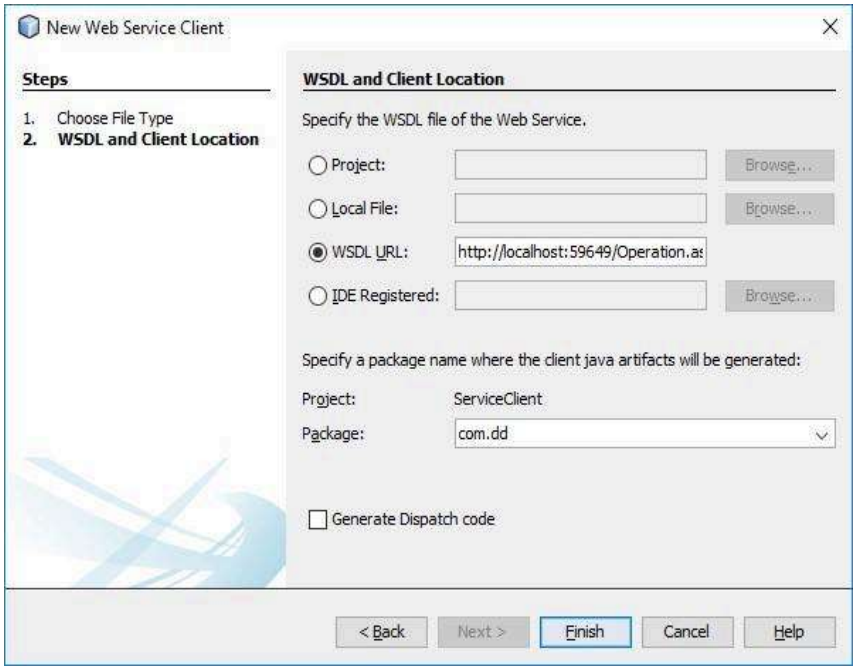
19. Now select Web Services in Categories section.

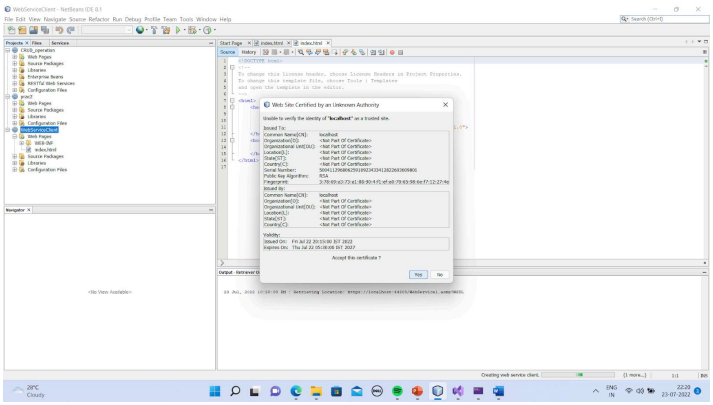


20. Select Web Service Client in File Types and Click on Next button.

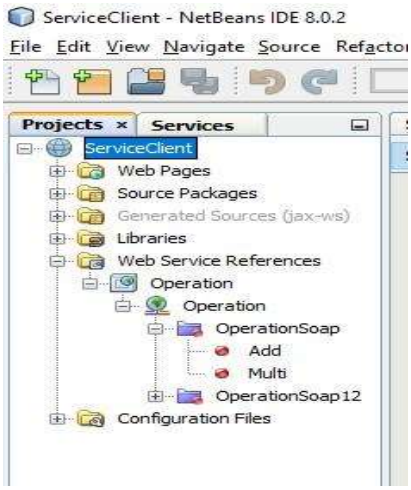


21. Select WSDL URL and paste the link that you have copied from browser on run of Visual Studio enter package name com.dd. After that click on Finish button.

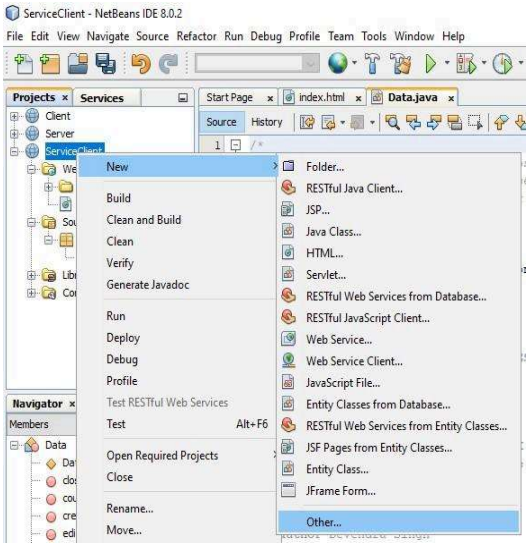




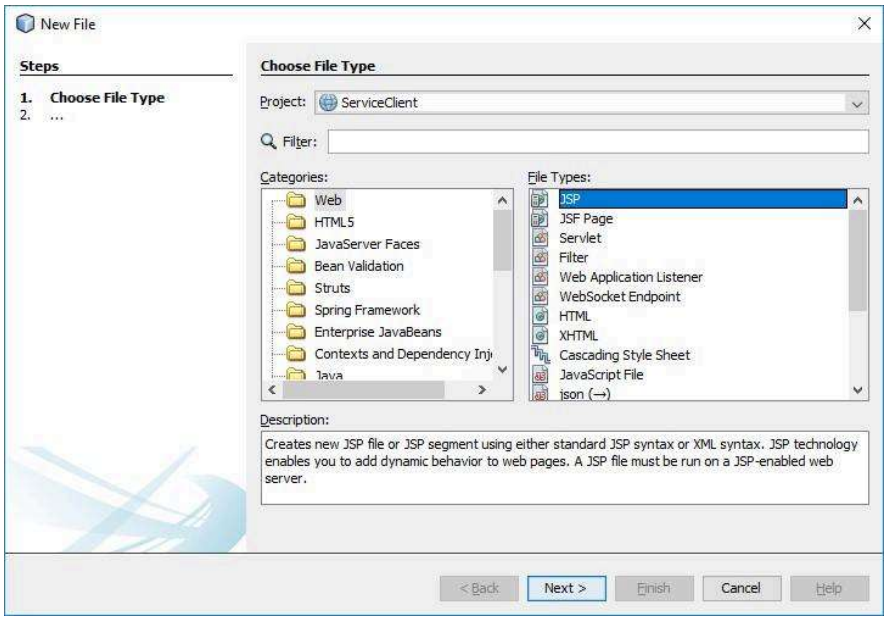
22. As you can see, we got both the service methods i.e. Add & Multi. But in this practical I'm going to use only one service method. You can do for both also.



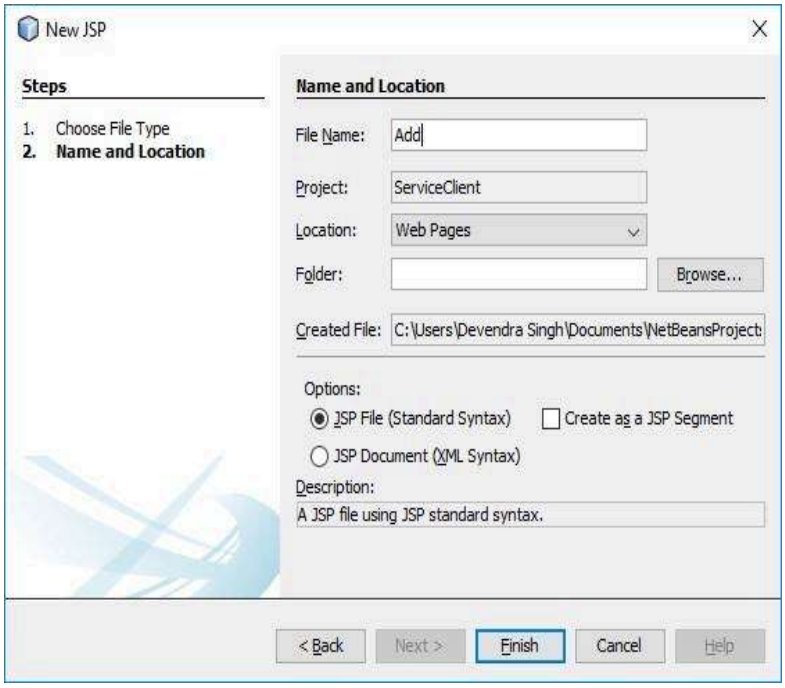
23. Now create a JSP page. Right click on ServiceClient -> New -> Other



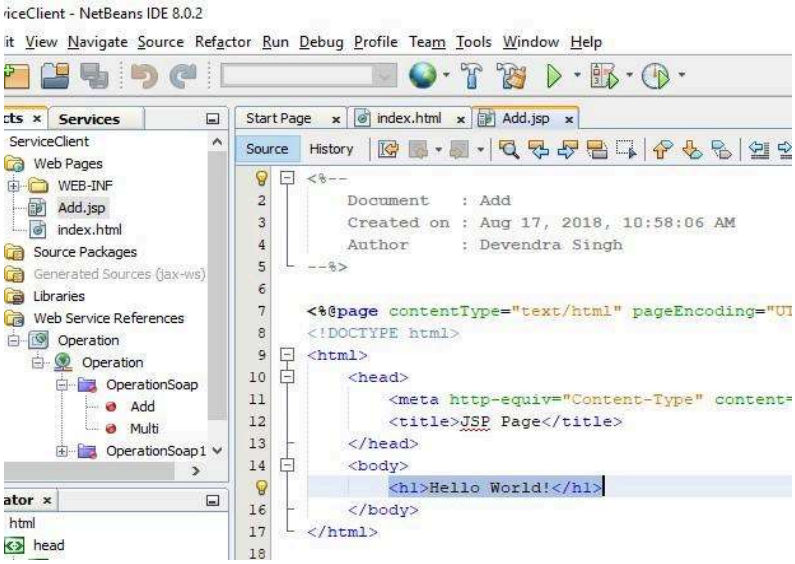
24. Select Web in Categories section -> Select JSP and click on Next button.



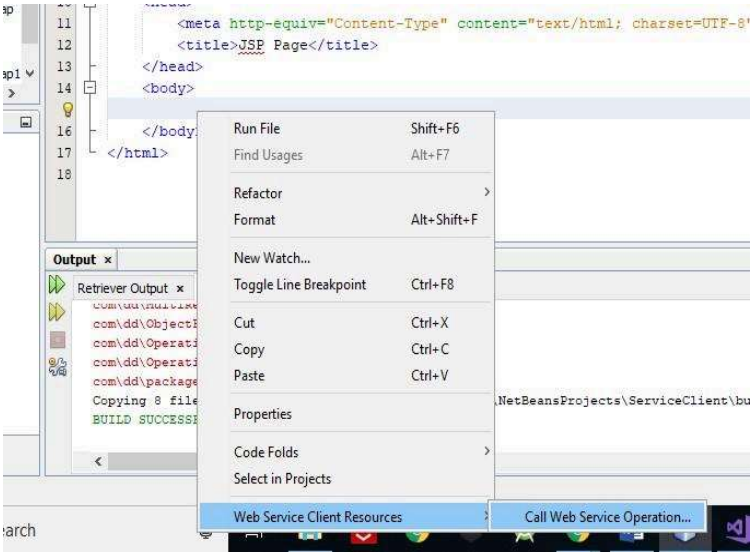
25. Enter File Name Add and click on Finish button.



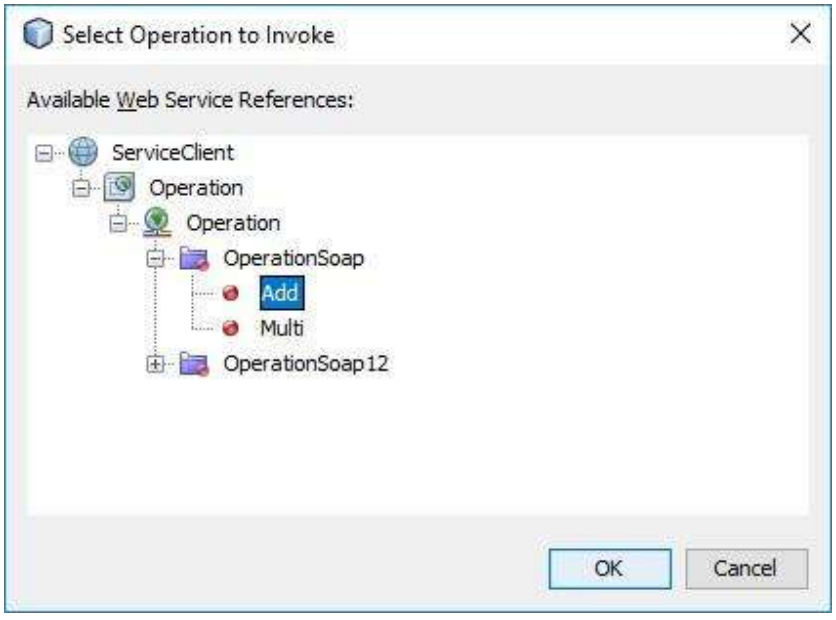
26. In Add.jsp file delete the selected part in body tag, because we don't need this.



27. Right click between body tag and select Call Web Service Operation.

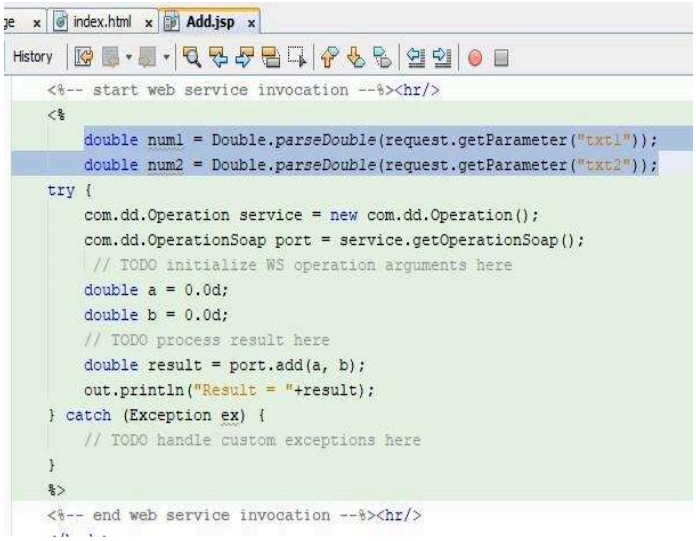


28. Expand and select Add. After select, click on OK button.

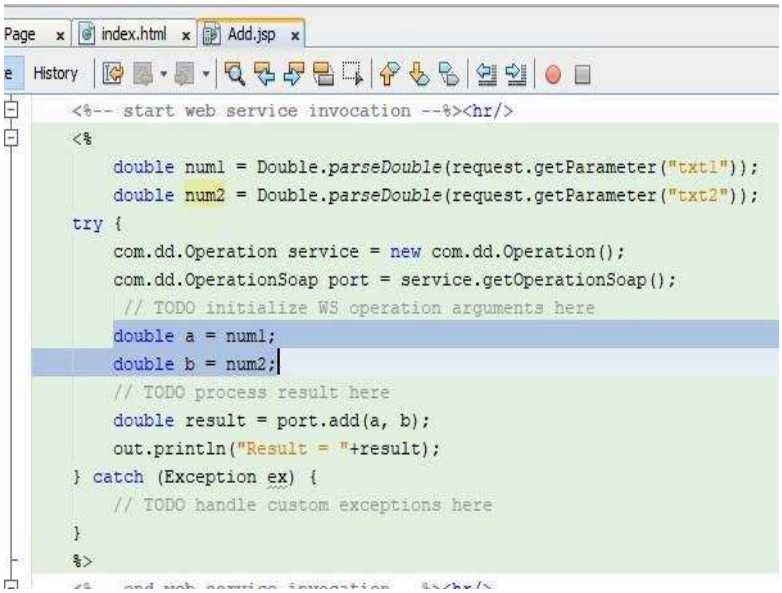


29. Now add the following code outside of try block.

```
double num1 = Double.parseDouble(request.getParameter("txt1"));
double num2 = Double.parseDouble(request.getParameter("txt2"));
```



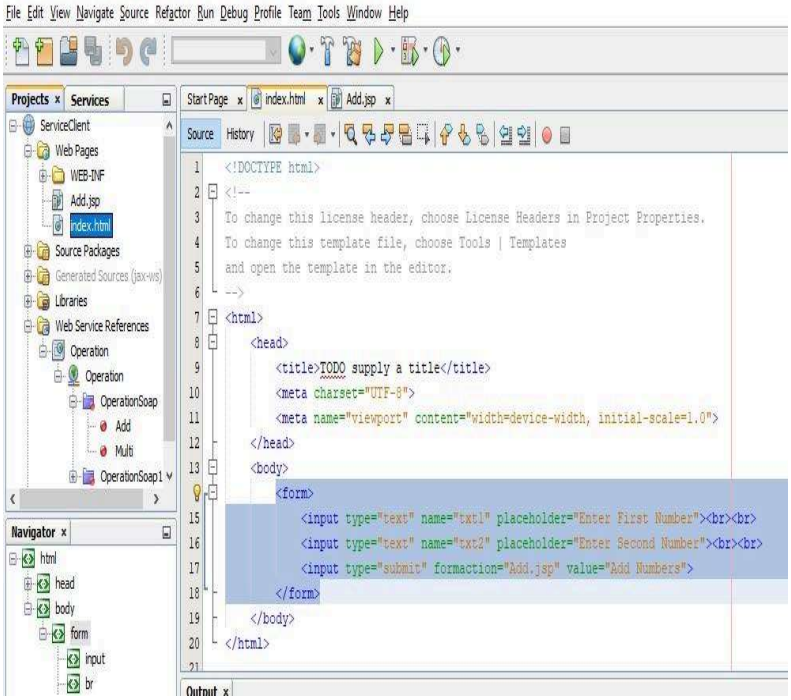
30. Now pass num1 & num2 to a & b variable respectively. After that press Ctrl+S to save this code.



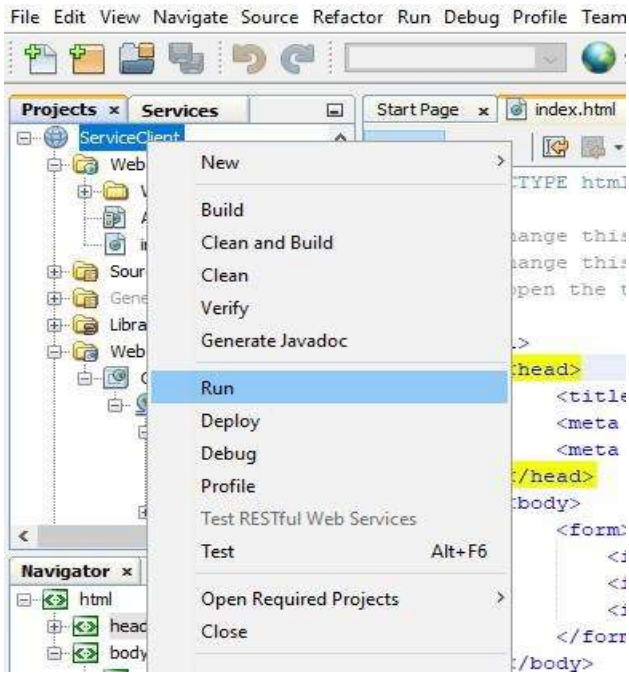
31. Now open index.html file of ServiceClient project and replace the contents of body tag with following code. After that press Ctrl+S to save it.

```
<form>
        <input type="text" name="txt1" placeholder="Enter First
        Number"><br><br>
        <input type="text" name="txt2" placeholder="Enter Second
        Number"><br><br>
```

```
<input type="submit" formaction="Add.jsp" value="Add Numbers">
</form>
```



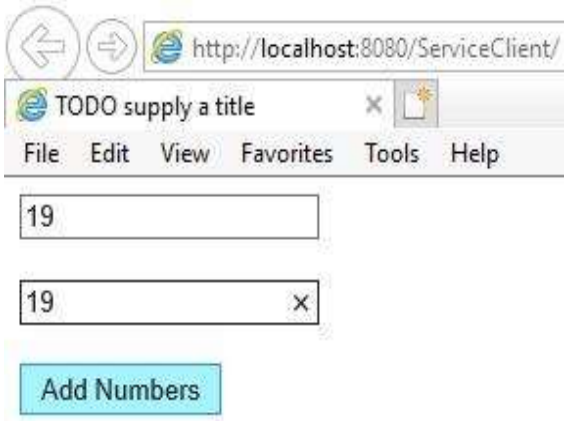
32. Now run the ServerClient web application.

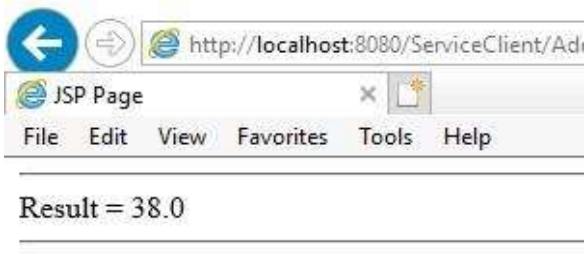


33. A window will open in browser as below.

34. Now enter two numbers and click on Add Numbers button. Wait to get result……..

You will get result on a new page.

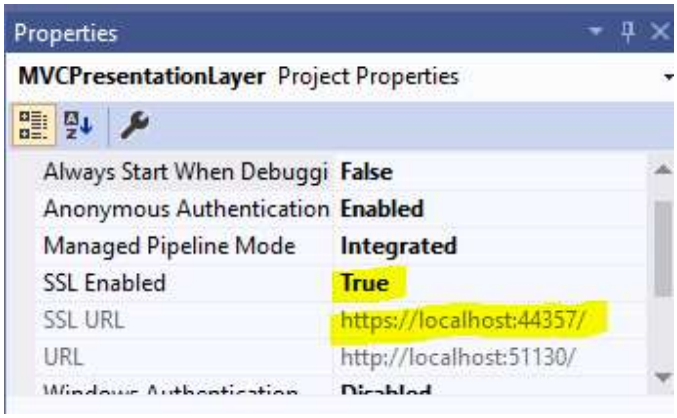


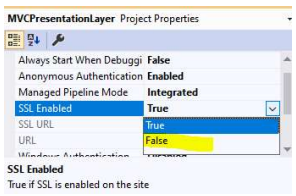


Note : In visual studio 22 if you don’t get the answer follow the following steps

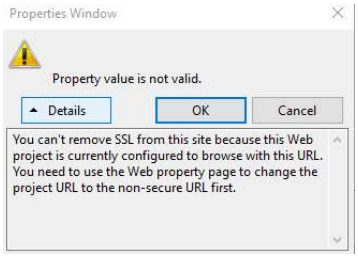
The Solution

To resolve this issue, just follow below steps. We can remove SSL from web properties. If you try to set property "SSL Enabled" to false it will throw below error.

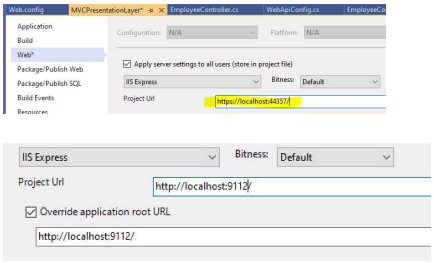




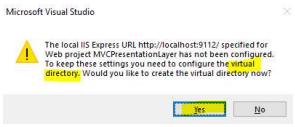
You can't remove SSL from this site because this Web project is currently configured to browse with this URL. You need to use the Web property page to change the project URL to the non-secure URL first.



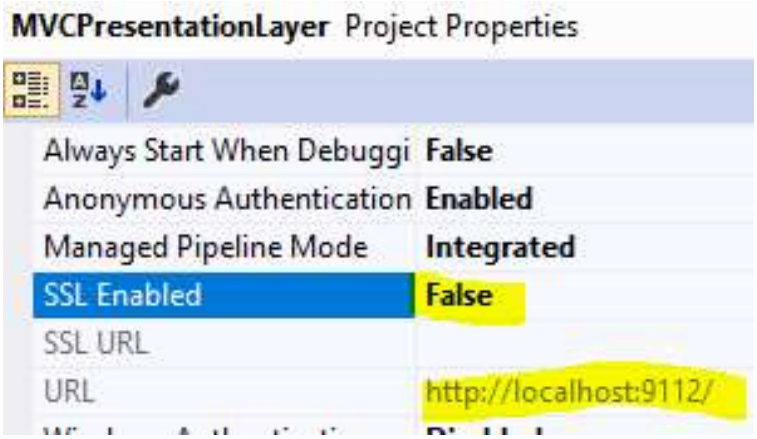
For that first open web property and change "https" to "http"

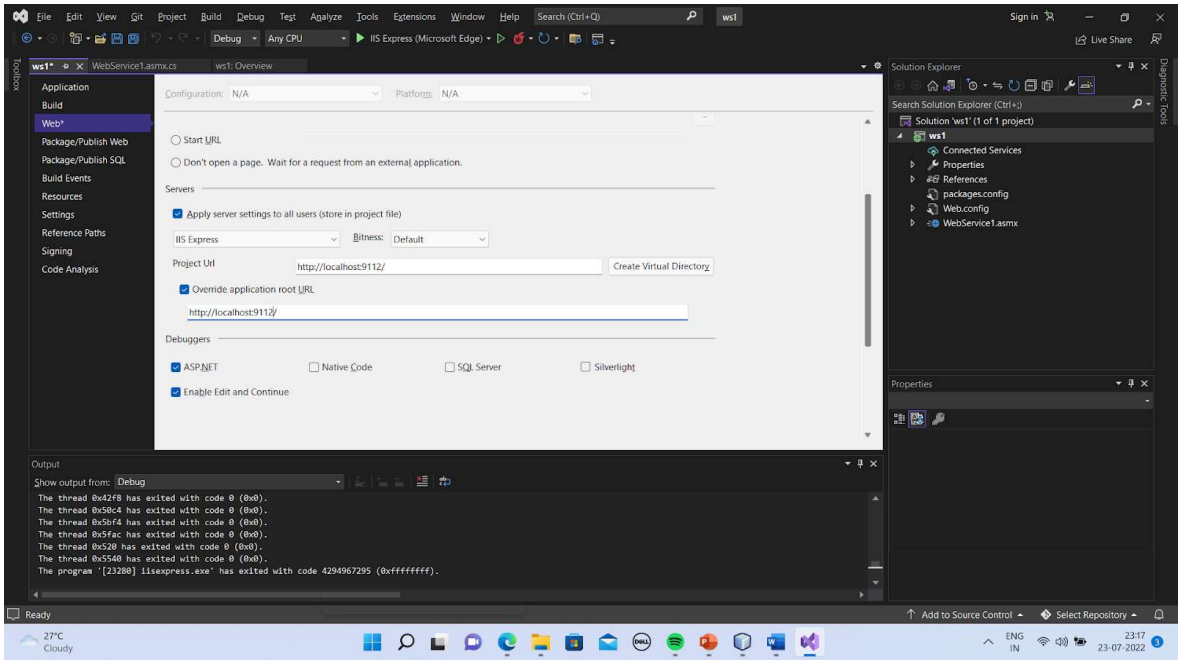


It will popup a message and ask to create a virtual directory to configure the application. Click "Yes" to allow.



Once setting are saved, you can disable the SSL from project property and you are done.





Practical - 3

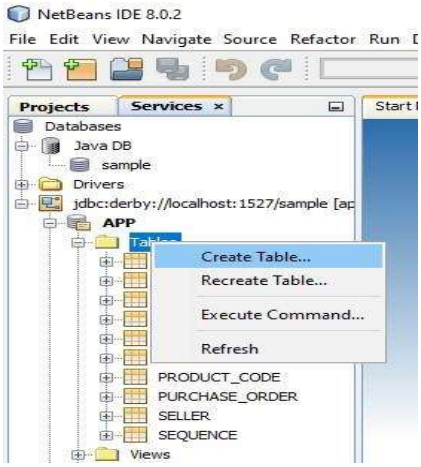
Aim: Write a program to implement RESTFull WebService.

1. Click on Window menu and click on Projects, Files & Services to open it.

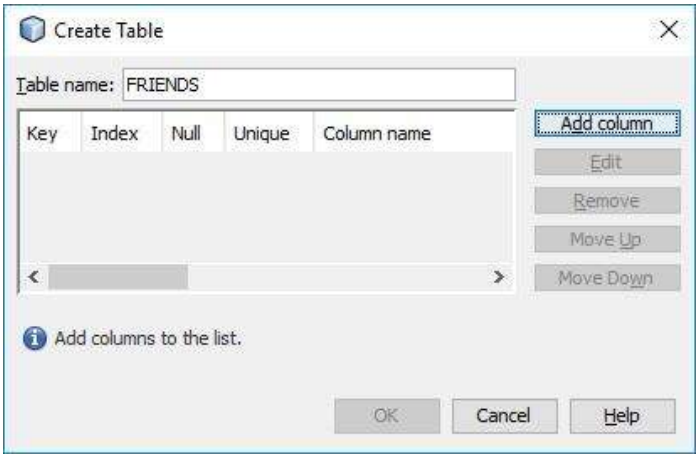
2. Right click on Java DB and then click on Start Server to start the server .

3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.

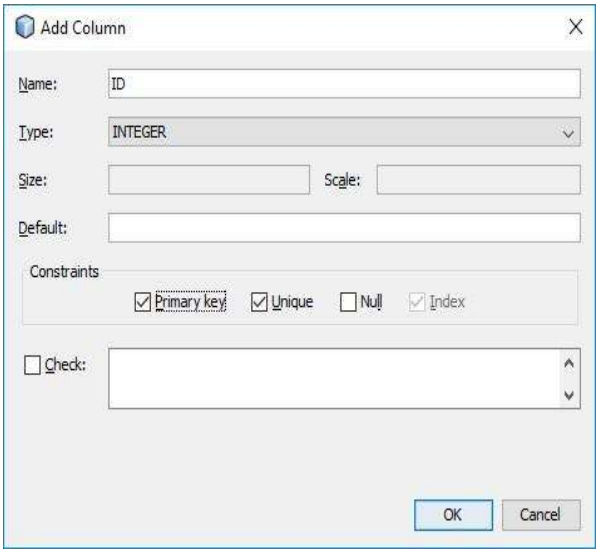
4. Now we are going to create a table in default database sample.
Right click on Table -> Create Table



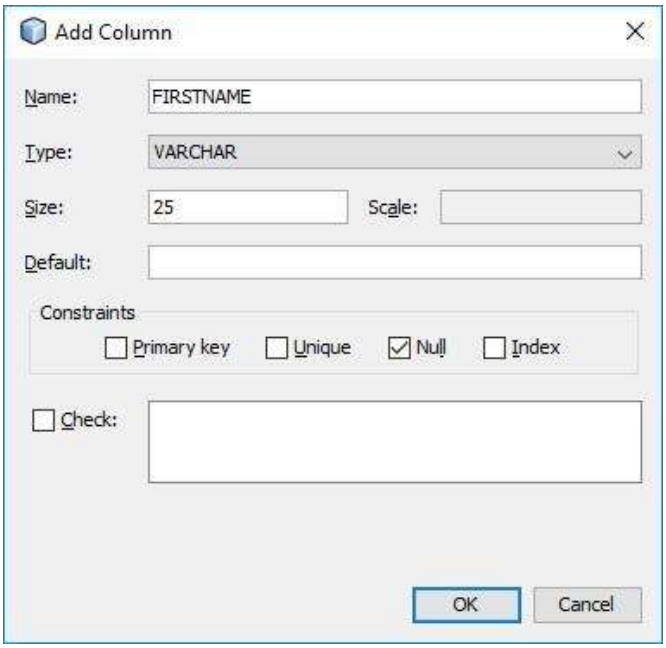
5. Give table name as FRIENDS.



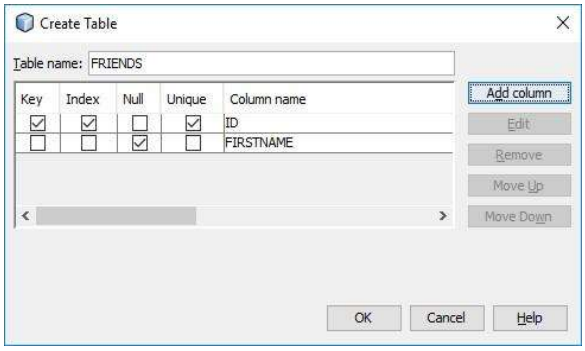
6. click on Add column button to add columns in table.
Enter details as in below pic and select Primary key. After that click on OK button.



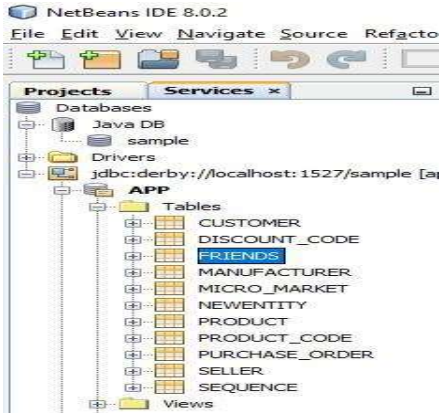
7. Now add second column with following detail. But don't select primary and click on OK button.



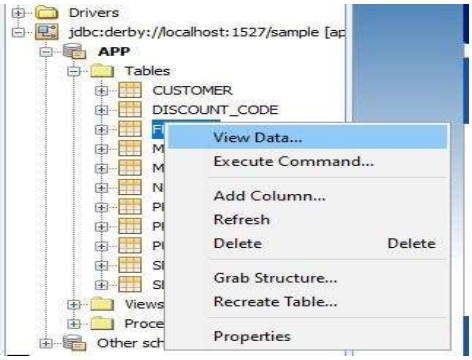
8. click on OK button.



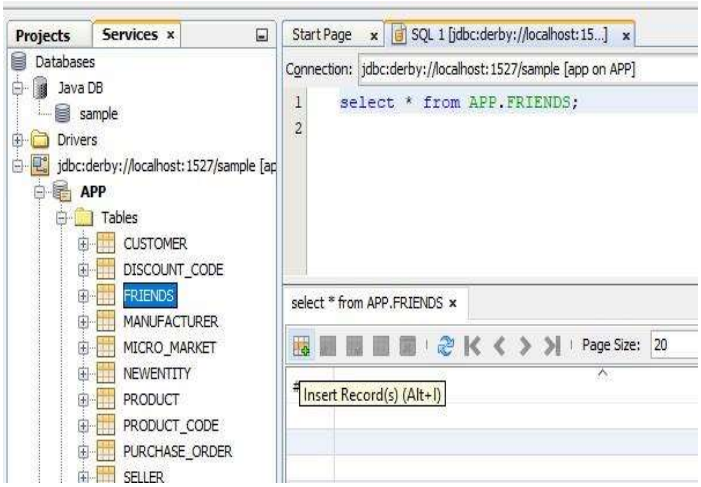
9. Now you can see a table with name FRIENDS in the table.



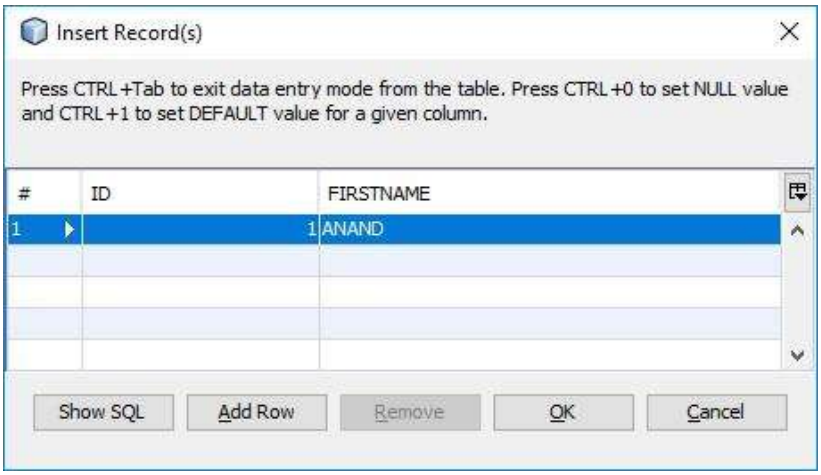
10. Right click on FRIENDS to view and add records into it.



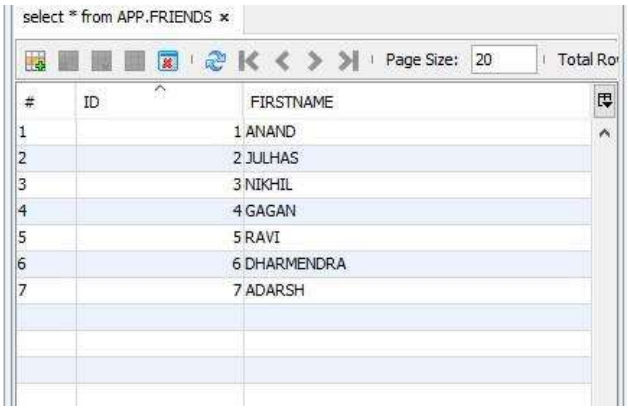
11. click on the leftmost icon in second panel to insert some record.



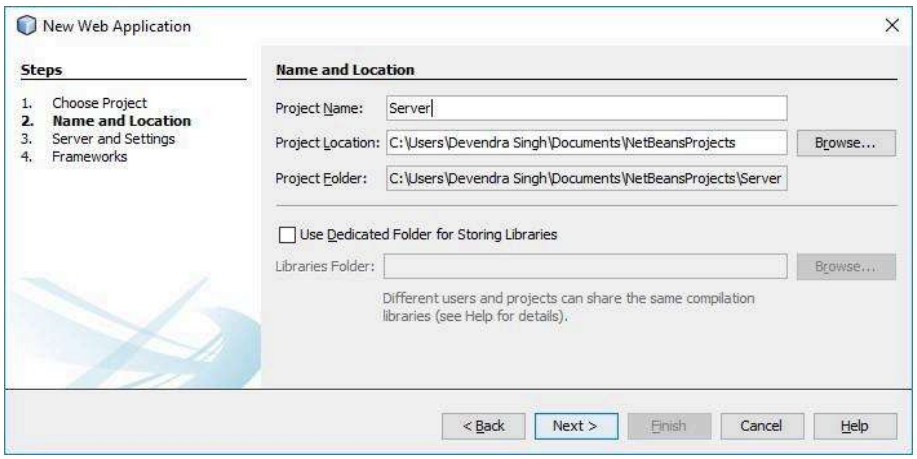
12. Insert a record and then click on Add Row button to insert more record. After that click on OK button to finish.



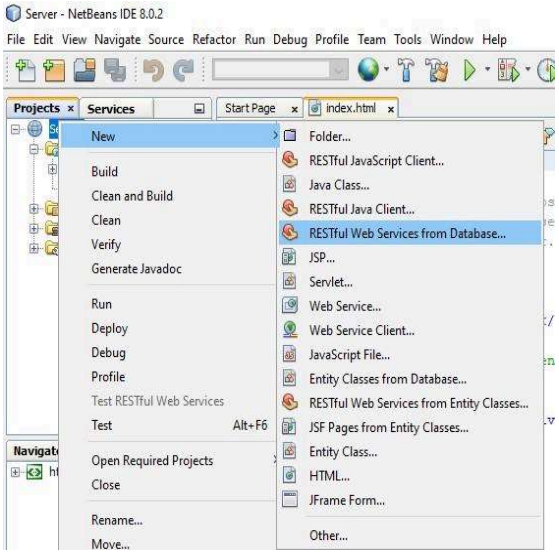
13. As you can see, I have entered 7 records.



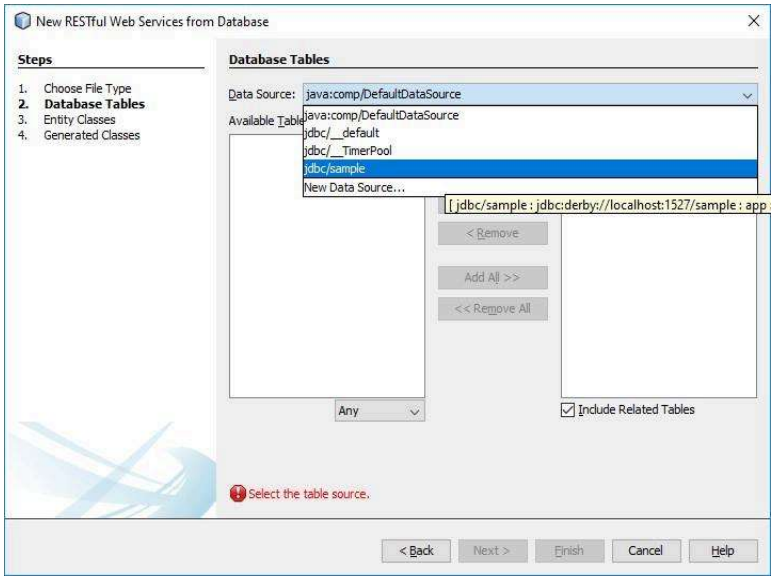
14. create a web application with name Server. After that click on Next and then Finish button.



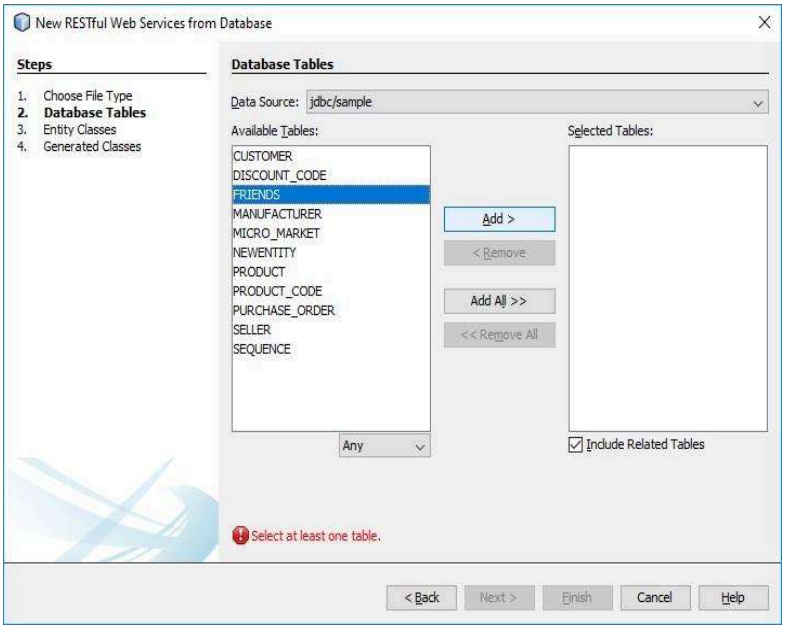
15. Now create a RESTful Web Service from Database by right click on project name.



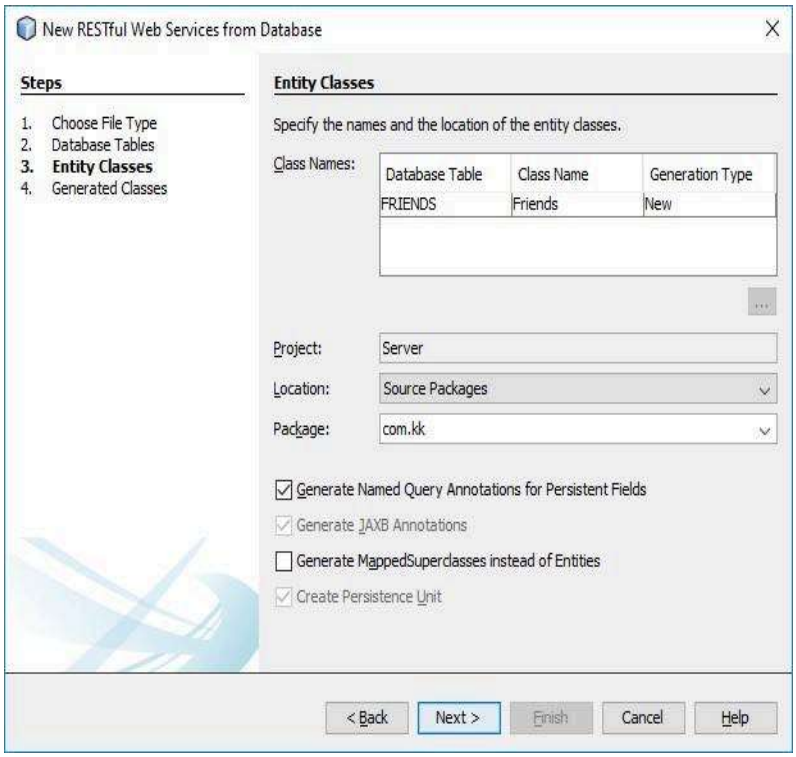
16. Choose Data Source jdbc/sample.



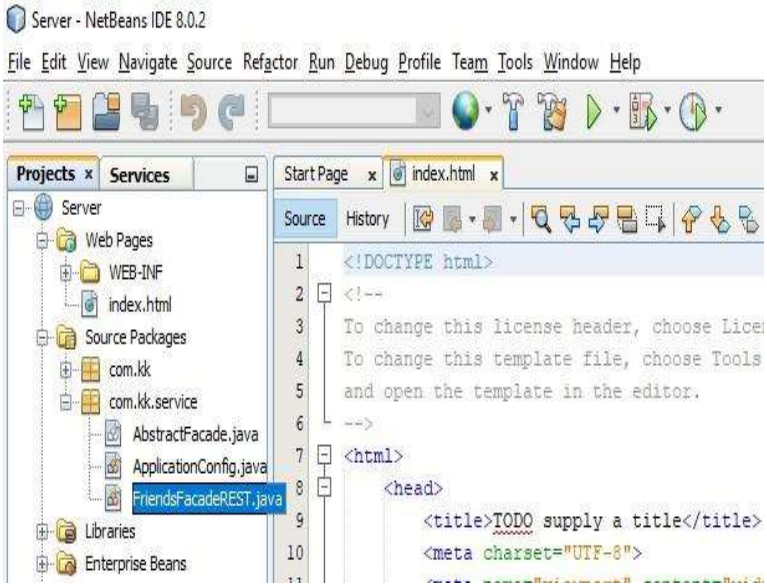
17. Now select FRIENDS and click on Add button. After that click on Next button.



18. Enter Package name as com.kk and click on Next button and then Finish.

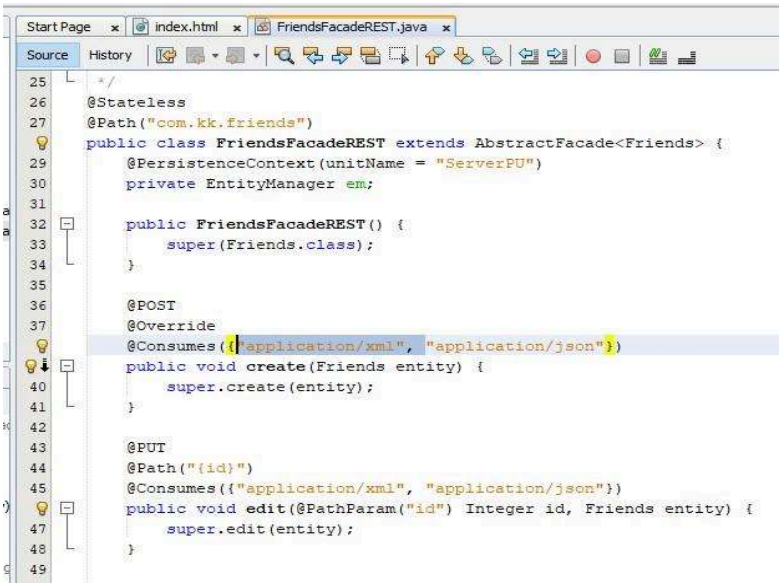


19. Now open selected file by double click on it.

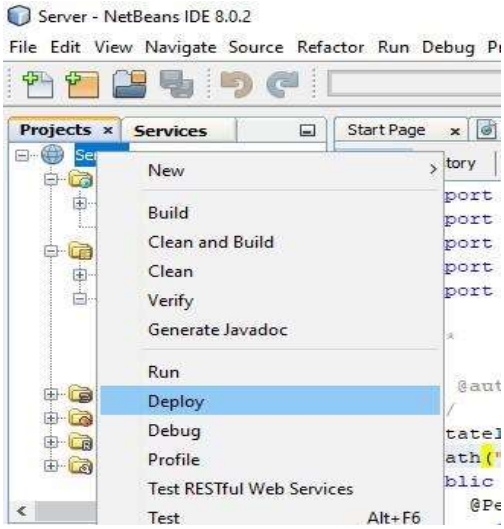


20. Now remove the selected part from every method in this file. So that it will communicate only in JSON format. You can also use methods to convert it. But this is easiest method.

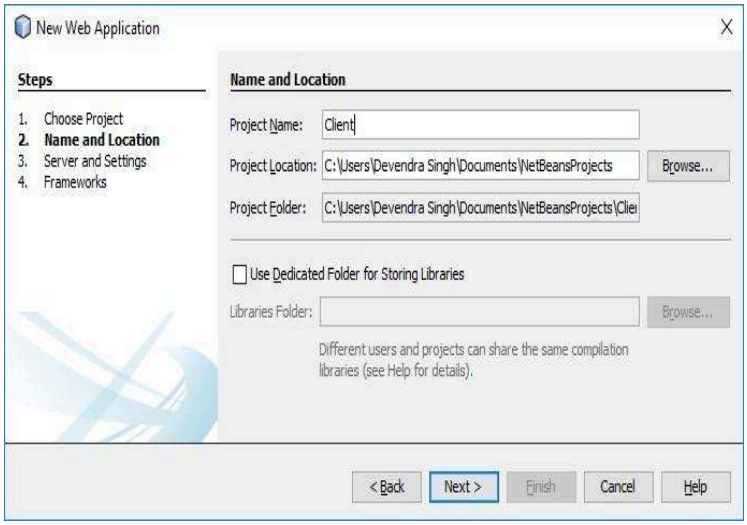
Note : If you get error “javax.ws.rs” doesnot exist, add Java EE 6 API Library in NetBeans IDE by doing myProject->Properties->Libraries->Add Library



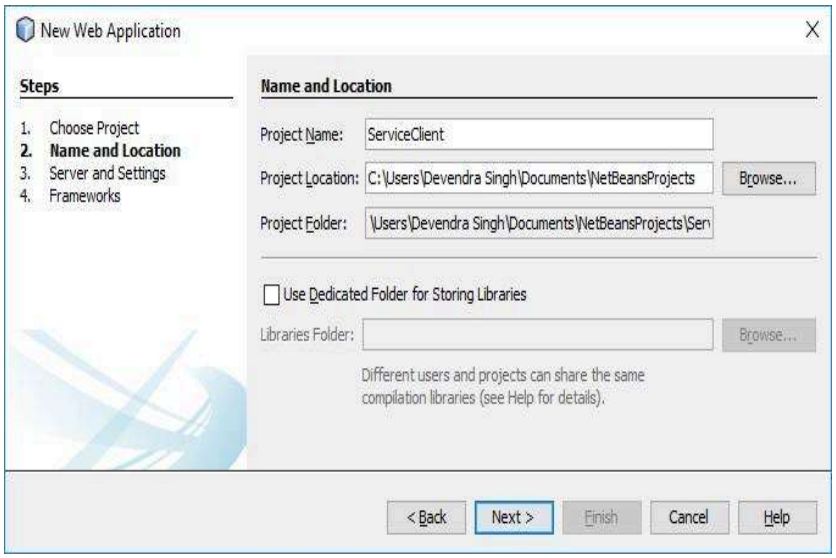
21. After that right click on project name and Deploy it.



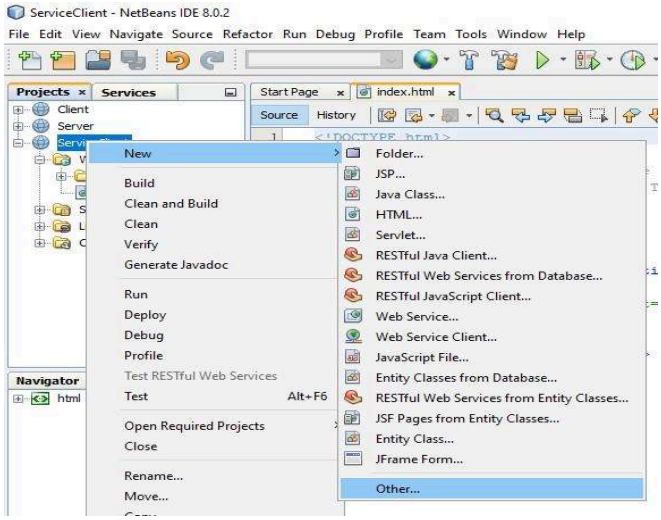
22. Now create one more Web Application as Client. After that click on Next and then Finish button.



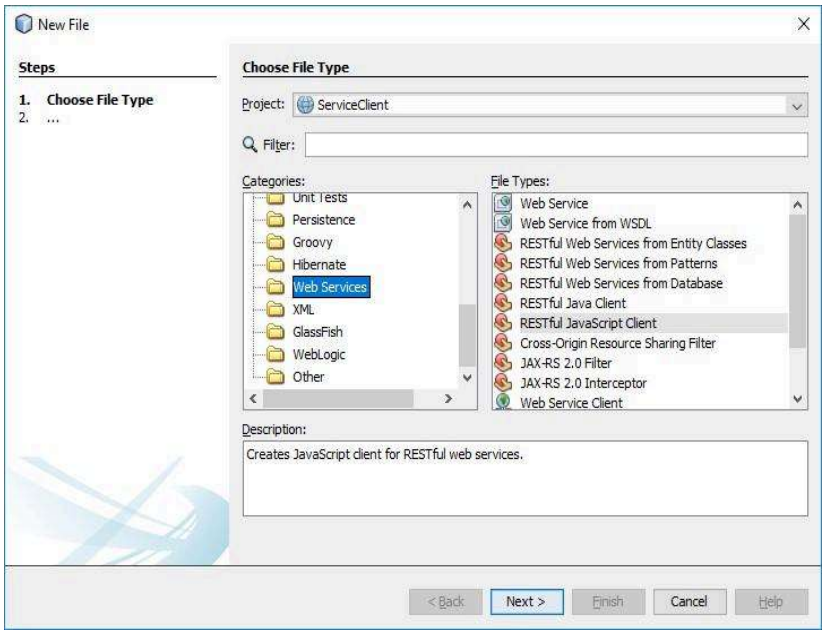
23. Create a Web Application with name ServiceClient.



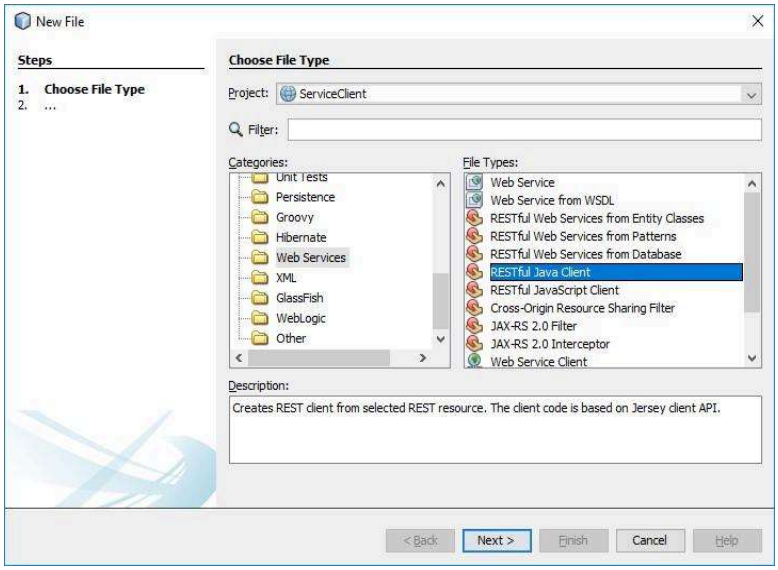
24. Now create a RESTful Java Client.
Right click on ServiceClient -> New -> Other.



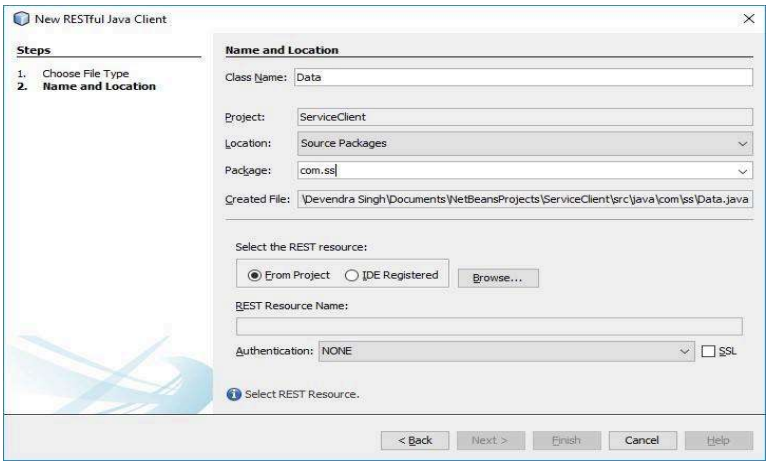
25. Drag down and select Web Services and in side panel select RESTful Java Client.



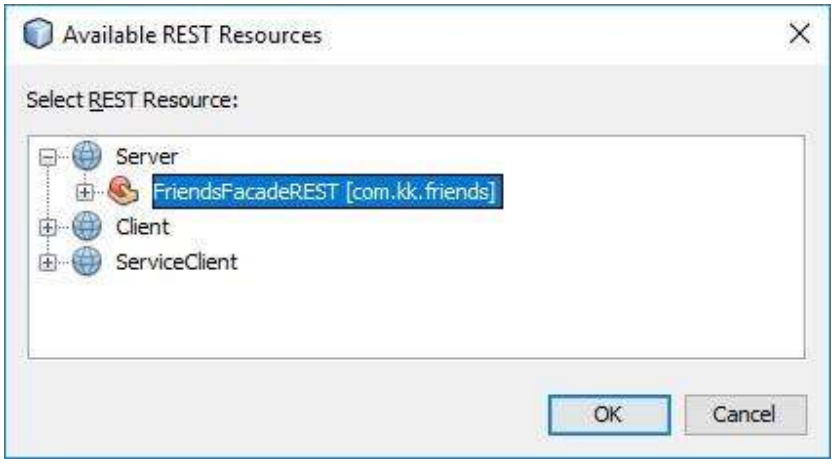
26. After select RESTful Java Client click on Next.



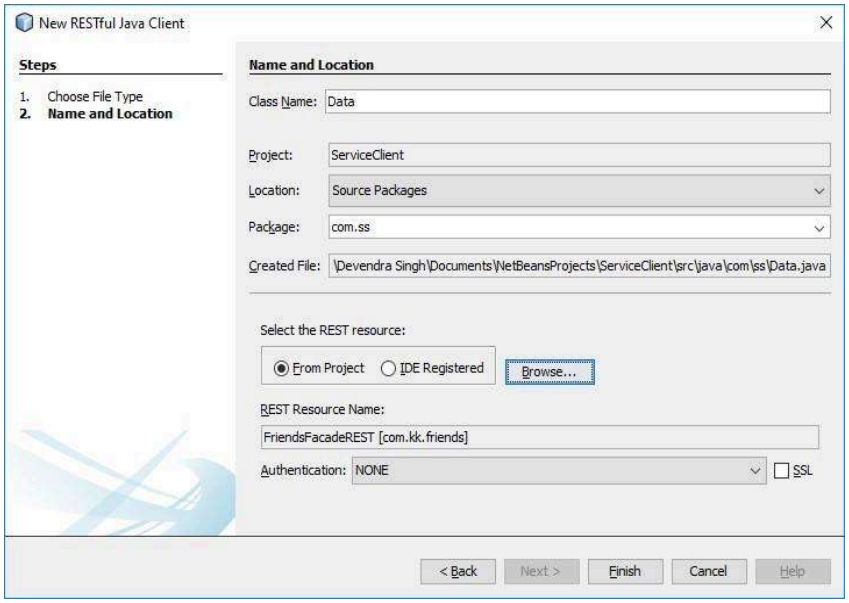
27. Enter following data. Class Name -> Data Package -> com.ss



28. Now click on Browse button and select the option into the below pic. After select click on OK button.

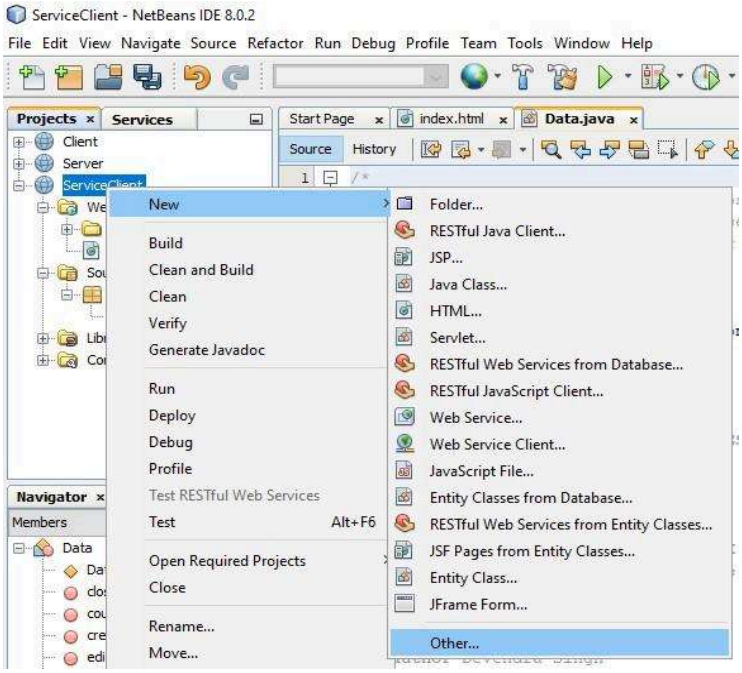


29. Click on Finish.

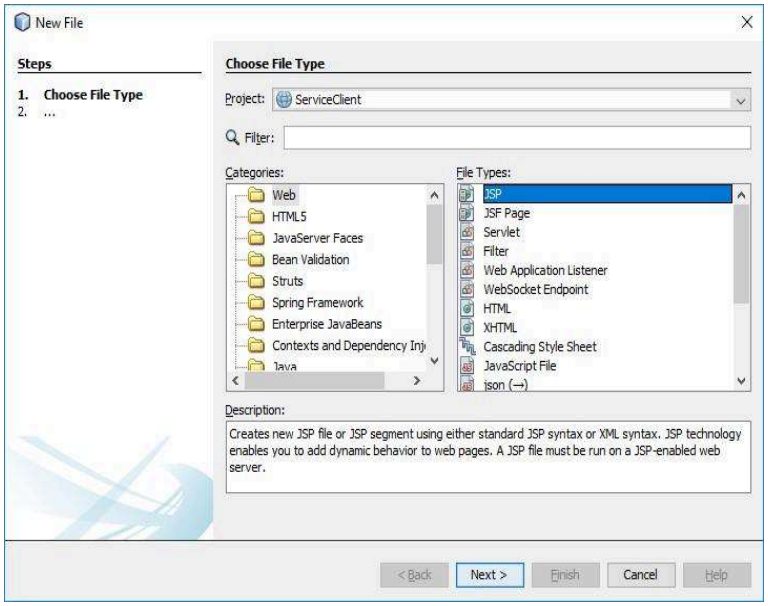


30. Now create a JSP page.

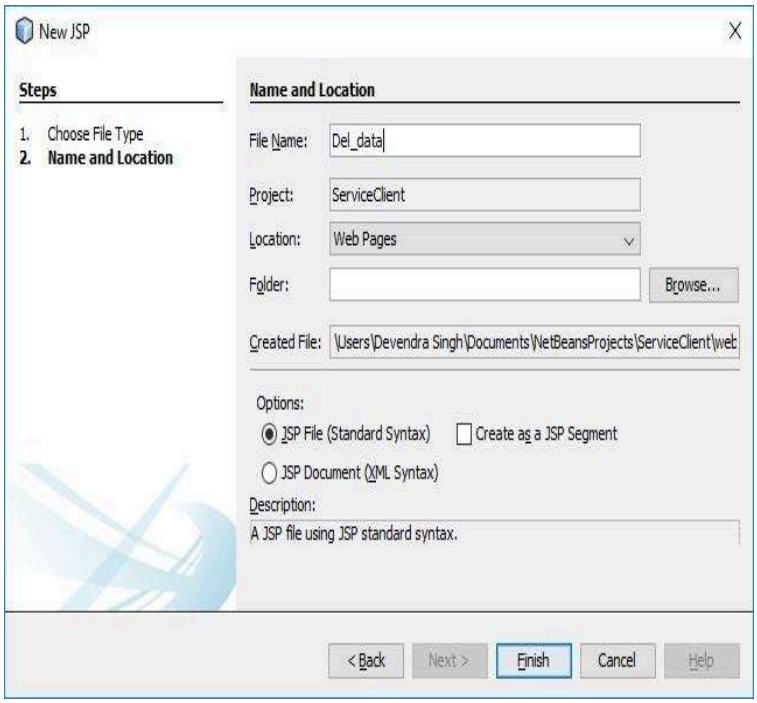
Right click on ServiceClient -> New -> Other



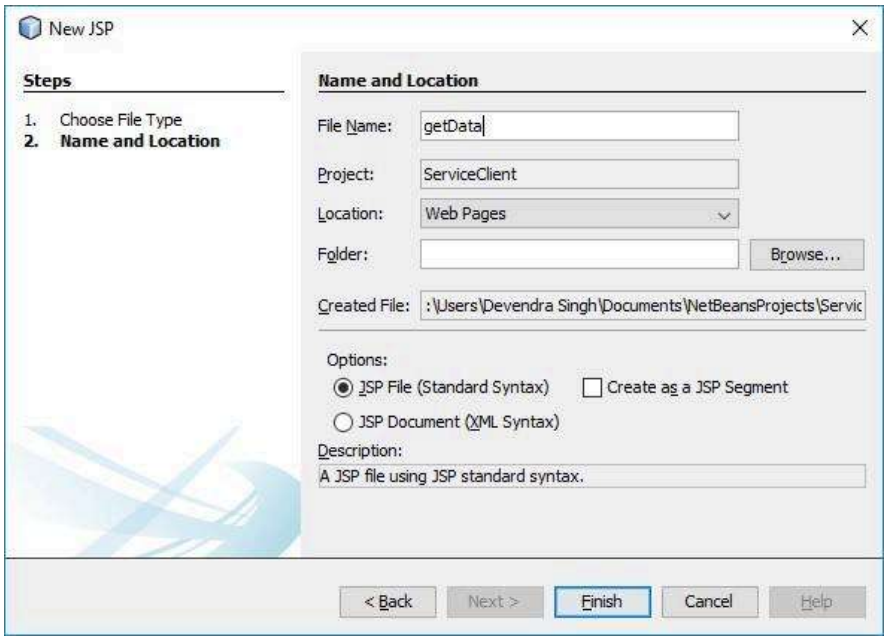
31. Select Web in Categories section -> Select JSP and click on Next button.



32. Enter File Name Del_data and click on Finish button.

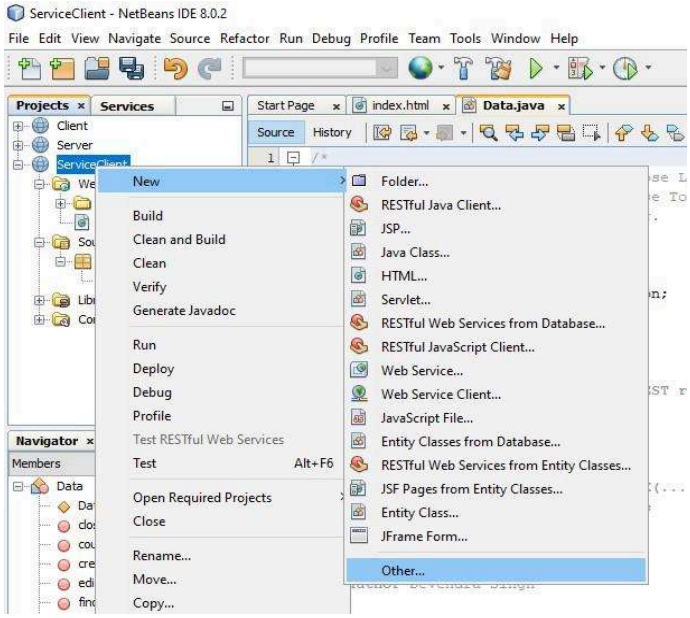


33. Now create one more JSP file by follow the step number 30, 31 & 32. But File Name will be getData.

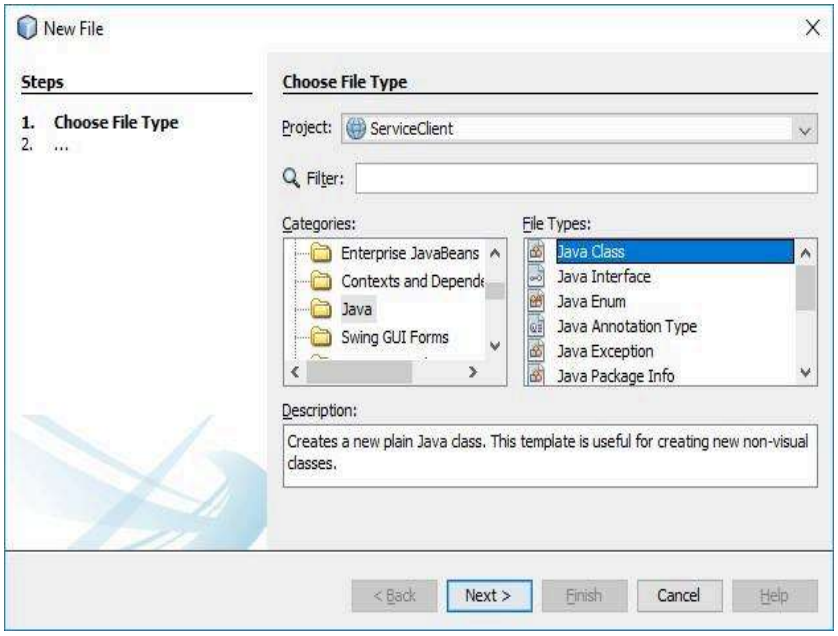


34. Now create a Java class.

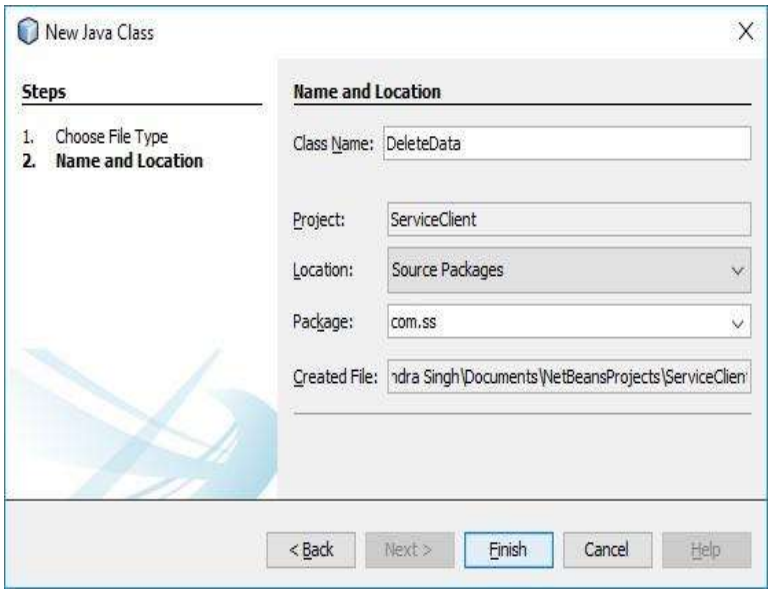
Right click on ServiceClient -> New -> Other



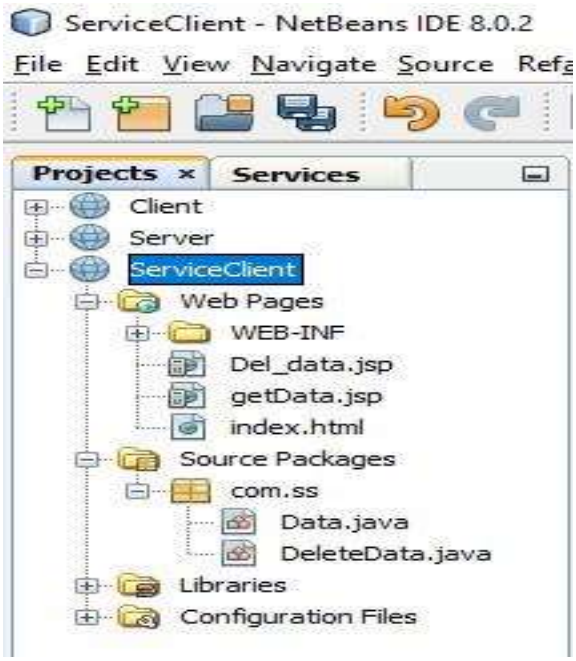
35. In Categories select Java and in File Types select Java Class. Click Next button.



36. Enter Class Name DeleteData and Package com.ss. After that click on Finish.



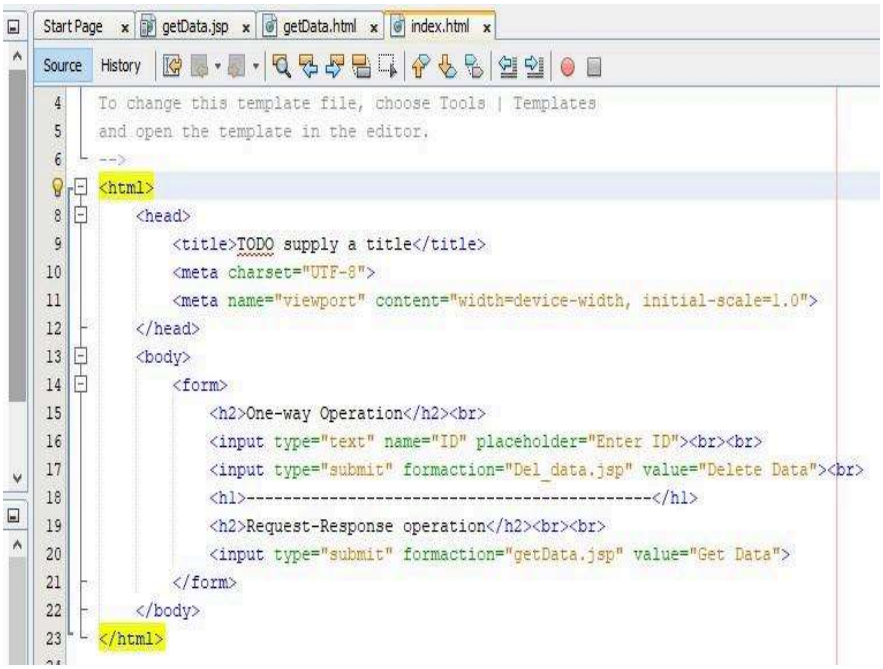
37. Your project file structure will look like below.



38. Now open the index.html of ServiceClient project by double click on it and add the following code in between body tag.

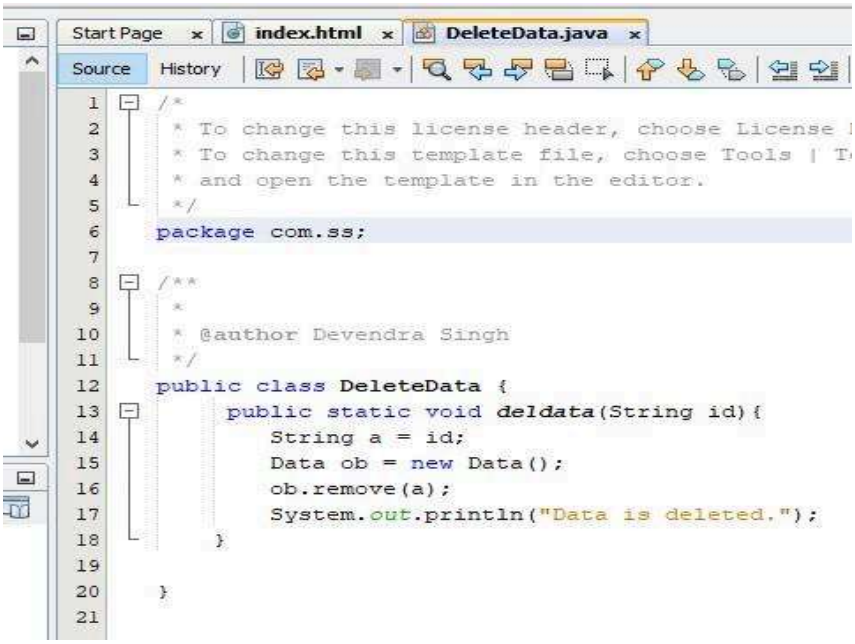
```
<form>
  <h2>One-way Operation</h2><br>
  <input type="text" name="ID" placeholder="Enter ID"><br><br>
  <input type="submit" formaction="Del_data.jsp" value="Delete Data"><br>

  <h1>-------------------------------------------</h1>

  <h2>Request-Response operation</h2><br><br>
```

```
    <input type="submit" formaction="getData.jsp" value="Get Data">
</form>
```



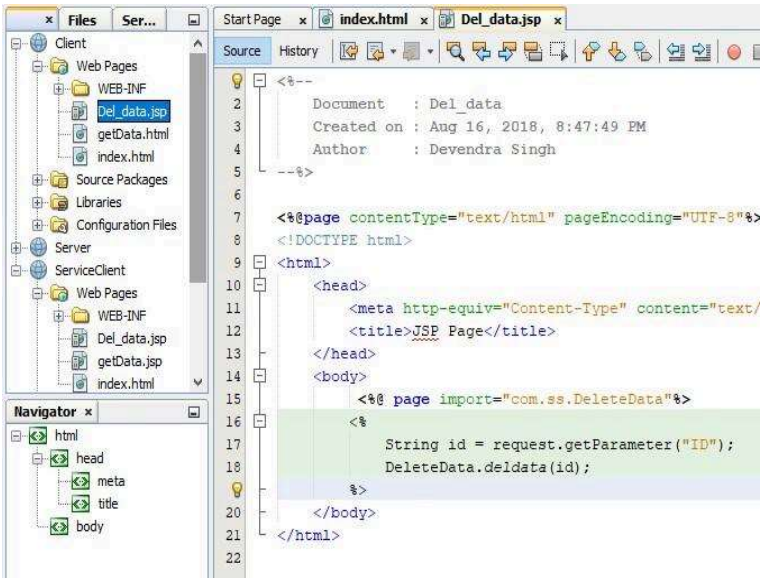
39. Now open DeleteData.java file by double click on it and add the following code in the class and save it by pressing Ctrl+S.

```
public static void deldata(String id){
        String a = id;
        Data ob = new Data();
        ob.remove(a);
        System.out.println("Data is deleted.");
    }
```



40. Now open the Del_data.jsp file and replace the contents of body with the following code.

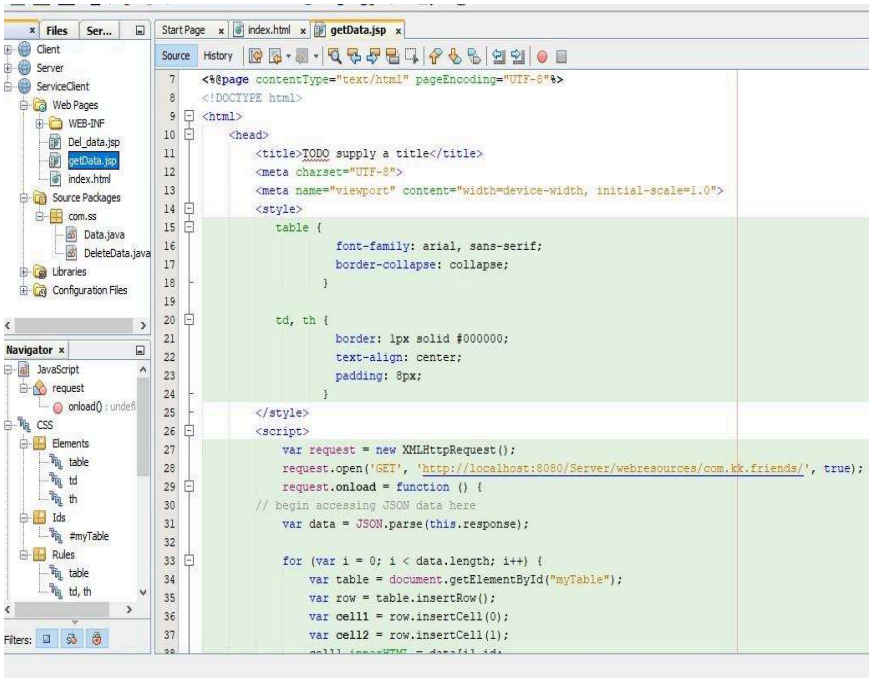
```
<%@ page import="com.ss.DeleteData"%>
    <%
      String id = request.getParameter("ID");
      DeleteData.deldata(id);
    %>
```



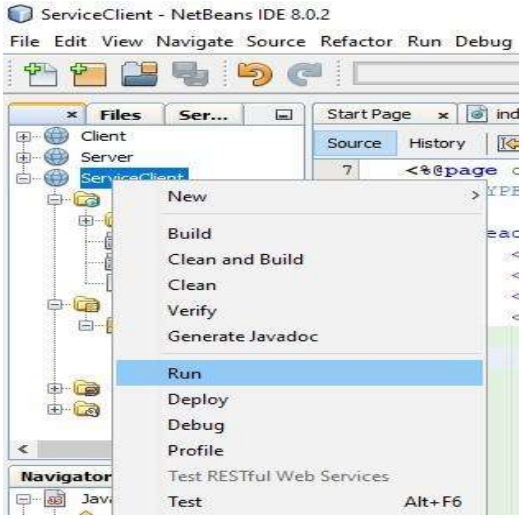
41. Now open the getData.jsp file and replace the contents of html tag with the following code and save it.

```
<head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initialscale=1.0">
```

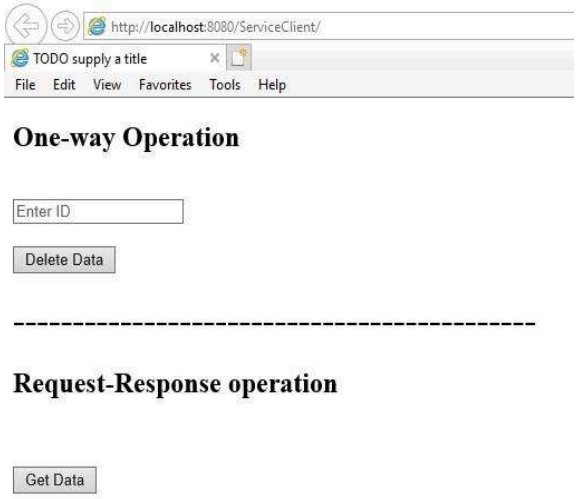
```
    <style>
    table {
                font-family: arial, sans-serif;
               border-collapse: collapse;
           }
    td, th {
             border: 1px solid #000000;
             text-align: center;
             padding: 8px;
           }
    </style>
    <script>
       var request = new XMLHttpRequest();
       request.open('GET','http://localhost:8080/Server/webresources/com.kk.friends/', true);
request.onload = function () {
                    // begin accessing JSON data here
                    var data = JSON.parse(this.response);
                    for (var i = 0; i < data.length; i++) {
                          var table = document.getElementById("myTable");
                            var row = table.insertRow();              var
                            cell1 = row.insertCell(0);              var
                            cell2 = row.insertCell(1);
                            cell1.innerHTML = data[i].id;

                            cell2.innerHTML = data[i].firstname;
       }
    };

    request.send();
    </script>

  </head>
  <body>
    <table id="myTable">
      <tr>
        <th> ID</th>
        <th>NAME</th>
      </tr>
    </table>

  </body>
```



42. Now Run the ServiceClient project.



43. On run the project following window will open in browser.



If you will click on Delete Data button, record of entered ID will be deleted.

If you will click on Get Data button Request for the data will be send and it will return the data as response. Hence it is two way operation.

44. By click on Get Data button.

ID	NAME
1	ANAND
2	JULHAS
3	NIKHIL
4	RAVI
5	GAGAN
6	DHARMENDRA
7	ADARSH
8	DEVENDRA

45. Entering ID 8 and clicking on Delete Data button.

46. Now go back and again click on Get Data button. Refresh the page, You will see data with id number 8 is deleted.

ID	NAME
1	ANAND
2	JULHAS
3	NIKHIL
4	RAVI
5	GAGAN
6	DHARMENDRA
7	ADARSH

Practical - 4

Aim: Develop application to consume Google's search / Google's Map RESTful Web service.

STEPS:

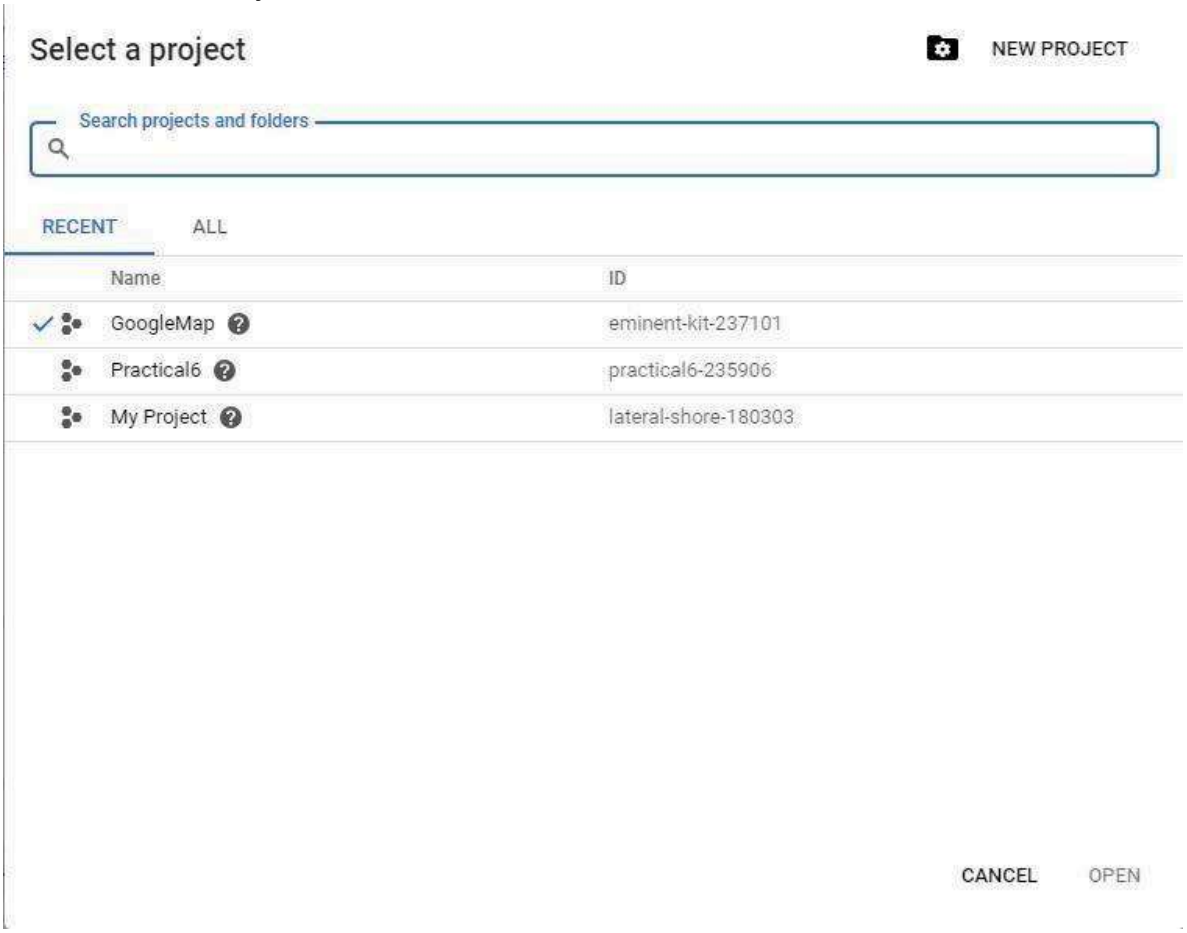
1. First of all we need to create a Java Web Application with any name, let it be GoogleMap here using Netbeans IDE.
2. The code inside the input.jsp will be similar to this input.jsp Input.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
                <title>JSP Page</title>
</head>
<body>
        <form action="index.jsp">
        <pre>
        Enter latitude:<input type="text" name="t1" />
        Enter longitude:<input type="text" name="t2" />
        <input type="submit" value="Show" />
        </pre>
        </form>
</body>
</html>
```

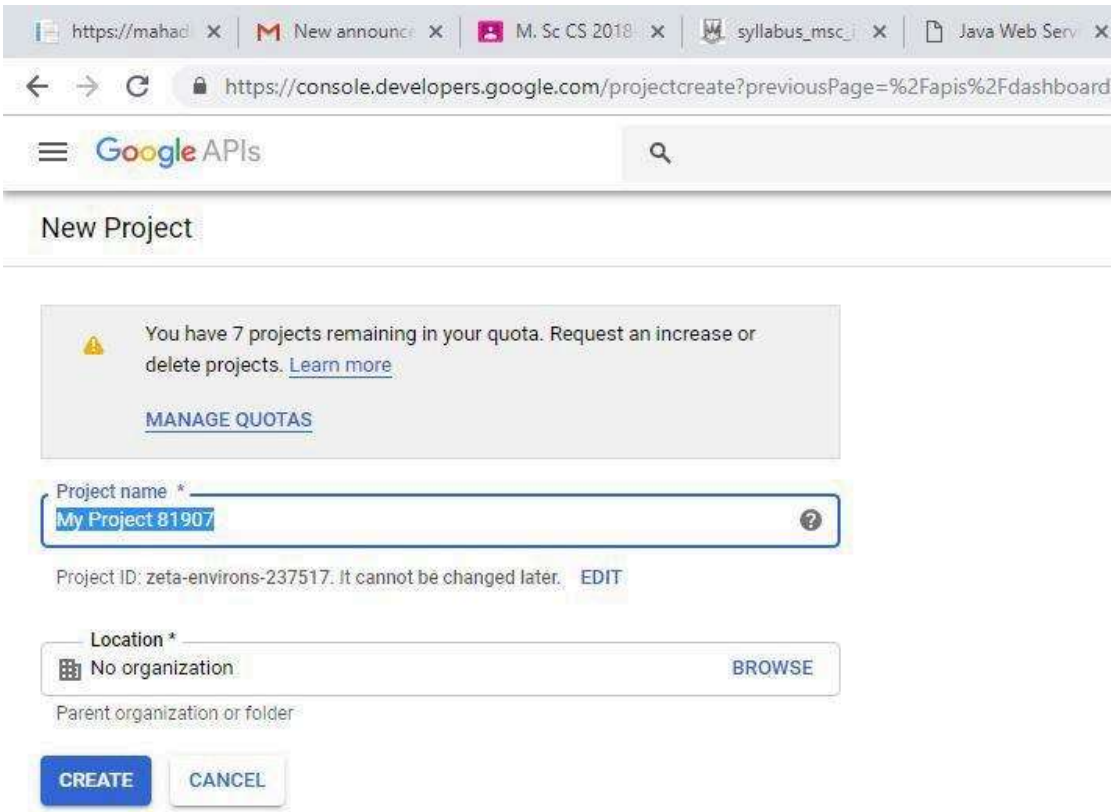
3. Before running the application we need the Google API key. The steps are shown here:

- Visit Google APIs Console (https://console.developers.google.com, you have to login with your Google account)

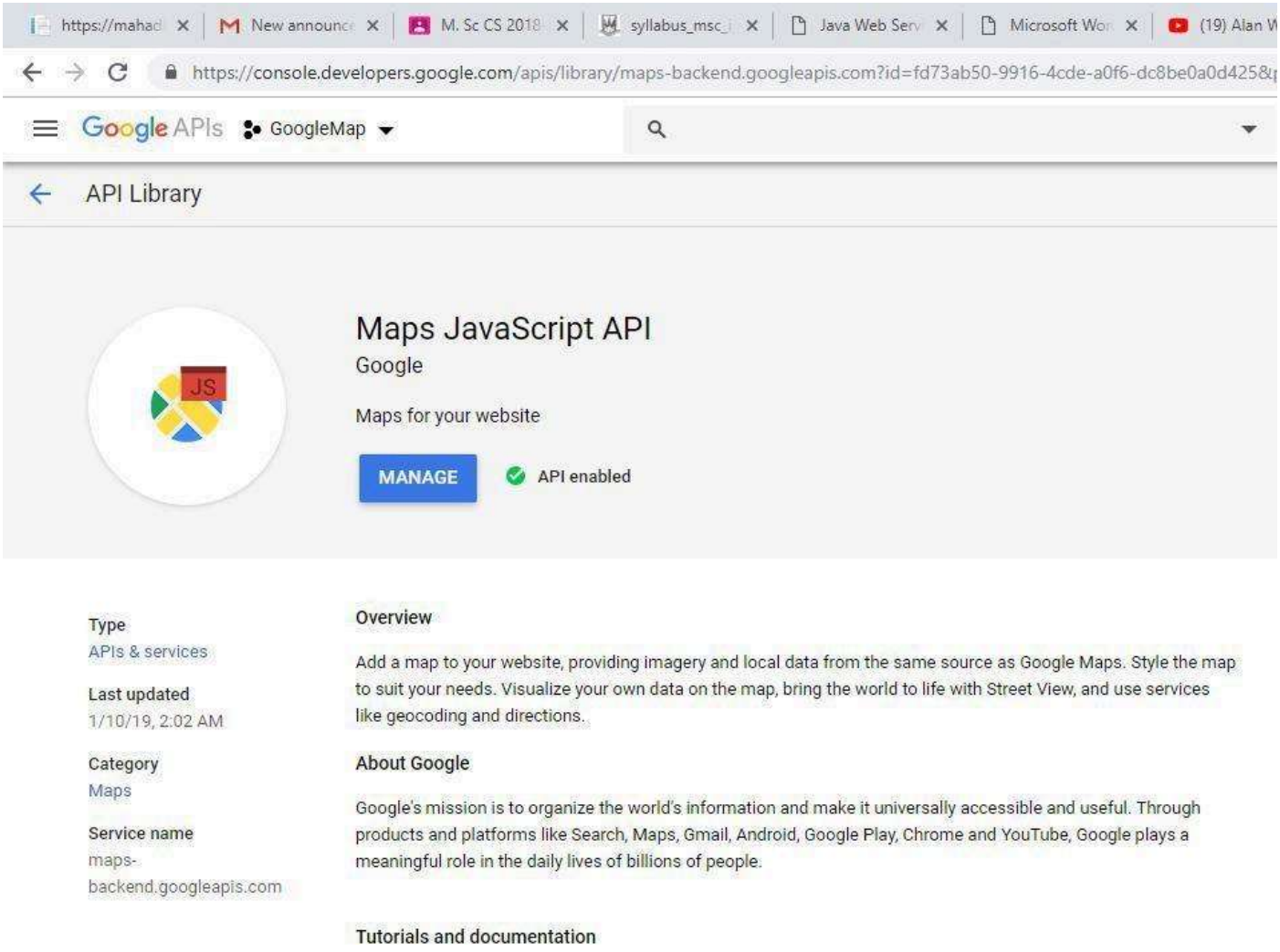
Create a new API Project.



- Enter the name to your project.



- Enable the Google Maps API V3.



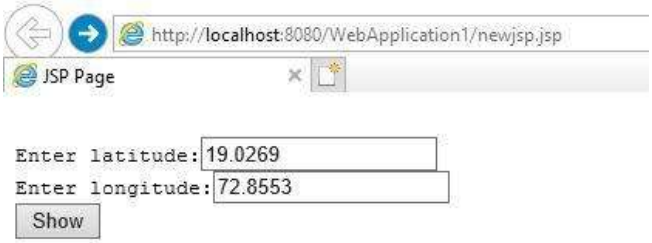
4. Create another file index.jsp

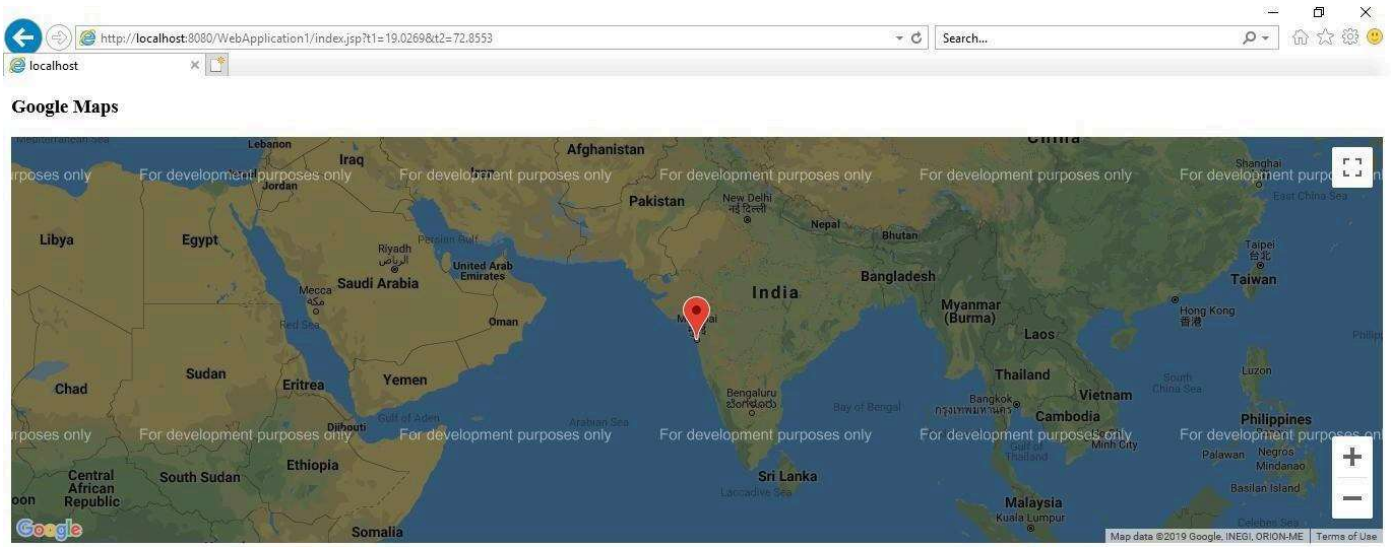
Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<style>
#map {
height: 400px;
width: 100%;
}
</style>
</head>
<body>
<%
```

```
double lati=Double.parseDouble(request.getParameter("t1")); double
longi=Double.parseDouble(request.getParameter("t2"));
%>

<h3> Google Maps </h3>

<div id="map"></div>

<script lang="javascript">
function initMap() {
var info={lat: <%=lati%>, lng: <%=longi%>};

var map = new google.maps.Map(document.getElementById('map'),

{

zoom: 4, center: info

});
var marker = new google.maps.Marker({
position: info,
map: map

});

}

</script>

<script async defer src="put
your key here">
</script>

</body>

</html>
```

Output :-




http://localhost:8080/WebApplication1/index.jsp?t1=19.0269&t2=72.8553
Search...
localhost
Google Maps
For development purposes only
Lebanon
Iraq
Jordan
Afghanistan
Pakistan
New Delhi
Nepal
Bhutan
Bangladesh
India
Myanmar (Burma)
Laos
Thailand
Vietnam
Cambodia
Libya
Egypt
Riyadh
Saudi Arabia
United Arab Emirates
Mecca
Oman
Red Sea
Sudan
Chad
Eritrea
Yemen
Gulf of Aden
Djibouti
Ethiopia
Somalia
Central African Republic
South Sudan
Bengaluru
Bay of Bengal
Arabian Sea
Sri Lanka
Laccadive Sea
Malaysia
Kuala Lumpur
Bangkok
Shanghai
East China Sea
Taipei
Taiwan
Hong Kong
South China Sea
Luzon
Philippines
Palawan
Negros
Mindanao
Basilan Island
Google
Map data ©2019 Google, INEGI, ORION-ME
Terms of Use

Practical 5

Aim: Installation and Configuration of virtualization using KVM.

Theory :

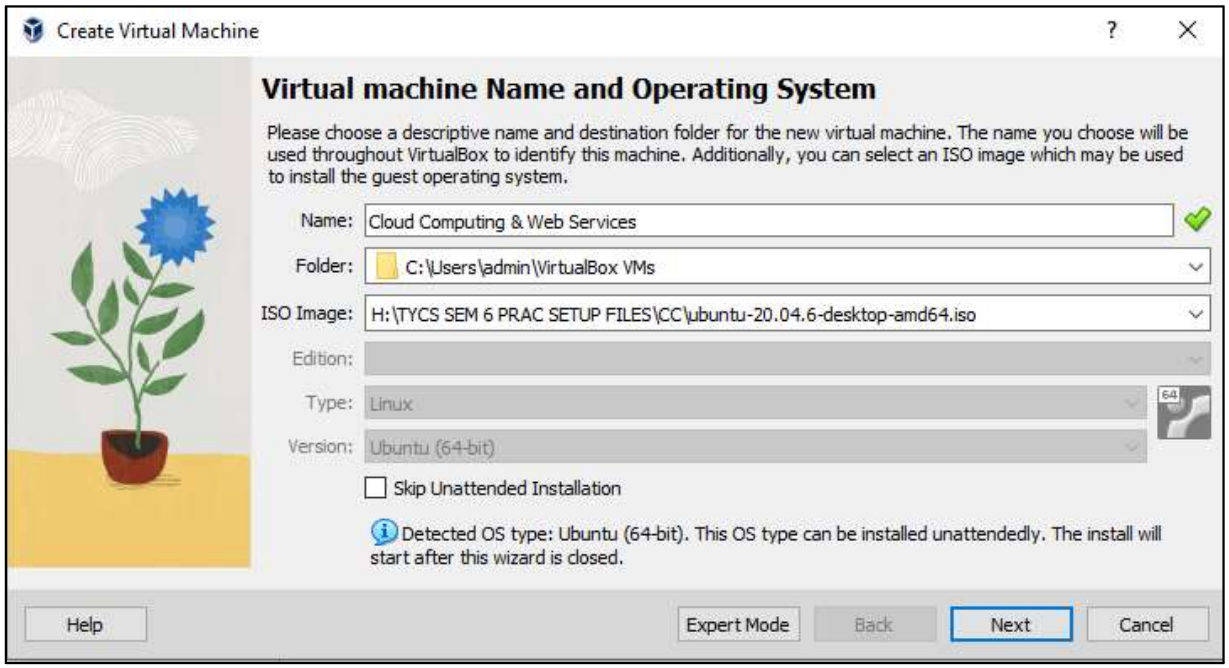
• KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm that provides the core virtualization infrastructure and a processor specific module, kvm-intel or kvm-amd.

• Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.

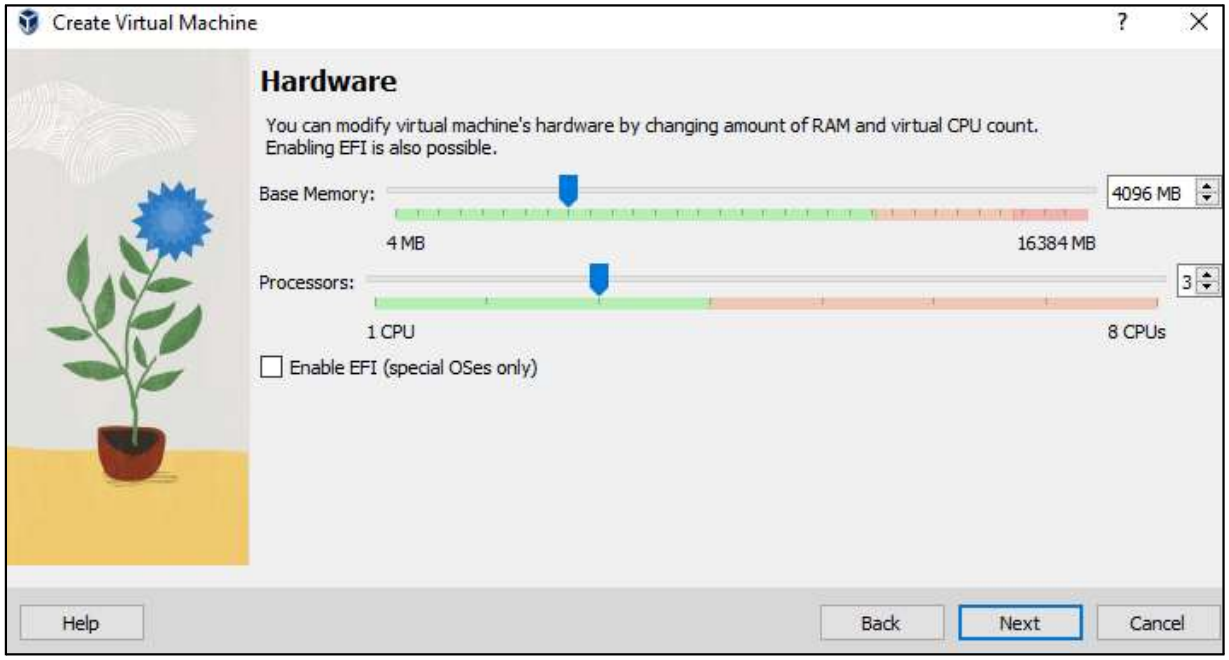
• KVM is open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The user space component of KVM is included in mainline QEMU, as of 1.3.

Requirements: Virtual Box, Ubuntu ISO file (latest version)

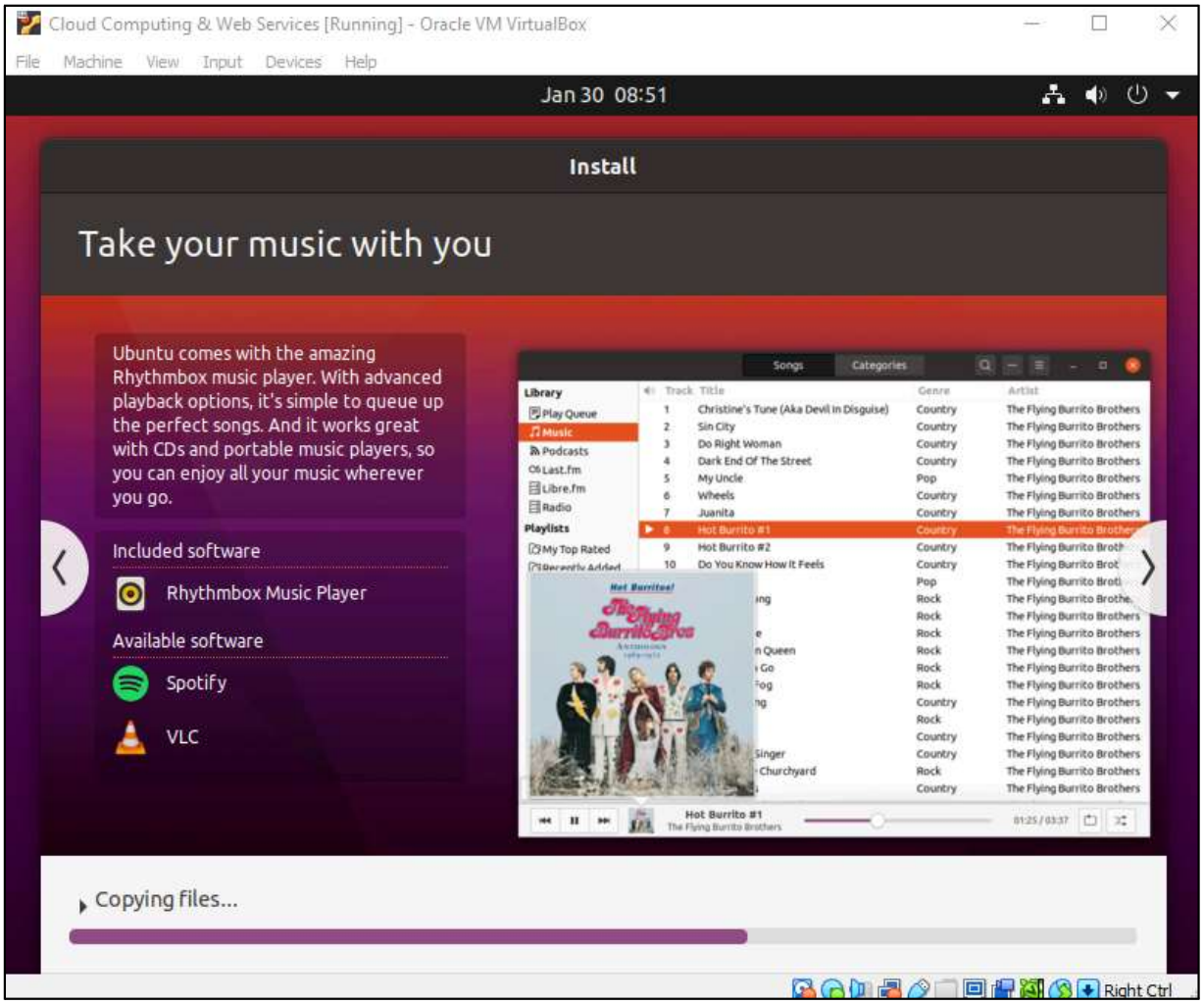
Steps:

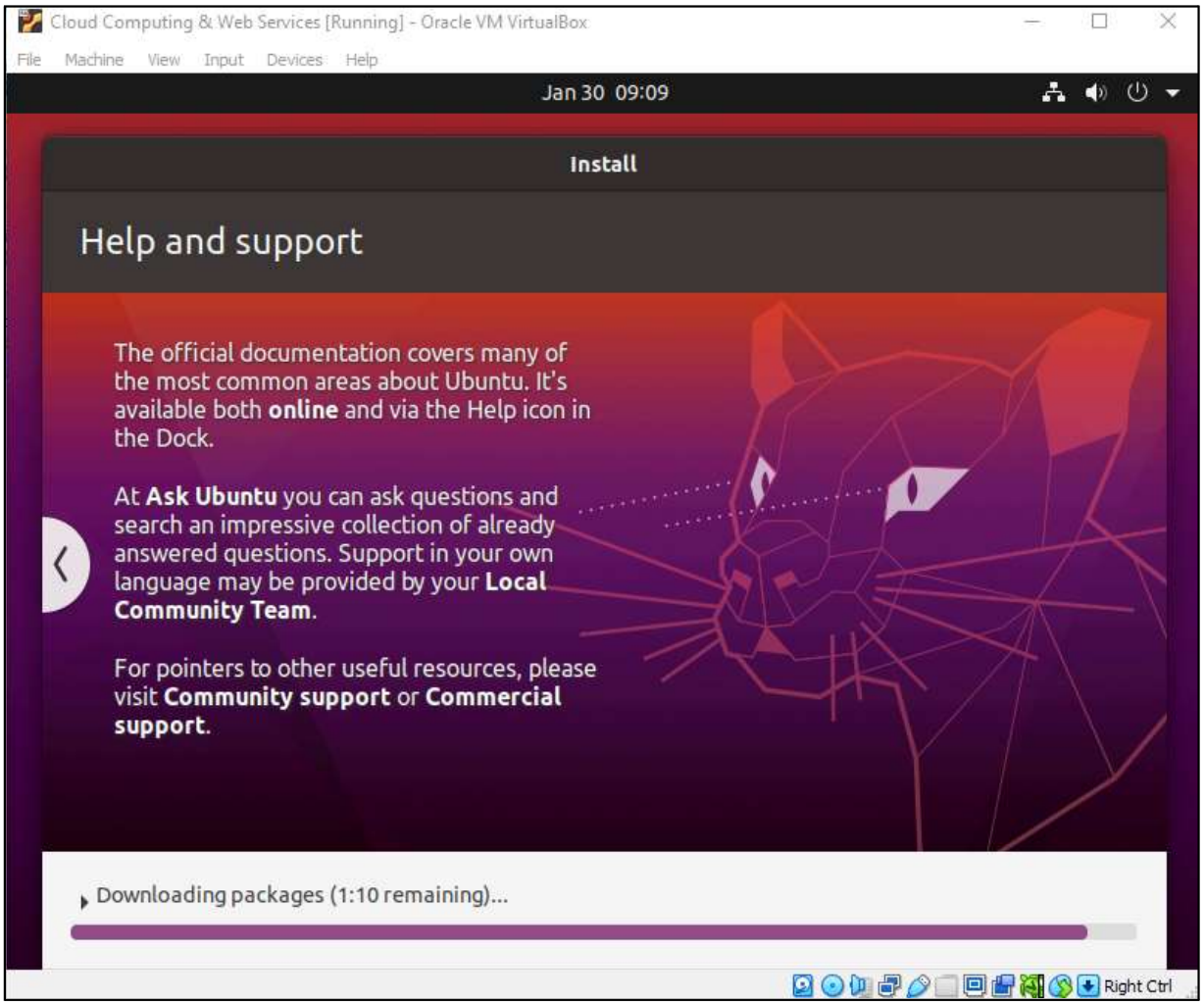


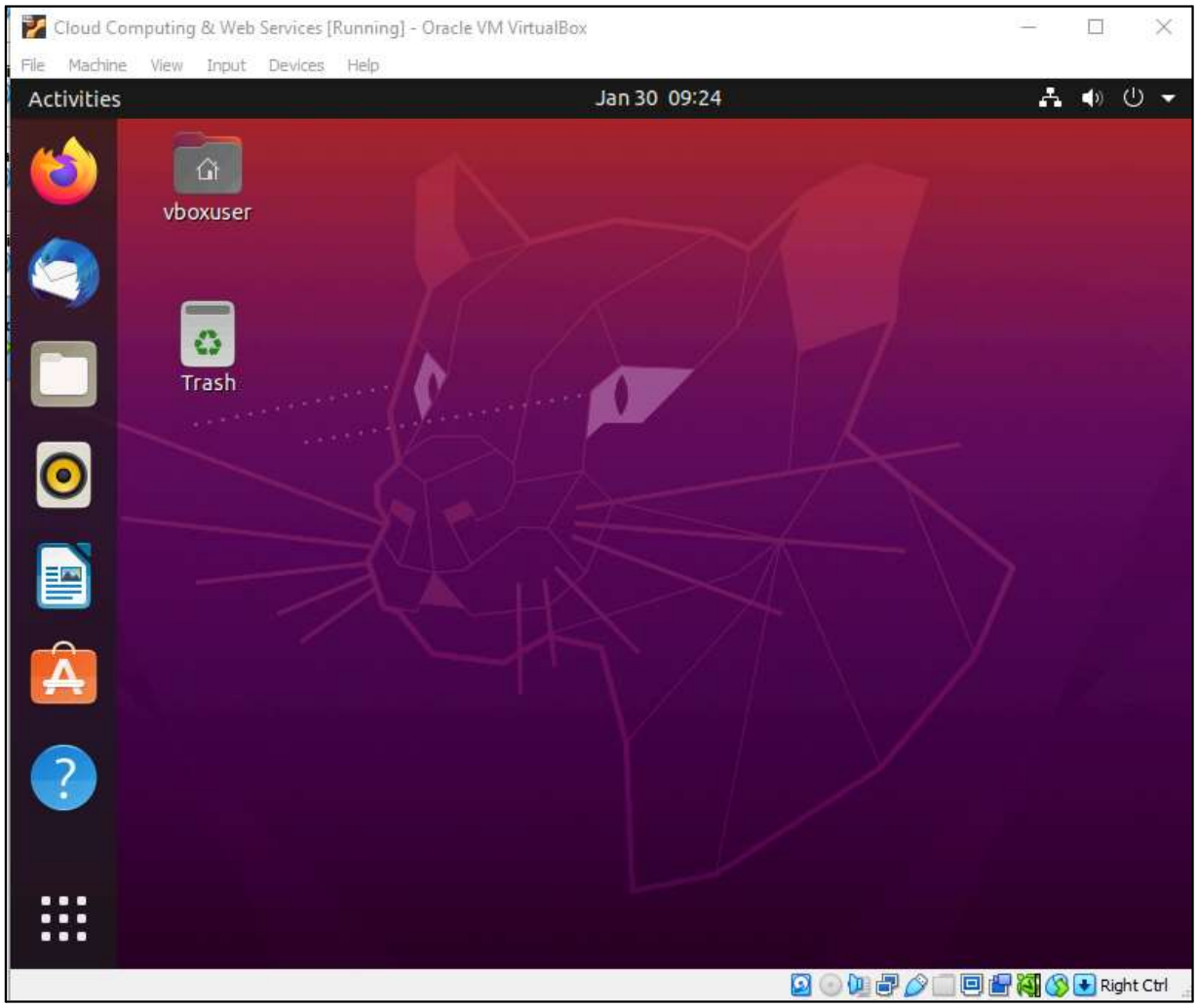

Create Virtual Machine
Unattended Guest OS Install Setup
You can configure the unattended guest OS install by modifying username, password, and hostname. Additionally you can enable guest additions install. For Microsoft Windows guests it is possible to provide a product key.
Username and Password
Username: vboxuser
Password: changeme
Repeat Password: changeme
Additional Options
Product Key: #####-#####-#####-#####
Hostname: CloudComputingWebService
Domain Name: myguest.virtualbox.org
Install in Background
Guest Additions
Guest Additions ISO: C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso
Help
Back
Next
Cancel

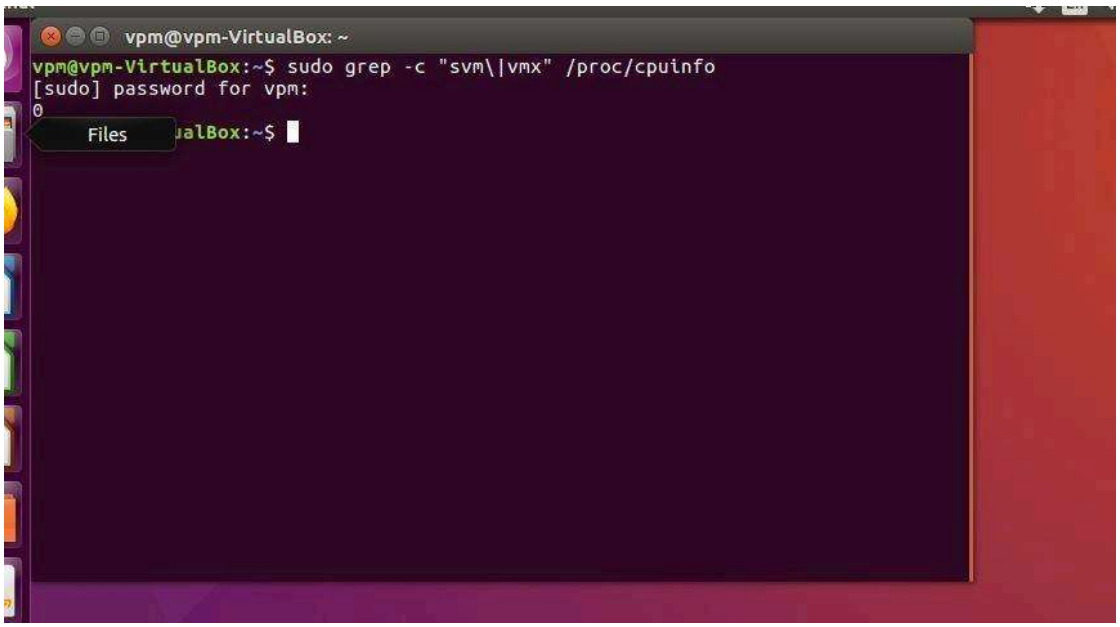

Create Virtual Machine
Hardware
You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.
Base Memory:
4096 MB
4 MB
16384 MB
Processors:
3
1 CPU
8 CPUs
Enable EFI (special OSes only)
Help
Back
Next
Cancel


Create Virtual Machine
Virtual Hard disk
If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select an existing one. Alternatively you can create a virtual machine without a virtual hard disk.
Create a Virtual Hard Disk Now
Disk Size:
25.00 GB
4.00 MB
2.00 TB
Pre-allocate Full Size
Use an Existing Virtual Hard Disk File
Windows 7.vdi (Normal, 32.00 GB)
Do Not Add a Virtual Hard Disk
Help
Back
Next
Cancel


Cloud Computing & Web Services [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jan 30 08:51
Install
Take your music with you
Ubuntu comes with the amazing Rhythmbox music player. With advanced playback options, it's simple to queue up the perfect songs. And it works great with CDs and portable music players, so you can enjoy all your music wherever you go.
Included software
Rhythmbox Music Player
Available software
Spotify
VLC
Copying files...
Right Ctrl


Cloud Computing & Web Services [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jan 30 09:09
Install
Help and support
The official documentation covers many of the most common areas about Ubuntu. It's available both **online** and via the Help icon in the Dock.
At **Ask Ubuntu** you can ask questions and search an impressive collection of already answered questions. Support in your own language may be provided by your **Local Community Team**.
For pointers to other useful resources, please visit **Community support** or **Commercial support**.
Downloading packages (1:10 remaining)...
Right Ctrl

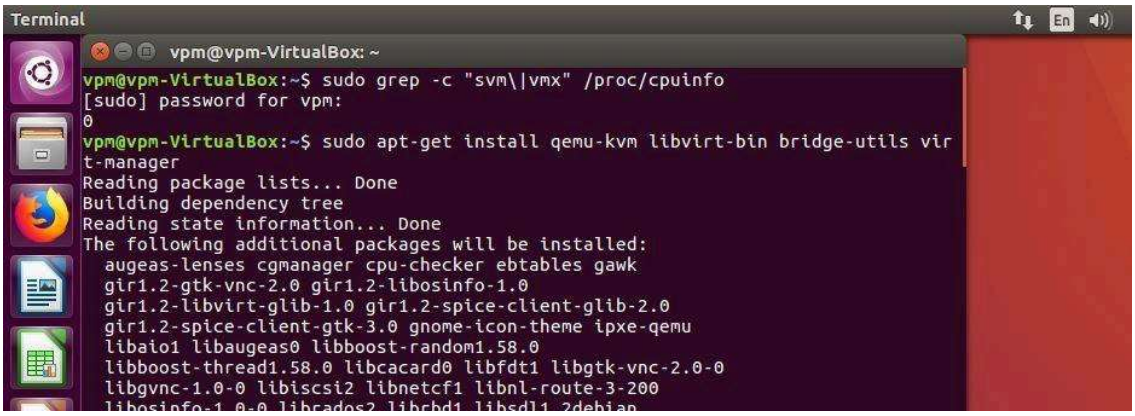

Cloud Computing & Web Services [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Jan 30 09:24
vboxuser
Trash
Right Ctrl


vpm@vpm-VirtualBox: ~
vpm@vpm-VirtualBox:~$ sudo grep -c "svm\|vmx" /proc/cpuinfo
[sudo] password for vpm:
0
Files
ualBox:~$

sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager

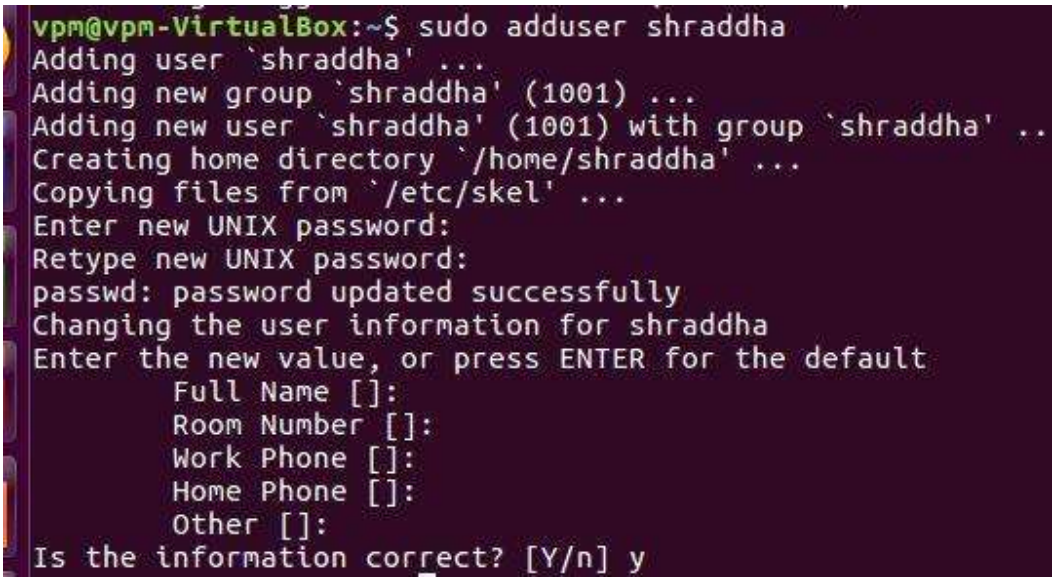
OR

sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils



If the above commands give error then run
sudo apt-get update

Sudo adduser <yourname>



Sudo adduser <yourname> libvirtd

```
vpm@vpm-VirtualBox:~$ sudo adduser shraddha libvirtd
Adding user `shraddha' to group `libvirtd' ...
Adding user shraddha to group libvirtd
Done.
vpm@vpm-VirtualBox:~$
```

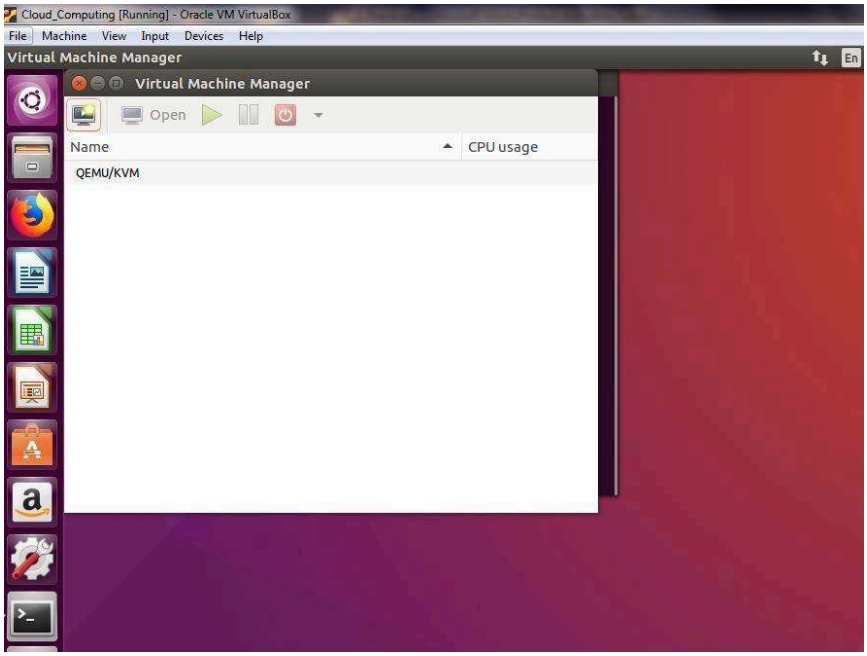
Sudo su <username>

virsh -c qemu:///system list

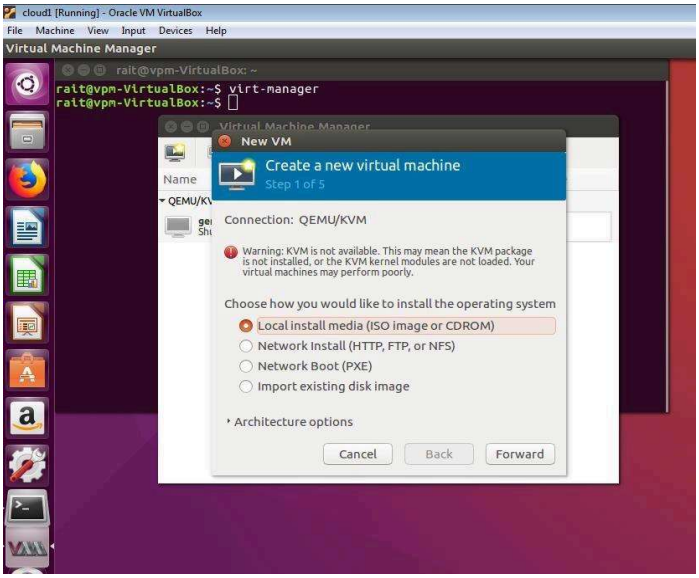
```
shraddha@vpm-VirtualBox:~$ virsh -c qemu:///system list
 Id    Name                           State
----------------------------------------------------

shraddha@vpm-VirtualBox:~$
```

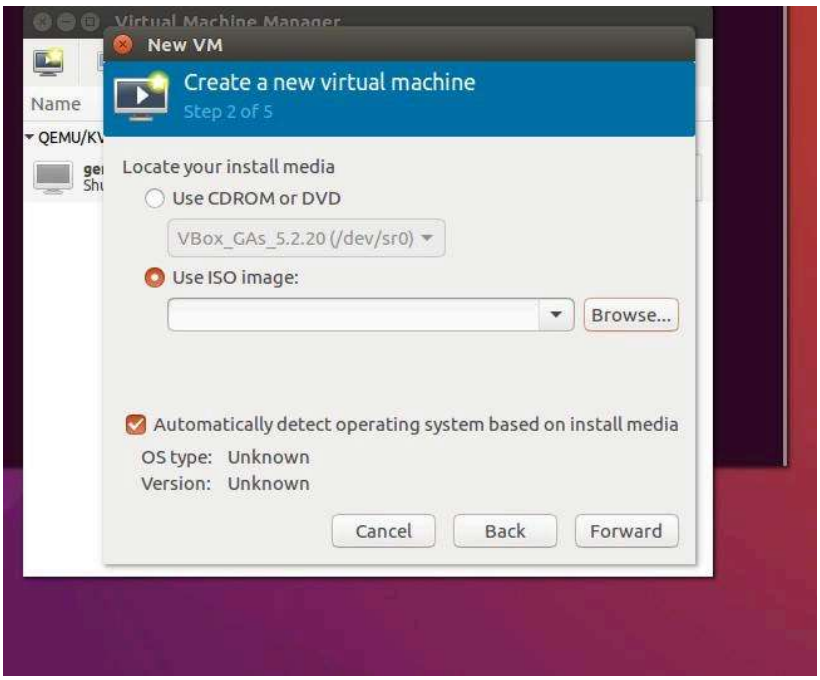
Virt-manager

```
shraddha@vpm-VirtualBox:~$ virt-manager
shraddha@vpm-VirtualBox:~$
```

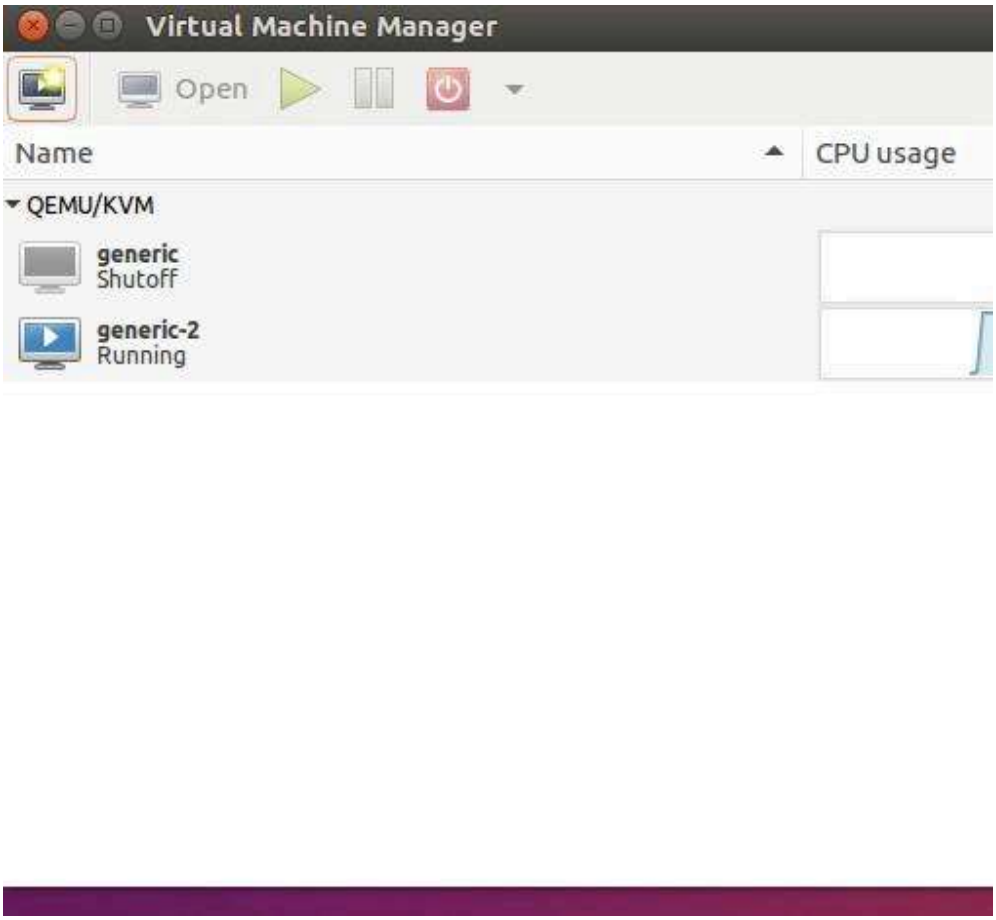


Click on new virtual machine in qemu and perform the following step


cloud1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Virtual Machine Manager
rait@vpm-VirtualBox: ~
rait@vpm-VirtualBox:~$ virt-manager
rait@vpm-VirtualBox:~$
Virtual Machine Manager
New VM
Create a new virtual machine
Step 1 of 5
Name
QEMU/KV
Connection: QEMU/KVM
Warning: KVM is not available. This may mean the KVM package is not installed, or the KVM kernel modules are not loaded. Your virtual machines may perform poorly.
Choose how you would like to install the operating system
Local install media (ISO image or CDROM)
Network Install (HTTP, FTP, or NFS)
Network Boot (PXE)
Import existing disk image
Architecture options
Cancel
Back
Forward


Virtual Machine Manager
New VM
Create a new virtual machine
Step 2 of 5
Name
QEMU/KV
Locate your install media
Use CDROM or DVD
VBox_GAs_5.2.20 (/dev/sr0)
Use ISO image:
Browse...
Automatically detect operating system based on install media
OS type: Unknown
Version: Unknown
Cancel
Back
Forward

Final output



Practical - 6

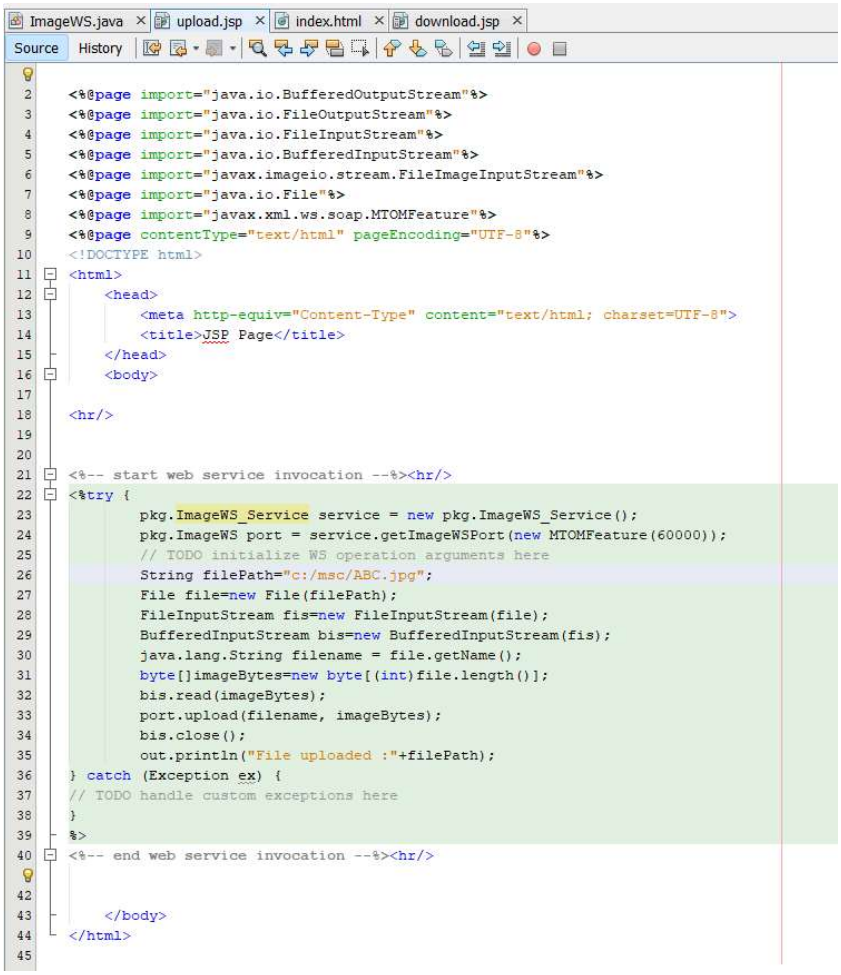
Aim : Develop application to download image/video from server or upload image/video to server using MTOM techniques

Step 1:

MTOMClient

Index.html

upload.jsp




```jsp
<%@page import="java.io.BufferedOutputStream"%>
<%@page import="java.io.FileOutputStream"%>
<%@page import="java.io.FileInputStream"%>
<%@page import="java.io.BufferedInputStream"%>
<%@page import="javax.imageio.stream.FileImageInputStream"%>
<%@page import="java.io.File"%>
<%@page import="javax.xml.ws.soap.MTOMFeature"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

<hr/>


<%-- start web service invocation --%><hr/>
<%try {
        pkg.ImageWS_Service service = new pkg.ImageWS_Service();
        pkg.ImageWS port = service.getImageWSPort(new MTOMFeature(60000));
        // TODO initialize WS operation arguments here
        String filePath="c:/msc/ABC.jpg";
        File file=new File(filePath);
        FileInputStream fis=new FileInputStream(file);
        BufferedInputStream bis=new BufferedInputStream(fis);
        java.lang.String filename = file.getName();
        byte[]imageBytes=new byte[(int)file.length()];
        bis.read(imageBytes);
        port.upload(filename, imageBytes);
        bis.close();
        out.println("File uploaded :"+filePath);
} catch (Exception ex) {
// TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>


    </body>
</html>
```

```jsp
<%@page import="java.io.BufferedOutputStream">
<%@page import="java.io.FileOutputStream">
<%@page import="java.io.FileInputStream">
<%@page import="java.io.BufferedInputStream">
<%@page import="javax.imageio.stream.FileImageInputStream">
<%@page import="java.io.File">
<%@page import="javax.xml.ws.soap.MTOMFeature">
<%@page contentType="text/html" pageEncoding="UTF-8">
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%-- start web service invocation --%><hr/>
<%
try {
        pkg.ImageWS_Service service = new pkg.ImageWS_Service();
        pkg.ImageWS port = service.getImageWSPort(new MTOMFeature(60000));
```

```
        // TODO initialize WS operation arguments here
        String filePath="c:/msc/ABC.jpg";
        File file=new File(filePath);
        FileInputStream fis=new FileInputStream(file);
        BufferedInputStream bis=new BufferedInputStream(fis);
        java.lang.String filename = file.getName();
        byte[]imageBytes=new byte[(int)file.length()];
        bis.read(imageBytes);
        port.upload(filename, imageBytes);
        bis.close();
        out.println("File uploaded :"+filePath);
} catch (Exception ex) {
        // TODO handle custom exceptions here
}
> <%-- end web service invocation --><hr/>
```

download.jsp

ImageWS.java | upload.jsp | index.html | download.jsp

Source | History

```
<%@page import="java.io.BufferedOutputStream"%>
<%@page import="java.io.FileOutputStream"%>
<%@page import="java.io.FileInputStream"%>
<%@page import="java.io.BufferedInputStream"%>
<%@page import="javax.imageio.stream.FileImageInputStream"%>
<%@page import="java.io.File"%>
<%@page import="javax.xml.ws.soap.MTOMFeature"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

<hr/>



<%-- start web service invocation --%><hr/>
<%
try {
        pkg.ImageWS_Service service = new pkg.ImageWS_Service();
        pkg.ImageWS port = service.getImageWSPort();
        // TODO initialize WS operation arguments here
        java.lang.String filename = "ABC.jpg";
        String filePath="c:/msc/download/"+filename;
        // TODO process result here
        byte[] fileBytes = port.download(filename);
        FileOutputStream fos=new FileOutputStream(filePath);
        BufferedOutputStream bos=new BufferedOutputStream(fos);
        bos.write(fileBytes);
        bos.close();
        out.println("File downloaded"+filePath);
} catch (Exception ex) {
// TODO handle custom exceptions here
}
%> <%-- end web service invocation --%><hr/>

    </body>
</html>

```

<%@page import="java.io.BufferedOutputStream">
<%@page import="java.io.FileOutputStream">
<%@page import="java.io.FileInputStream">
<%@page import="java.io.BufferedInputStream">
<%@page import="javax.imageio.stream.FileImageInputStream">
<%@page import="java.io.File">
<%@page import="javax.xml.ws.soap.MTOMFeature">
<%@page contentType="text/html" pageEncoding="UTF-8">
<!DOCTYPE html>
<html>

```
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>

<%-- start web service invocation --><hr/>
<%
try {
	pkg.ImageWS_Service service = new pkg.ImageWS_Service();
	pkg.ImageWS port = service.getImageWSPort();
	// TODO initialize WS operation arguments here
	java.lang.String filename = "ABC.jpg";
	String filePath="c:/msc/download/"+filename;
	// TODO process result here
	byte[] fileBytes = port.download(filename); FileOutputStream
	fos=new FileOutputStream(filePath); BufferedOutputStream
	bos=new BufferedOutputStream(fos); bos.write(fileBytes);
	bos.close();
	out.println("File downloaded"+filePath);
} catch (Exception ex) {
	// TODO handle custom exceptions here
}
> <%-- end web service invocation --><hr/>
</body>
</html>
```

Step 2 :

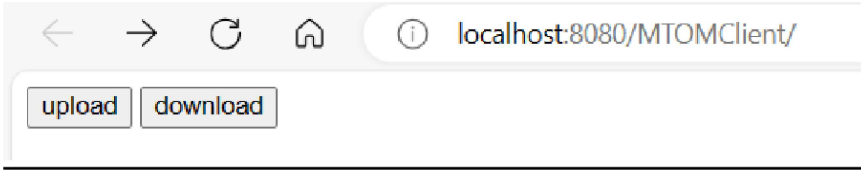
MTOMServer

ImageWS.java

```
package mypkg;
import java.io.*;
import javax.jws.Oneway;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.xml.ws.soap.MTOM;

@MTOM(enabled=true,threshold=60000)
@WebService(serviceName = "ImageWS")
public class ImageWS {
	@WebMethod(operationName = "upload")
	@Oneway
	public void upload(@WebParam(name = "Filename") String Filename, @WebParam(name =
```

```
                "ImageBytes") byte[] ImageBytes)
                {
                                                        String filePath="C:/MSC/upload/"+Filename;
                try
                {
                                                        FileOutputStream
                                                        fos=new
                                                        FileOutputStream(filePat
                                                        h);
                                                        BufferedOutputStream
                }                                       bos=new
                                                        BufferedOutputStream(f
                                                        os);
                                                        bos.write(ImageBytes);
                                                        bos.close();
                                                        System.out.println("Recieved file :"+filePath);
                catch(Exception ex)
                {
                        System.out.println(ex);
                }

                }
                @WebMethod(operationName = "download")
                public byte[] download(@WebParam(name = "Filename") String Filename) {
                String filePath="C:/MSC/upload/"+Filename;
                System.out.println("Sending file :"+filePath);
                try
                {
                        File file=new File(filePath);
                        FileInputStream fis=new FileInputStream(file);
                        BufferedInputStream bis=new BufferedInputStream(fis);
                        byte[] fileBytes=new byte[(int)file.length()];
                        bis.read(fileBytes);
                        bis.close();
                        return fileBytes;
                }
                catch(Exception ex)
                {
                        System.out.println(ex);
                }
                return null;
        }
}
```

Output:



Practical 7

Aim: Implement FOSS-Cloud Functionality VSI (Virtual Server Infrastructure) Infrastructure as a Service (IaaS), Creating Virtual Machine or Storage

FOSS Cloud Installation:

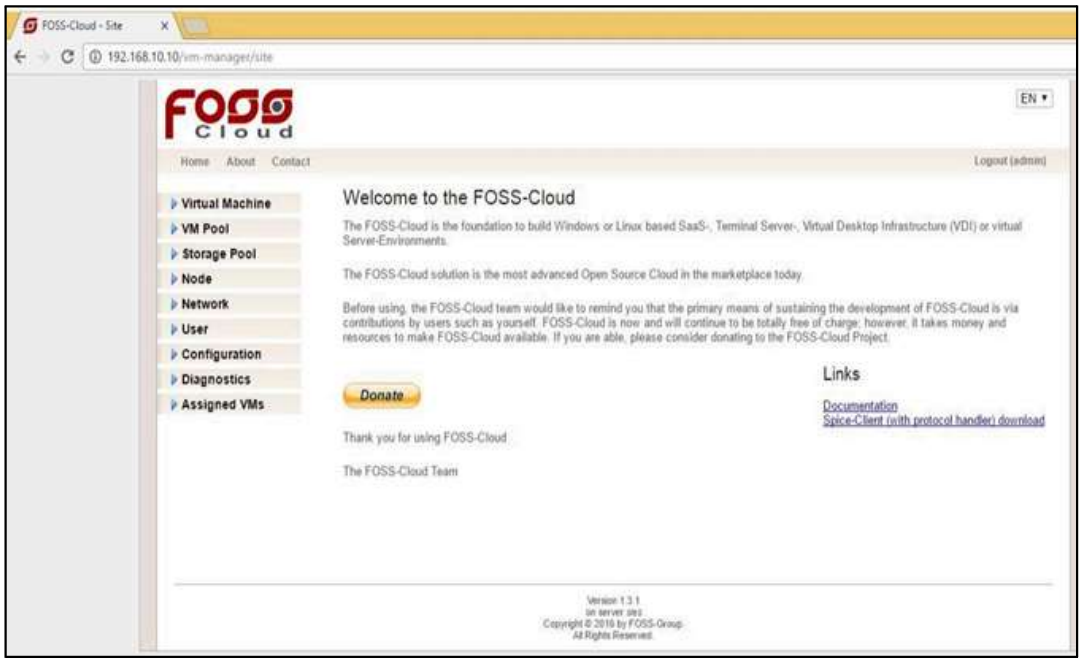
Steps:

1. Choose your keymap.
2. Confirm that you want to start: yes
3. Choose Demo-System: 1
4. Choose a Block-Device: sda
5. Confirm that you want to continue: yes
6. Confirm that you want to continue: yes
7. Choose the network interface: eth0

8. Choose if you want to use automatic network configuration: Yes 9. Reboot your system: yes
10. Login as root and run "fc-node-configuration -n demo-system --password admin"

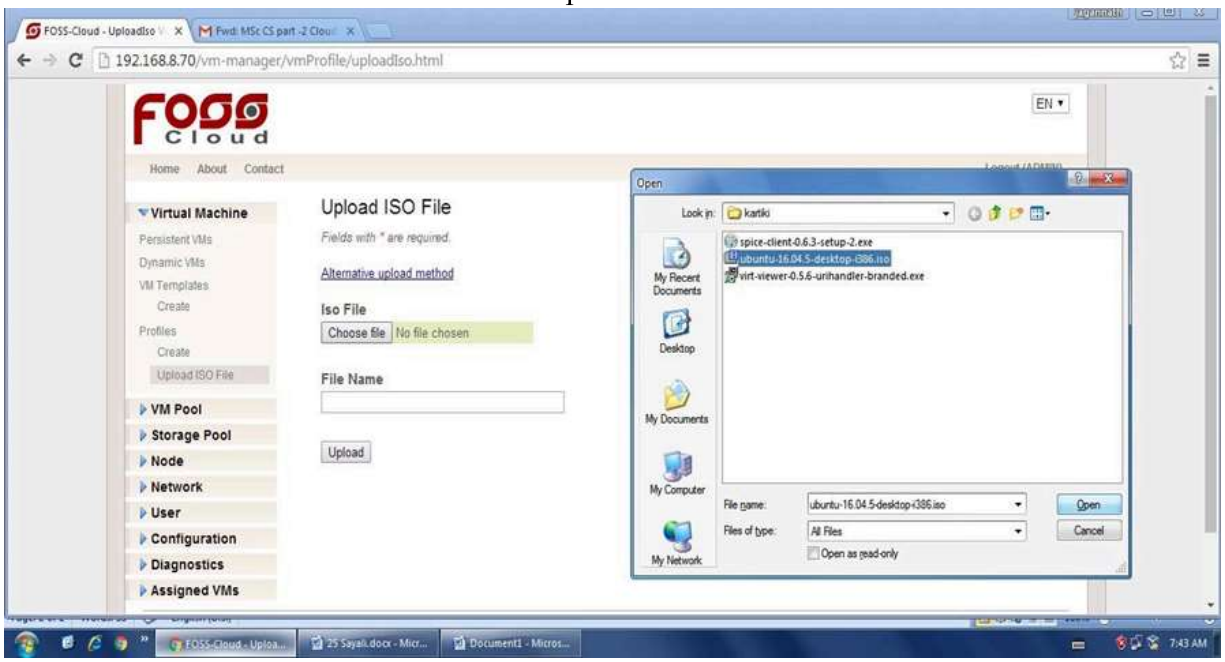
Login as admin
Password : admin

Home page of FOSS Cloud at Client



Uploading ISO files to foss cloud

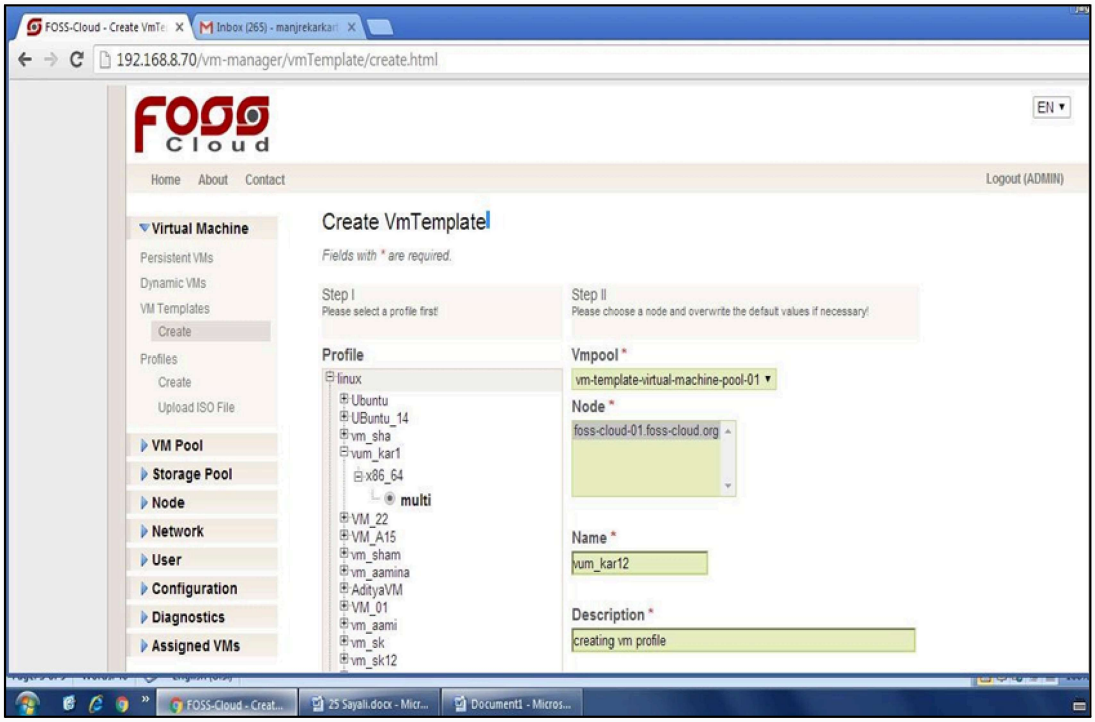
- Go to virtual machine>Profile>Upload ISO file >file name.



Creating a Profile

The profile creates the relationship between the ISO file and the FOSS-Cloud.

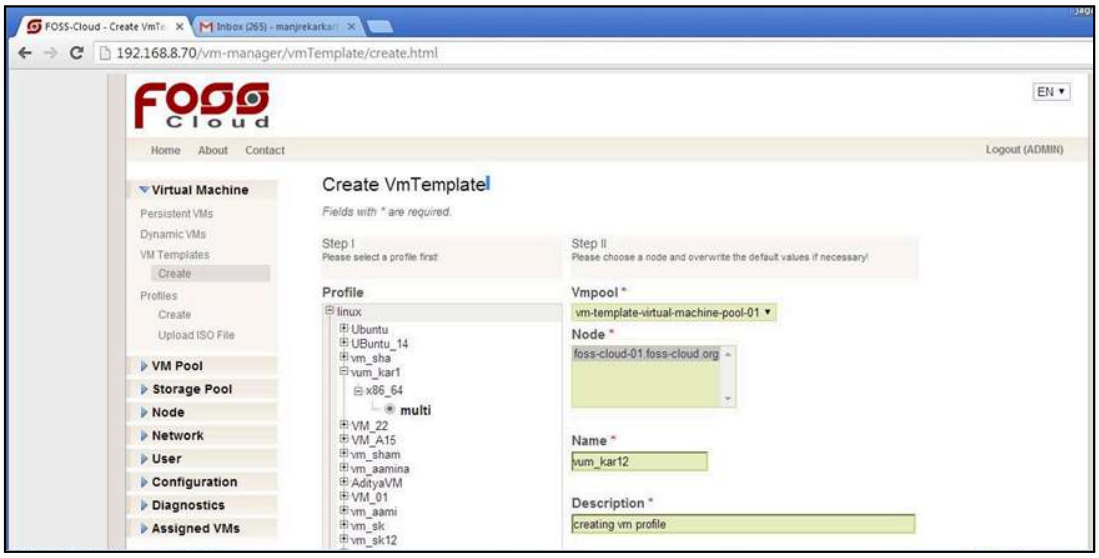
1. Open Virtual Machines
2. Choose vm-Profiles
3. Choose the right Base Profile
4. Choose the right architecture (Windows only x86_64)
5. Chose the language (it is a information - not keyboard relevant)
6. Choose the ISO file
7. Fill out name and description
8. Choose the amount of memory and volume capacity
9. Choose amount of CPU
10. Choose clock offset (normally Windows is "localtime" and Linux is "utc")



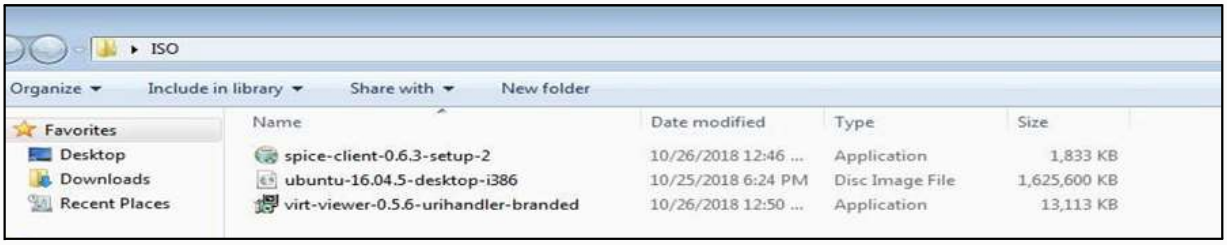
Creating Template

1. Choose the profile you have prepared before
2. Add the vm-pool and one or more nodes, where you will run this vm (when the chosen vm-pool has only one node assigned, you don't have a choice)
3. You can change all the other informations you have entered before

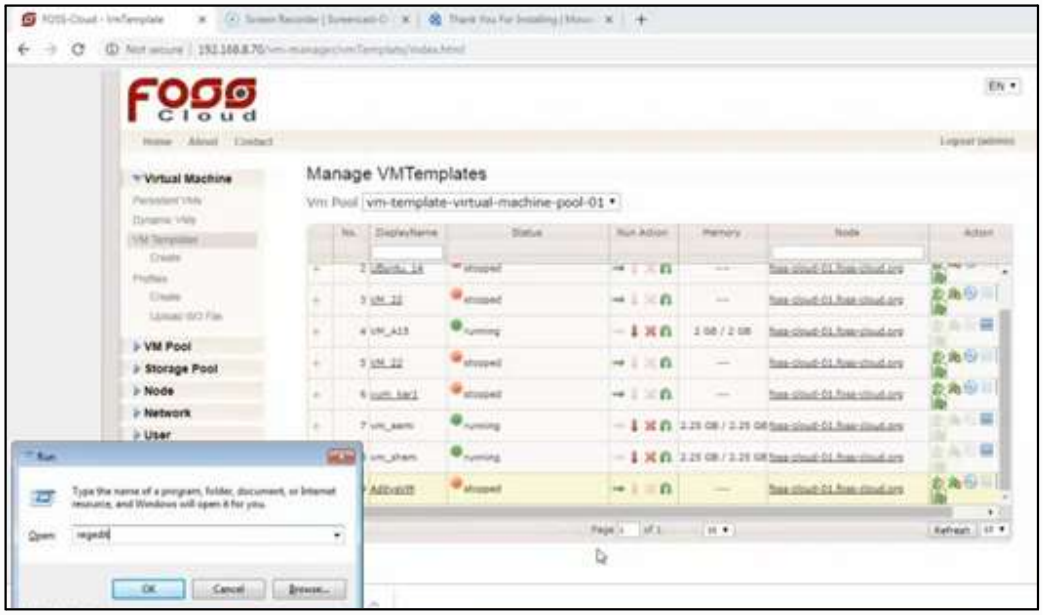
Click on "create" and the template is ready for installing the guest operating system.



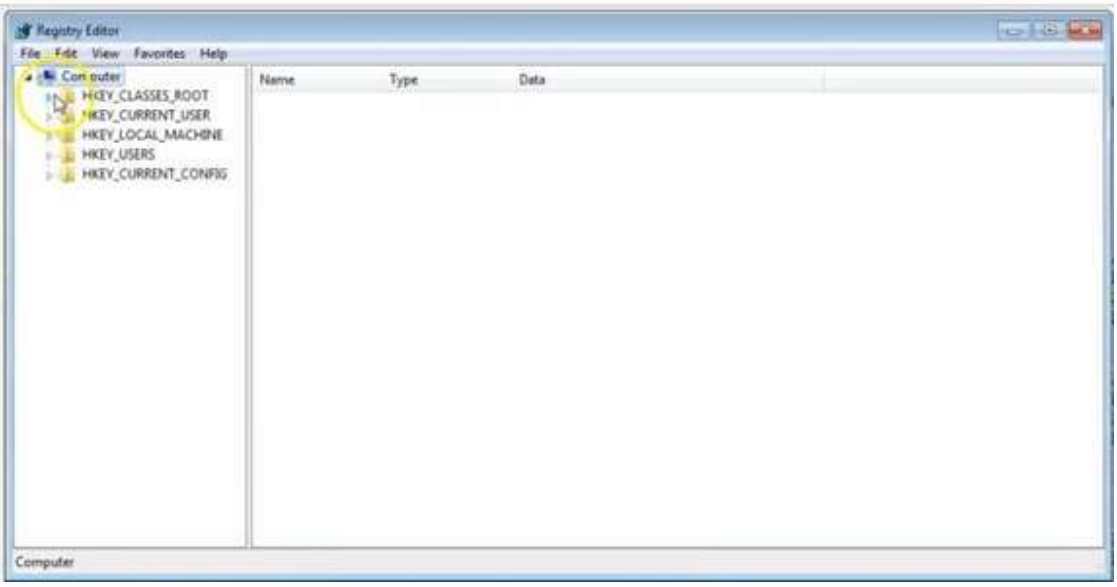
In stall virt Viewer and spice

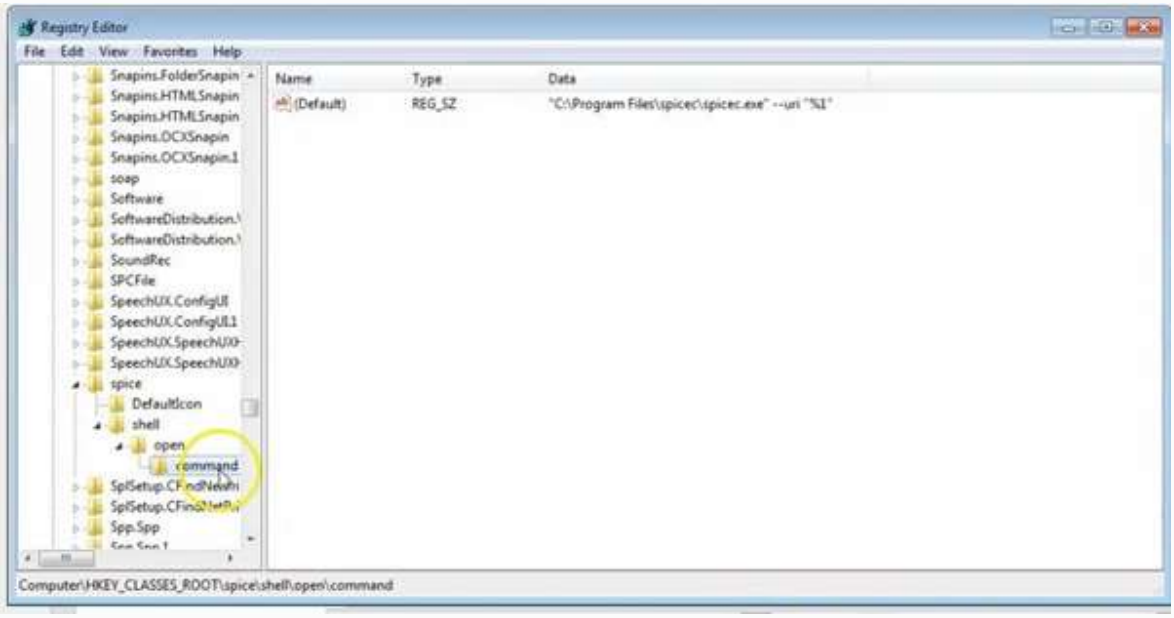


Go to regedit

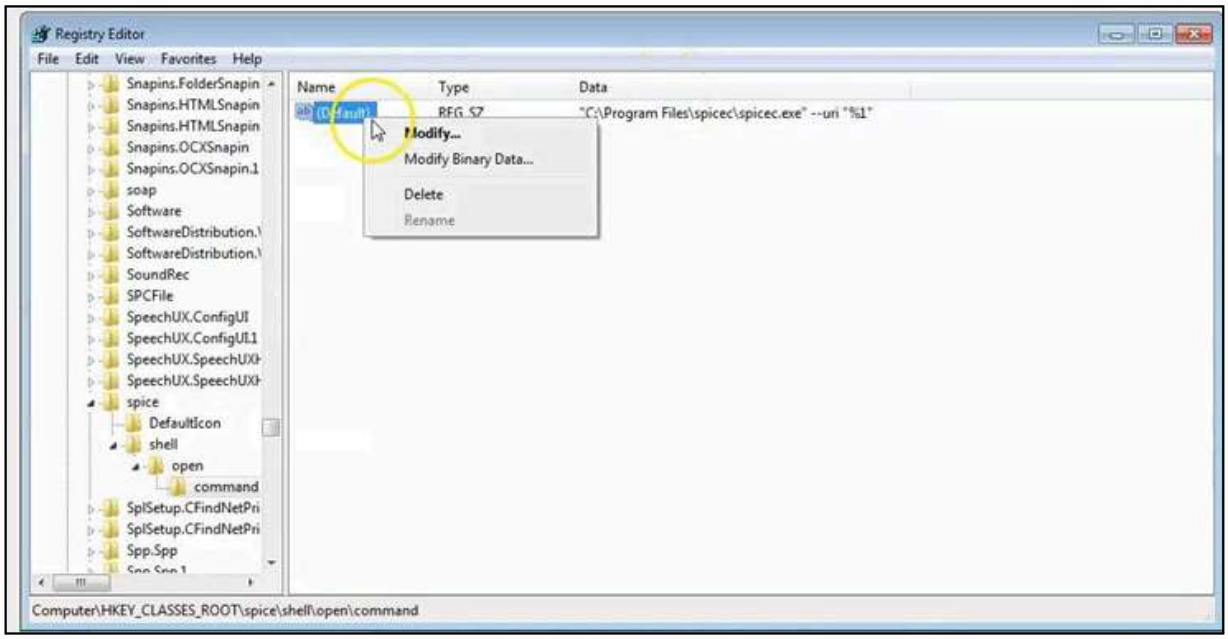


Go to Computer > HKEY_CLASSES_ROOT > SPICE SHELL > COMMAND

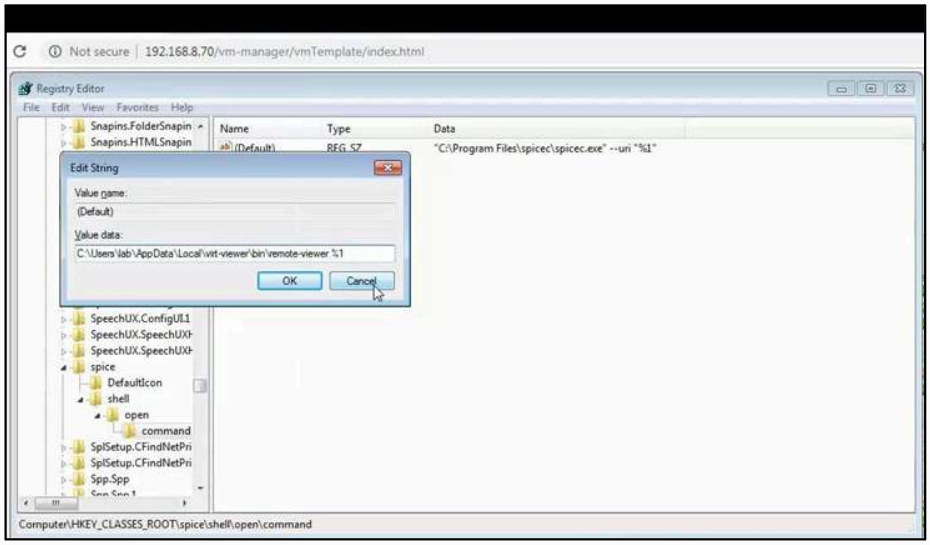




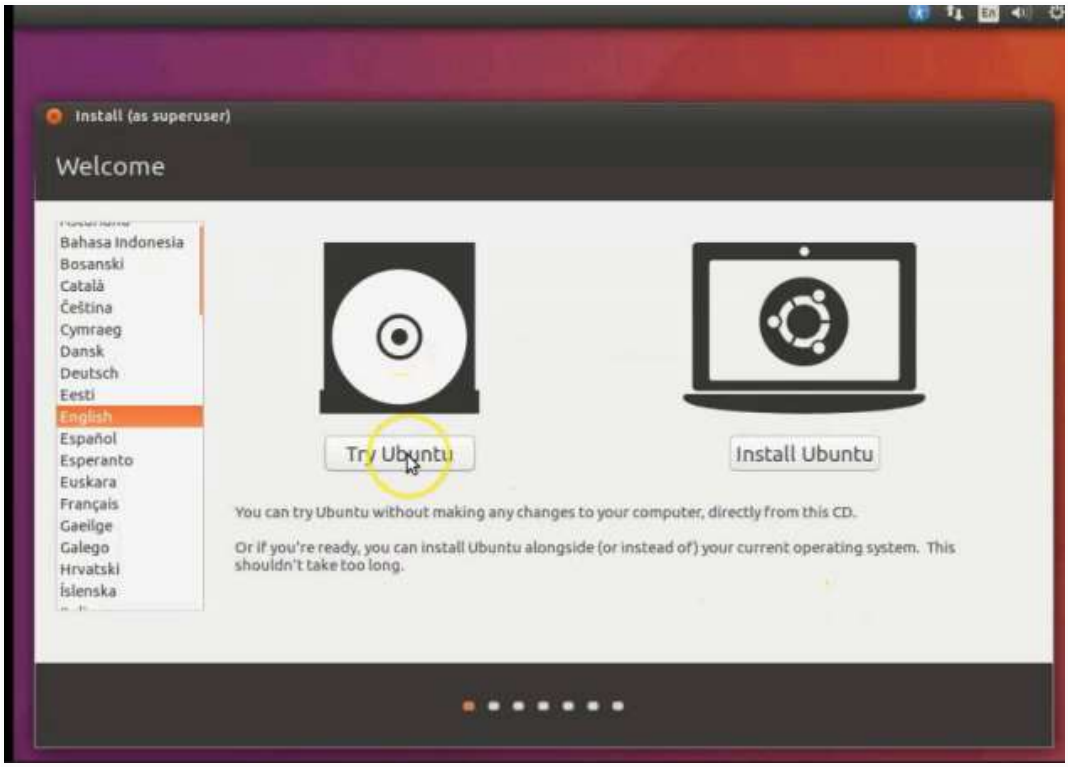
Go to Default and click Modify



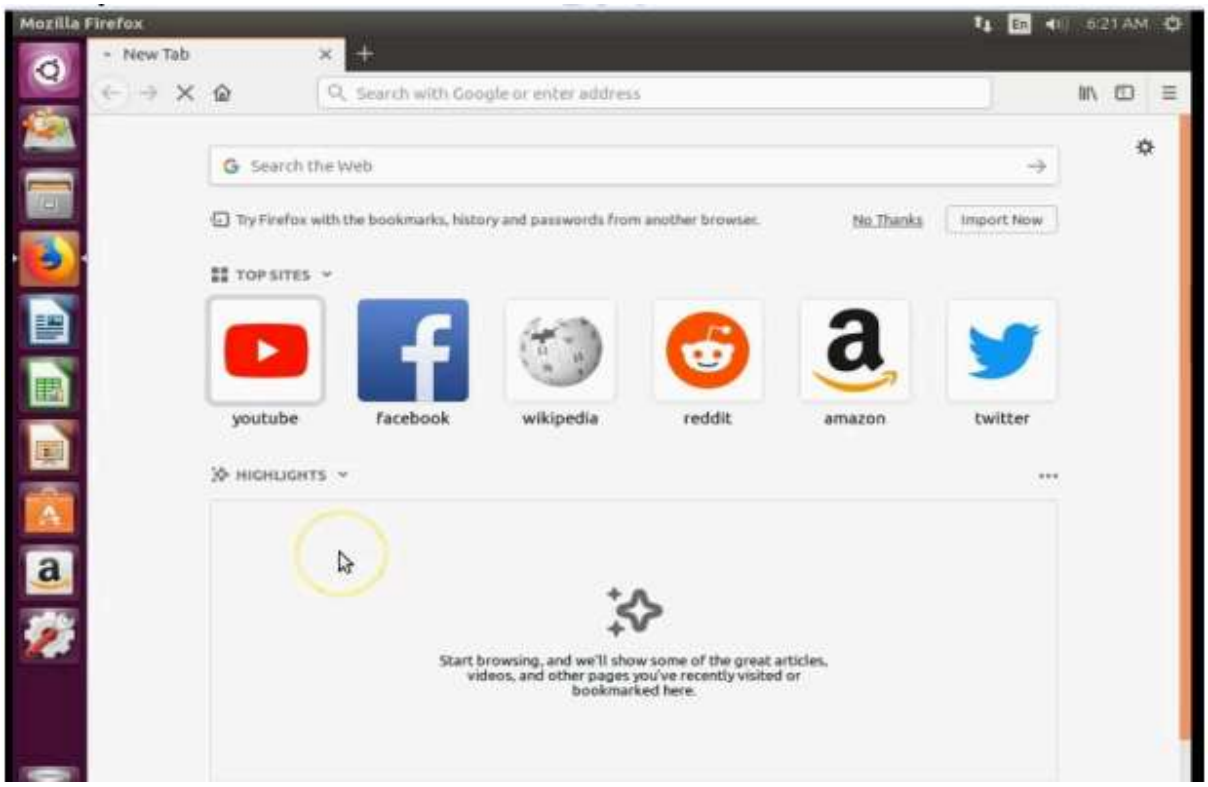
Paste path of virt viewerà virt_viewer_path %1 (Actual path of Virt Viewer)



Run Your VM and View using VM Template



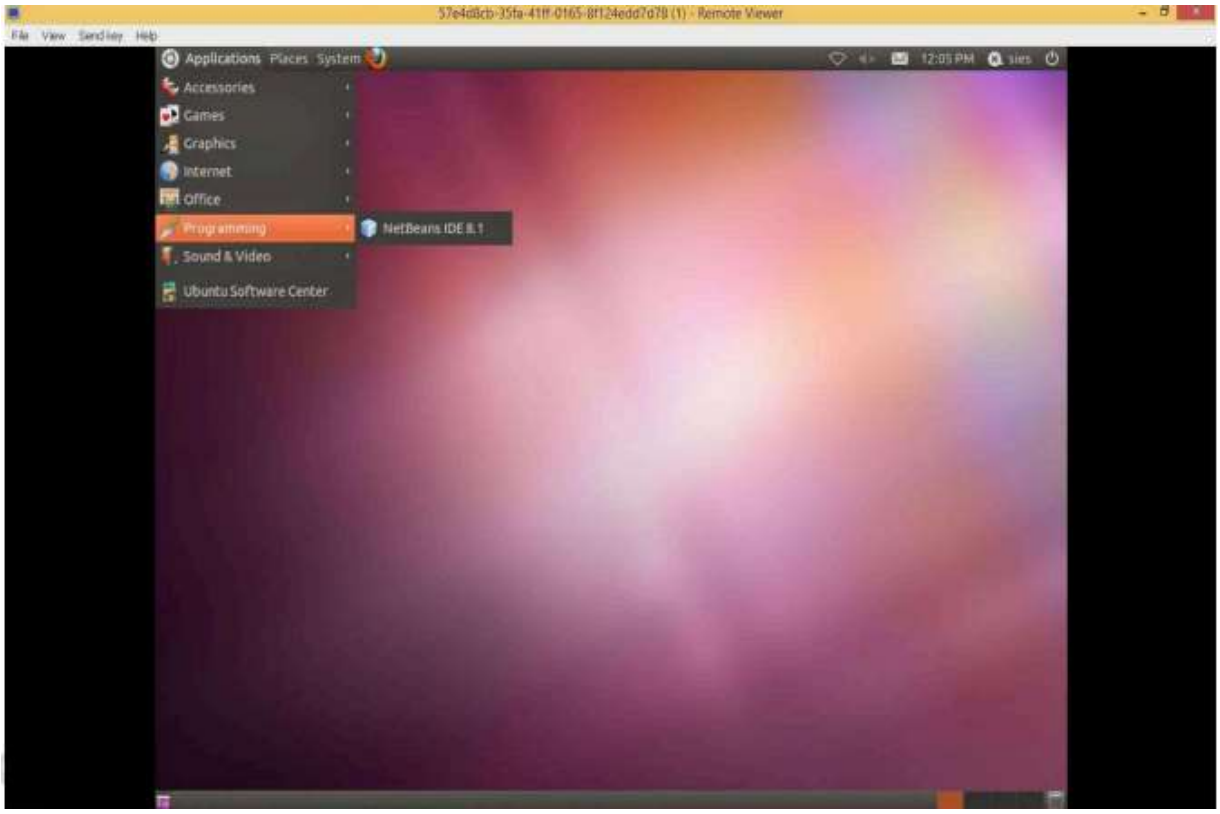
Open Browser in Ubuntu.


Mozilla Firefox
6:21 AM
New Tab
Search with Google or enter address
Search the Web
Try Firefox with the bookmarks, history and passwords from another browser.
No Thanks
Import Now
TOP SITES
youtube
facebook
wikipedia
reddit
amazon
twitter
HIGHLIGHTS
Start browsing, and we'll show some of the great articles, videos, and other pages you've recently visited or bookmarked here.

Practical 8

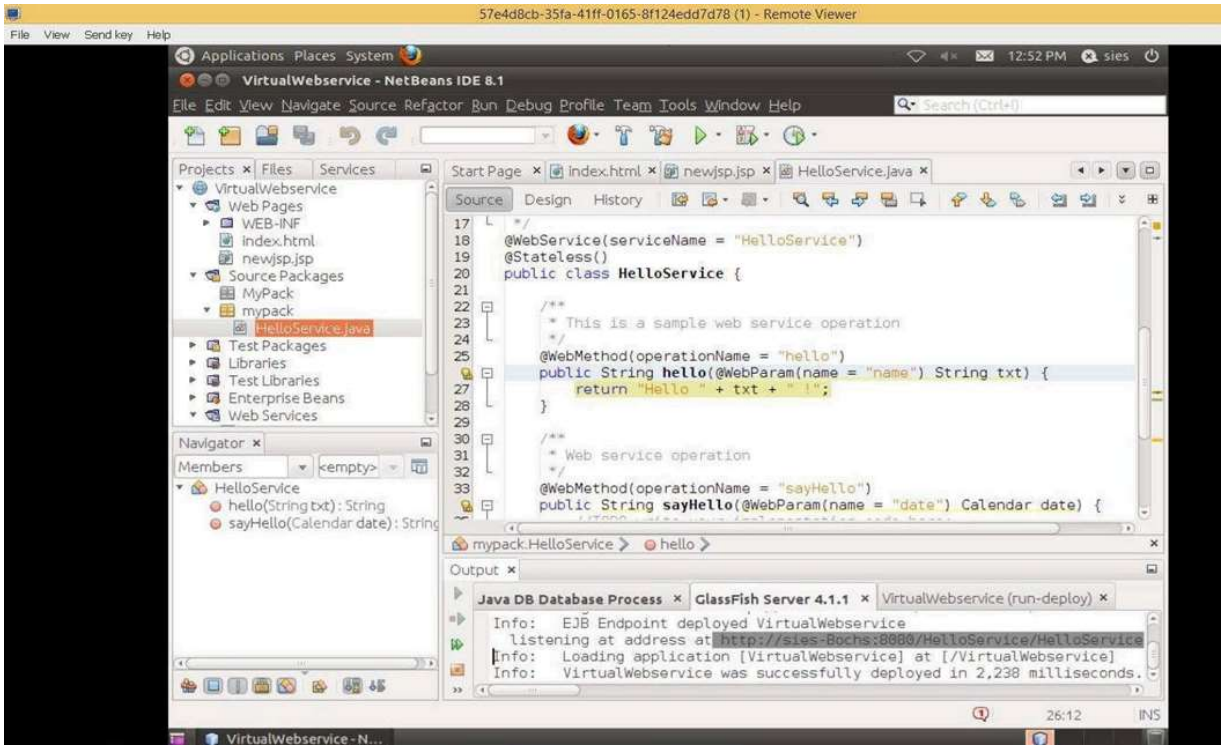
Aim: Implement FOSS-Cloud Functionality - VSI Platform as a Service (PaaS)

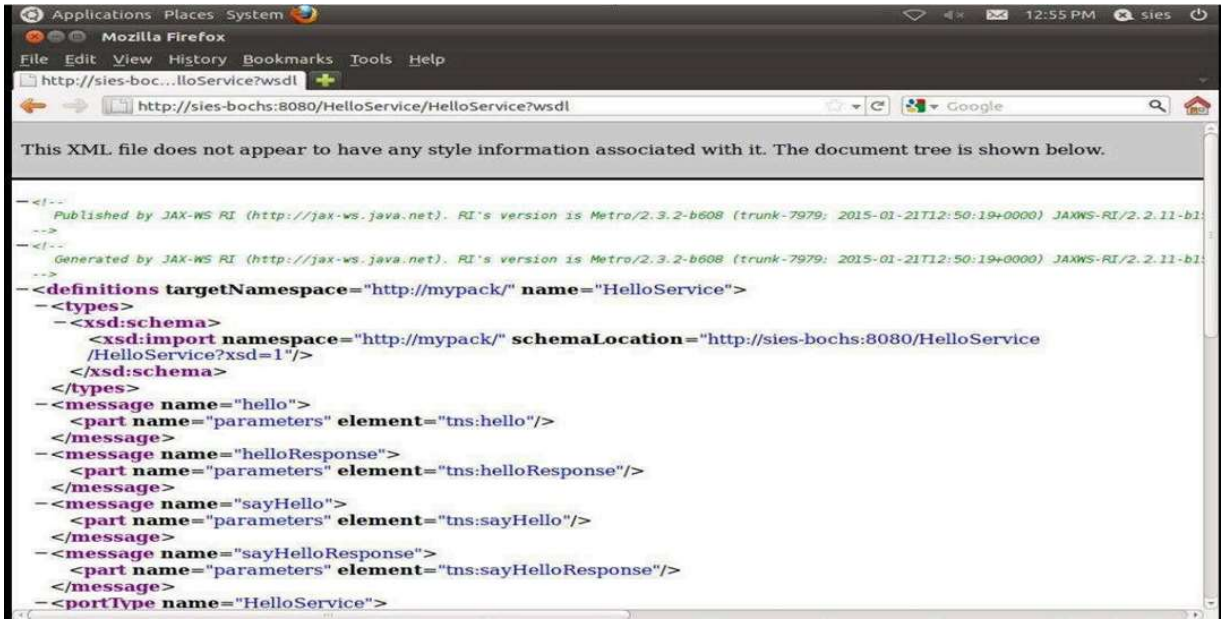
Software development Kits can be made available in the Virtual Machines That can be implemented as Platform as a Service.
Installation of Netbeans, Eclipse, Visual Studio and DBMS can be done in the appropriate Virtual Machines.



Aim: Implement FOSS-Cloud Functionality - VSI Software as a Service (SaaS)

Applications created and deployed in the virtual machines can be accessed by the outside world. This can be implemented as Software as a Service


57e4d8cb-35fa-41ff-0165-8f124edd7d78 (1) - Remote Viewer
File View Send key Help
Applications Places System
12:52 PM sies
VirtualWebservice - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
VirtualWebservice
Web Pages
WEB-INF
index.html
newjsp.jsp
Source Packages
MyPack
mypack
HelloService.java
Test Packages
Libraries
Test Libraries
Enterprise Beans
Web Services
Navigator
Members
HelloService
hello(String txt) : String
sayHello(Calendar date) : String
Start Page
index.html
newjsp.jsp
HelloService.java
Source Design History
@WebService(serviceName = "HelloService")
@Stateless()
public class HelloService {
/**
* This is a sample web service operation
*/
@WebMethod(operationName = "hello")
public String hello(@WebParam(name = "name") String txt) {
return "Hello " + txt + " !";
}
/**
* Web service operation
*/
@WebMethod(operationName = "sayHello")
public String sayHello(@WebParam(name = "date") Calendar date) {
mypack.HelloService
hello
Output
Java DB Database Process
GlassFish Server 4.1.1
VirtualWebservice (run-deploy)
Info: EJB Endpoint deployed VirtualWebservice
listening at address at http://sies-Bochs:8080/HelloService/HelloService
Info: Loading application [VirtualWebservice] at [/VirtualWebservice]
Info: VirtualWebservice was successfully deployed in 2,238 milliseconds.
26:12
INS
VirtualWebservice - N...


Applications Places System
12:55 PM sies
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://sies-boc...lloService?wsdl
http://sies-bochs:8080/HelloService/HelloService?wsdl
Google
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<!--
Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b1
-->
<!--
Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b1
-->
<definitions targetNamespace="http://mypack/" name="HelloService">
<types>
<xsd:schema>
<xsd:import namespace="http://mypack/" schemaLocation="http://sies-bochs:8080/HelloService
/HelloService?xsd=1"/>
</xsd:schema>
</types>
<message name="hello">
<part name="parameters" element="tns:hello"/>
</message>
<message name="helloResponse">
<part name="parameters" element="tns:helloResponse"/>
</message>
<message name="sayHello">
<part name="parameters" element="tns:sayHello"/>
</message>
<message name="sayHelloResponse">
<part name="parameters" element="tns:sayHelloResponse"/>
</message>
<portType name="HelloService">

Practical - 10

Aim: How to Install OpenStack on Ubuntu with DevStack

Minimum Requirements

Before we begin, ensure you have the following minimum prerequisites

1. A fresh Ubuntu 18.04 installation
2. User with sudo privileges
3. 4 GB RAM
4. 2 vCPUs
5. Hard disk capacity of 10 GB
6. Internet connection

Step 1: Update and Upgrade the System

To start off, log into your Ubuntu 18.04 system using SSH protocol and update & upgrade system repositories using the following command.
apt update -y && apt upgrade -y

Sample Output

```
root@ubuntu:/# apt update -y && apt ugrade -y
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:5 http://archive.canonical.com/ubuntu bionic InRelease [10.2 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:7 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:8 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:9 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
Get:10 http://us-central1.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [627 kB]
```

Next reboot the system using the command.
sudo reboot
OR
init 6

Step 2: Create Stack user and assign sudo priviledge

Best practice demands that devstack should be run as a regular user with sudo privileges. With that in mind, we are going to add a new user called “stack” and assign sudo privileges. To create stack user execute
sudo adduser -s /bin/bash -d /opt/stack -m stack

Next, run the command below to assign sudo privileges to the user
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

Sample Output

```
root@ubuntu:/# sudo useradd -s /bin/bash -d /opt/stack -m stack
root@ubuntu:/#
root@ubuntu:/# echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
stack ALL=(ALL) NOPASSWD: ALL
root@ubuntu:/#
```

Step 3: Install git and download DevStack

Once you have successfully created the user ‘stack’ and assigned sudo privileges, switch to the user using the command.
su - stack

In most Ubuntu 18.04 systems, git comes already installed. If by any chance git is missing, install it by running the following command.
sudo apt install git -y

Sample output

```
root@ubuntu:~# su - stack
stack@ubuntu:~$
stack@ubuntu:~$ sudo apt install git -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.17.1-1ubuntu0.4).
The following packages were automatically installed and are no longer required:
  grub-pc-bin libnuma1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Using git, clone devstack’s git repository as shown.
git clone https://git.openstack.org/openstack-dev/devstack
Sample output

```
stack@ubuntu:~$ git clone https://git.openstack.org/openstack-dev/devstack
Cloning into 'devstack'...
warning: redirecting to https://opendev.org/openstack/devstack/
remote: Enumerating objects: 43615, done.
remote: Counting objects: 100% (43615/43615), done.
remote: Compressing objects: 100% (12575/12575), done.
remote: Total 43615 (delta 31152), reused 42370 (delta 30360)
Receiving objects: 100% (43615/43615), 8.27 MiB | 24.61 MiB/s, done.
Resolving deltas: 100% (31152/31152), done.
stack@ubuntu:~$
stack@ubuntu:~$ ls
devstack
stack@ubuntu:~$
```

Step 4: Create devstack configuration file

In this step, navigate to the devstack directory.
cd devstack

Then create a local.conf configuration file.
vim local.conf

Paste the following content
[[local|localrc]]

 # Password for KeyStone, Database, RabbitMQ and Service
ADMIN_PASSWORD=StrongAdminSecret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Host IP - get your Server/VM IP address from ip addr command
HOST_IP=10.208.0.10

Save and exit the text editor. NOTE:

1. The **ADMIN_PASSWORD** is the password that you will use to log in to the OpenStack login page. The default username is **admin.**
2. The **HOST_IP** is your system's IP address that is obtained by running **ifconfig** or **ip addr** commands.

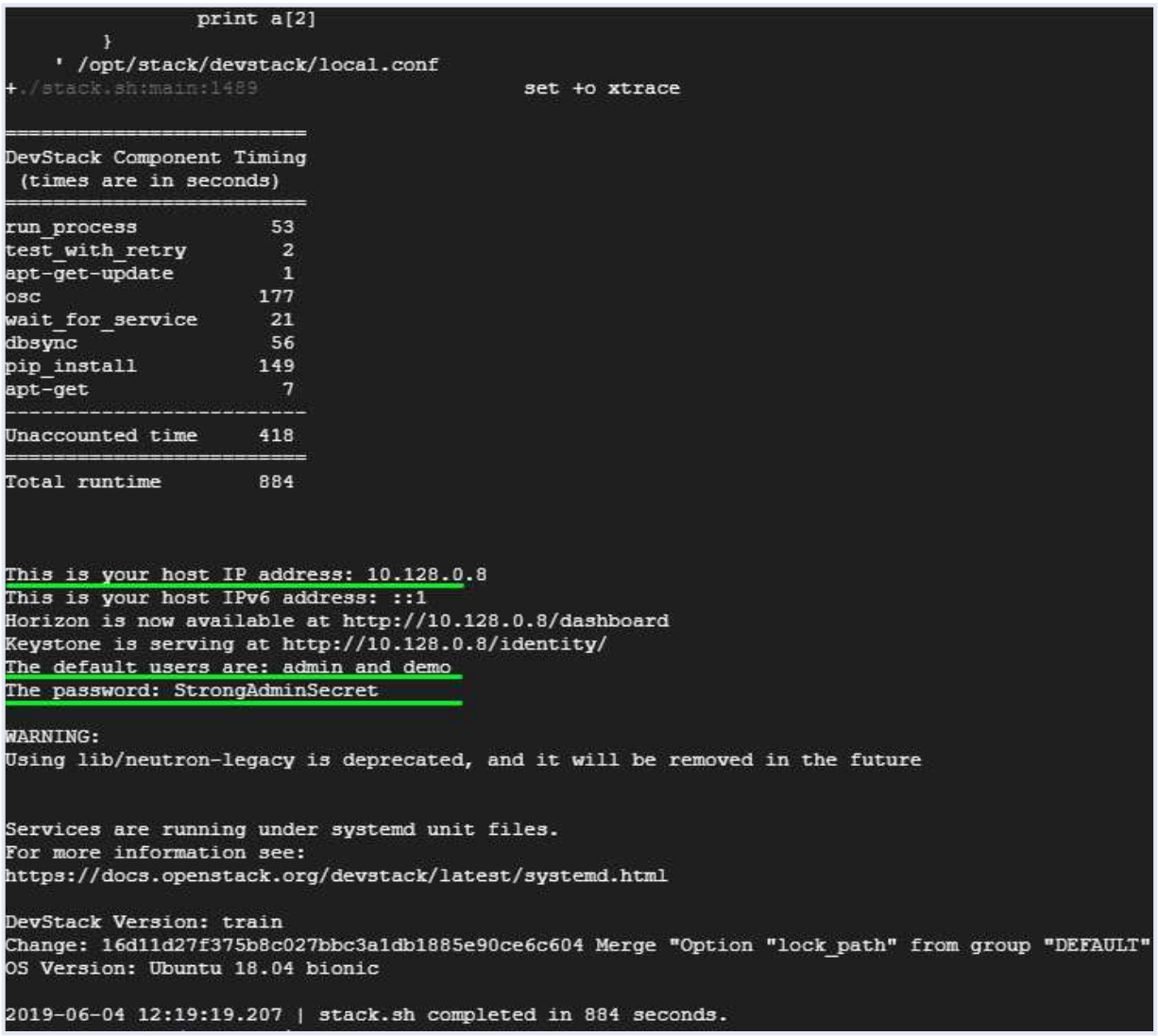
Step 5: Install OpenStack with Devstack

To commence the installation of OpenStack on Ubuntu 18.04, run the script below contained in devstack directory.
./stack.sh

The following features will be installed:

- Horizon – OpenStack Dashboard
- Nova – Compute Service
- Glance – Image Service
- Neutron – Network Service
- Keystone – Identity Service
- Cinder – Block Storage Service
- Placement – Placement API

The deployment takes about 10 to 15 minutes depending on the speed of your system and internet connection. In our case, it took roughly 12 minutes. At the very end, you should see output similar to what we have below.



This confirms that all went well and that we can proceed to access OpenStack via a web browser.

Step 6: Accessing OpenStack on a web browser

To access OpenStack via a web browser browse your Ubuntu's IP address as shown.
https://server-ip/dashboard This directs you to a login page as shown.

Enter the credentials and hit “Sign In” You should be able to see the Management console dashboard as shown below.