

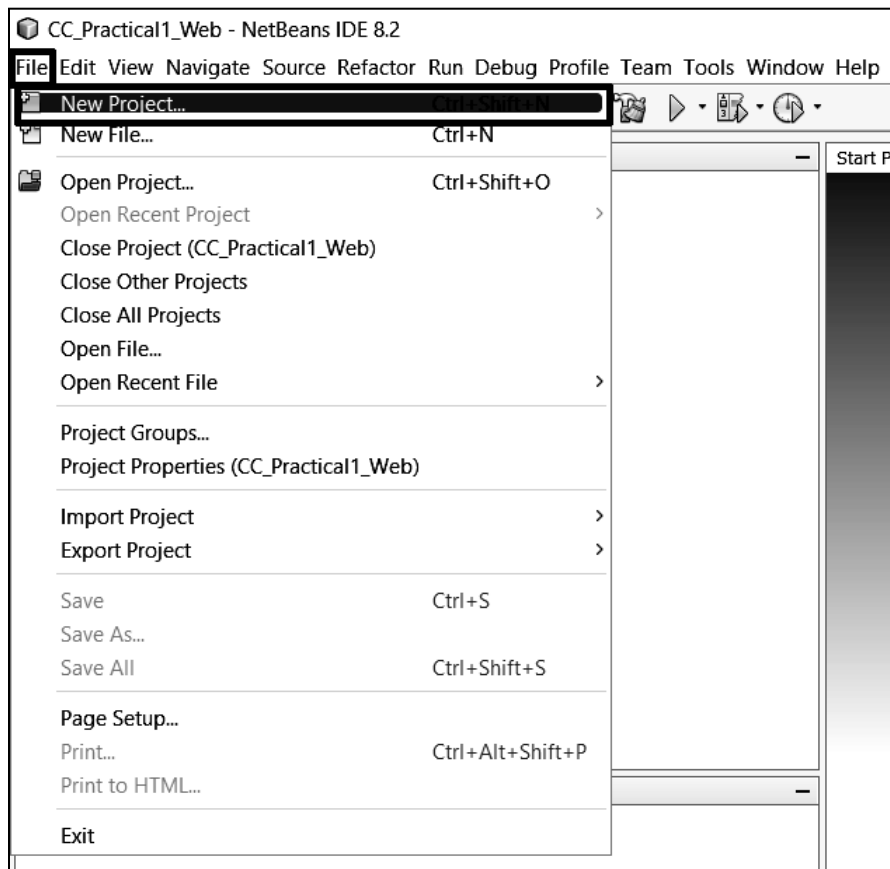
Practical: 1

Aim: Web Service Implementation.

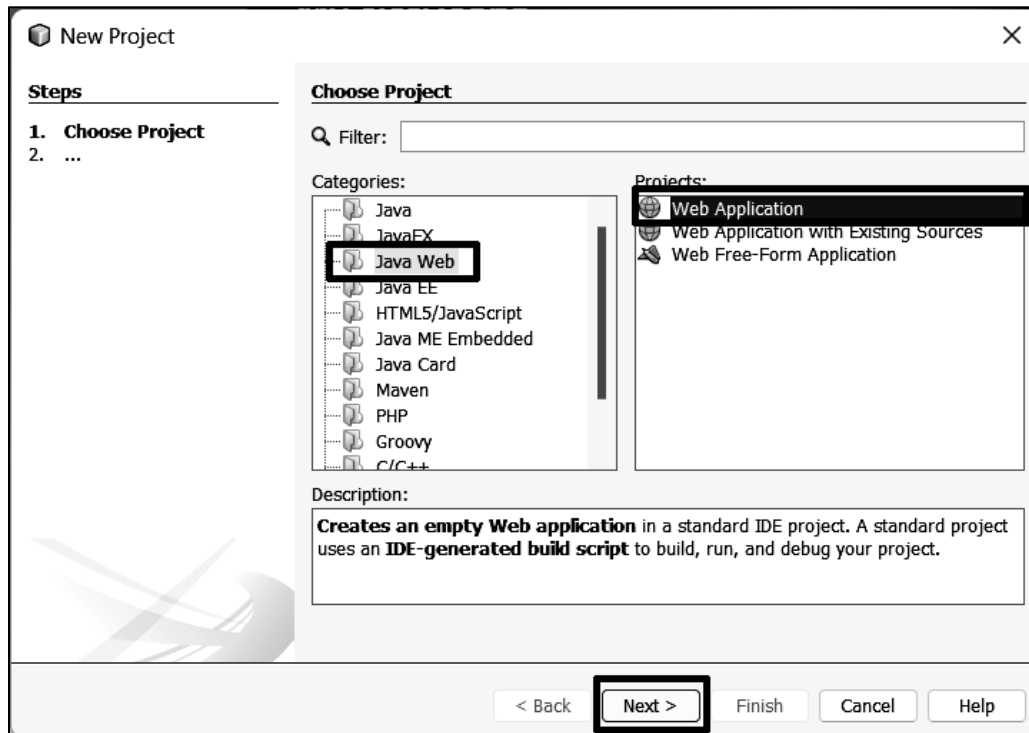
Web Service: A Web service is any software, application or cloud technology system built for interoperable machine-to-machine or application-to-application interaction over a network. They allow different applications to seamlessly communicate and share data with each other.

1) First create java web application project.

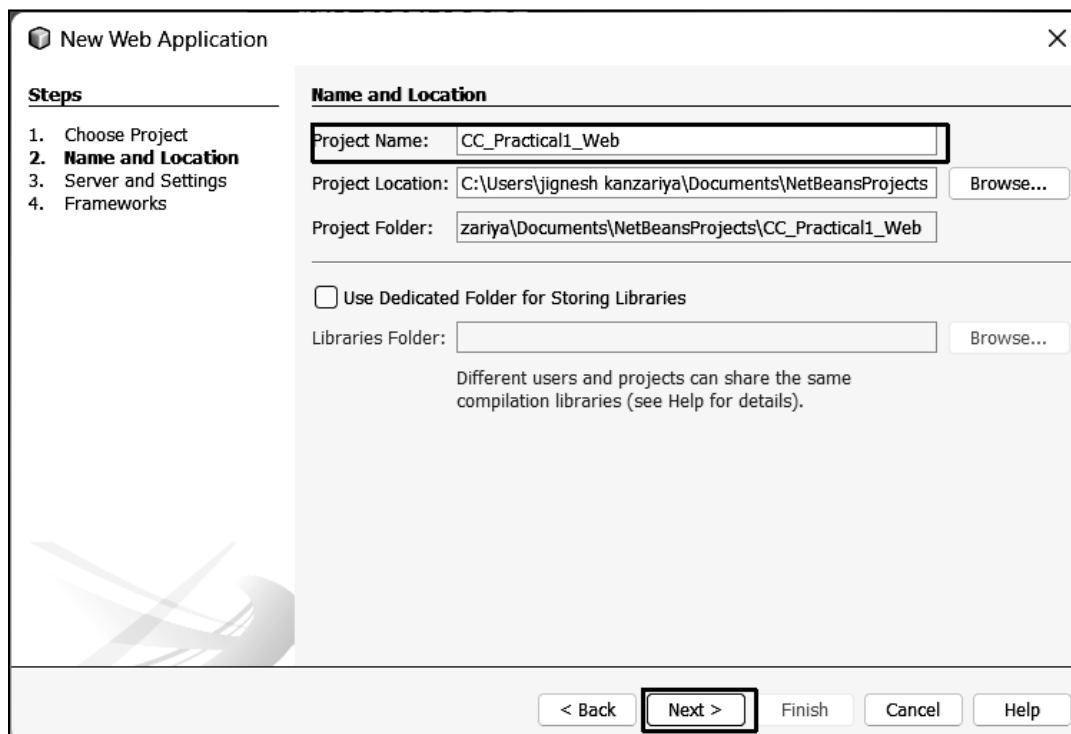
Go to File>New Project



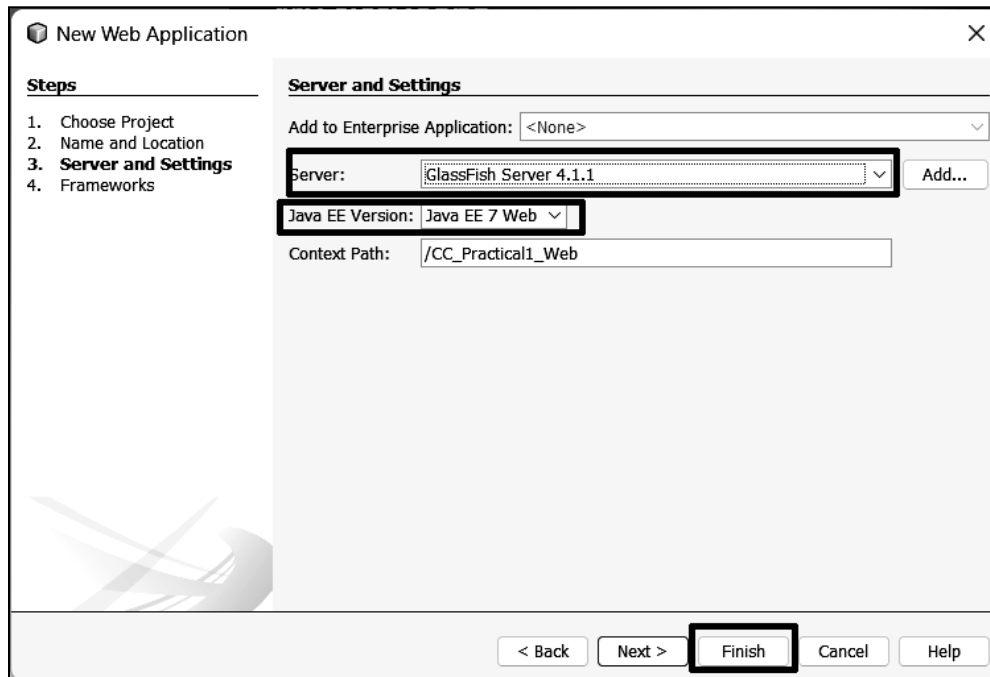
- 2) On right hand side list select “Java Web” Categories and on left hand side list select “Web Application” and then click “Next”



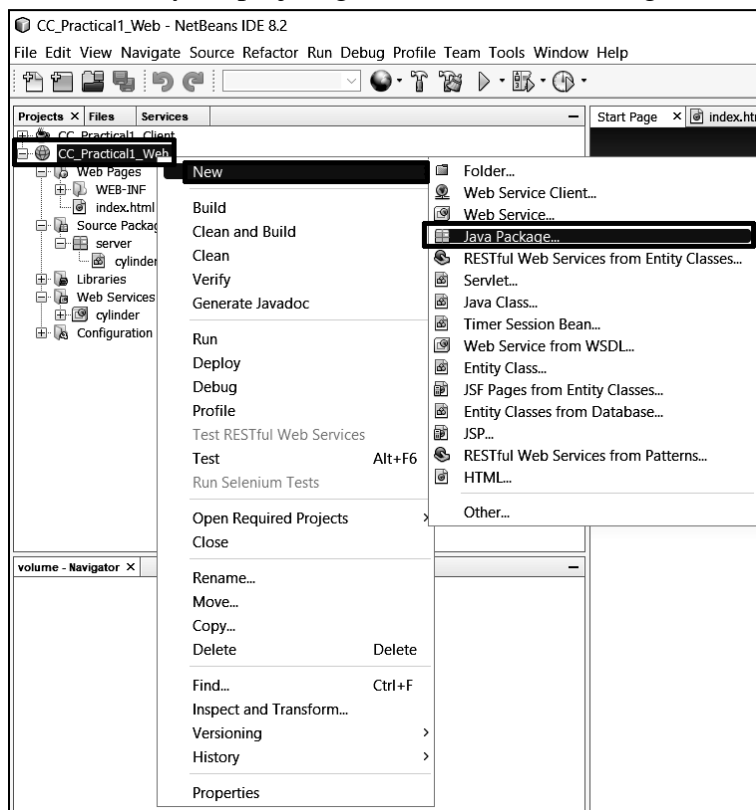
- 3) Enter name of your project in “Project Name” field and click next.



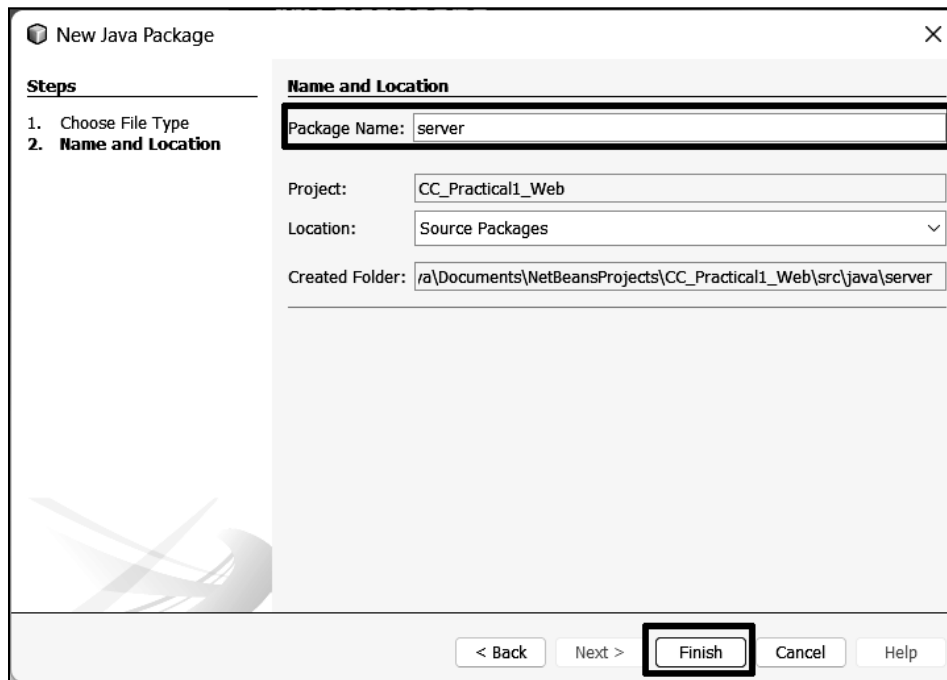
- 4) Select Server:- “GlassFish Server 4.1.1”. Select Java EE Version:- “Java EE 7 Web” then click “Finish” button.



- 5) After that “Add Java Package”. On right side “Projects” tab you can see your project. Right Click on your project go to New > Java Package.

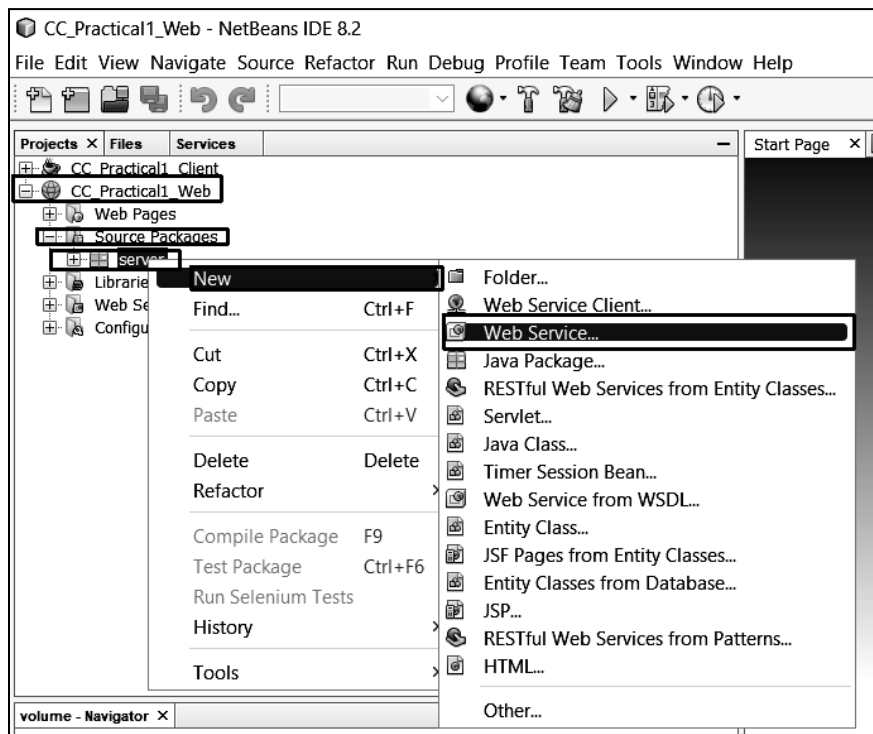


6) Add Java Package Name and click “Finish” button.

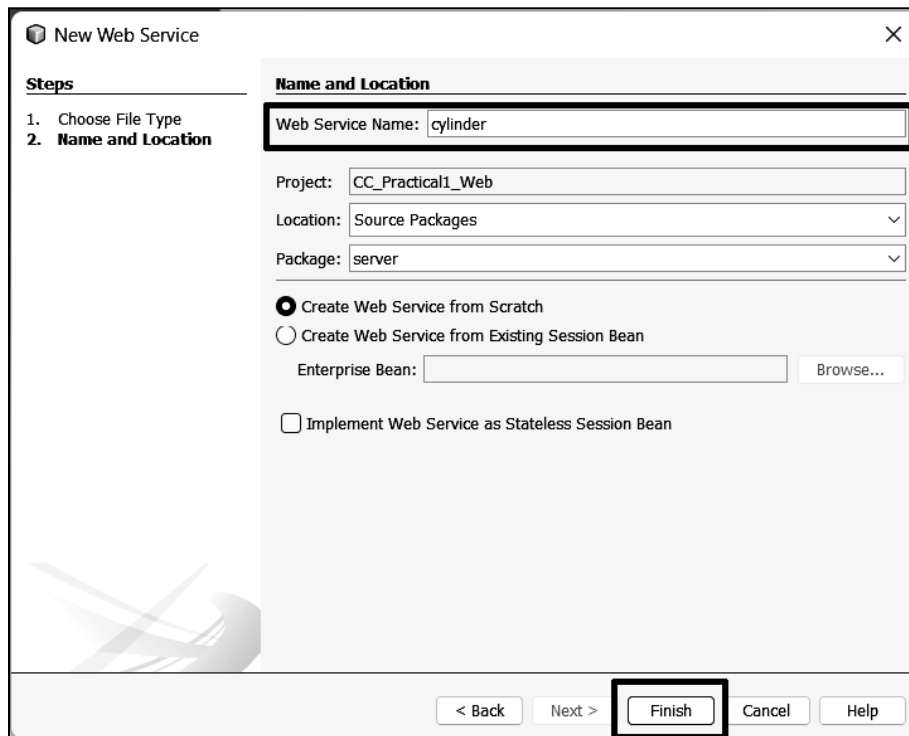


7) Create Web Service.

[your web project name] > Source Packages > [your package name] > Right Click > New > Web Service.

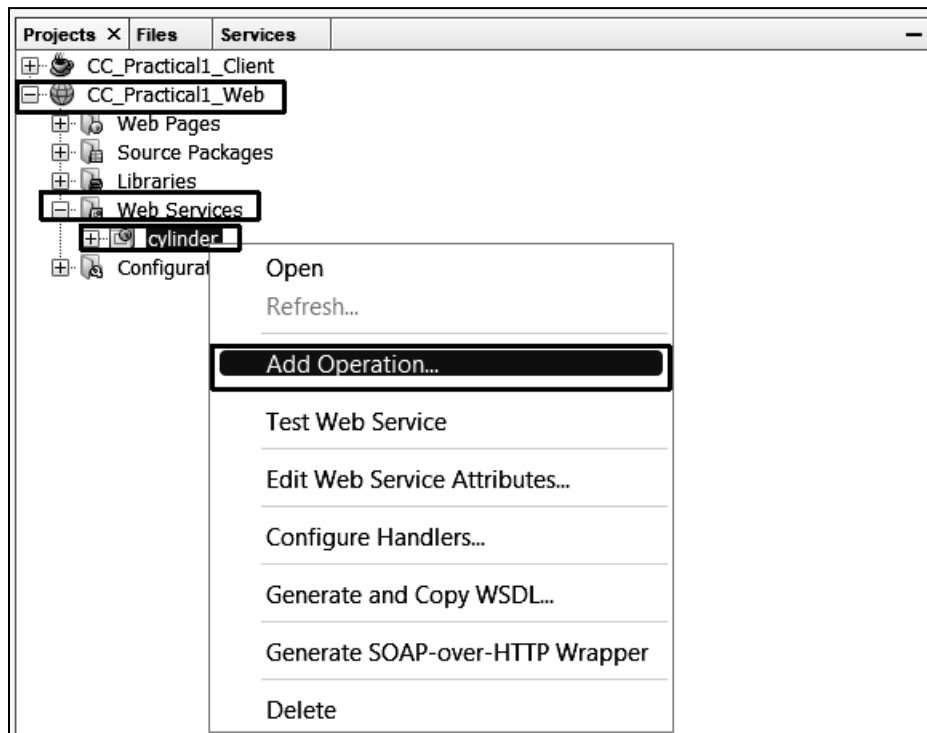


8) Add Web Service Name then click on “Finish”.

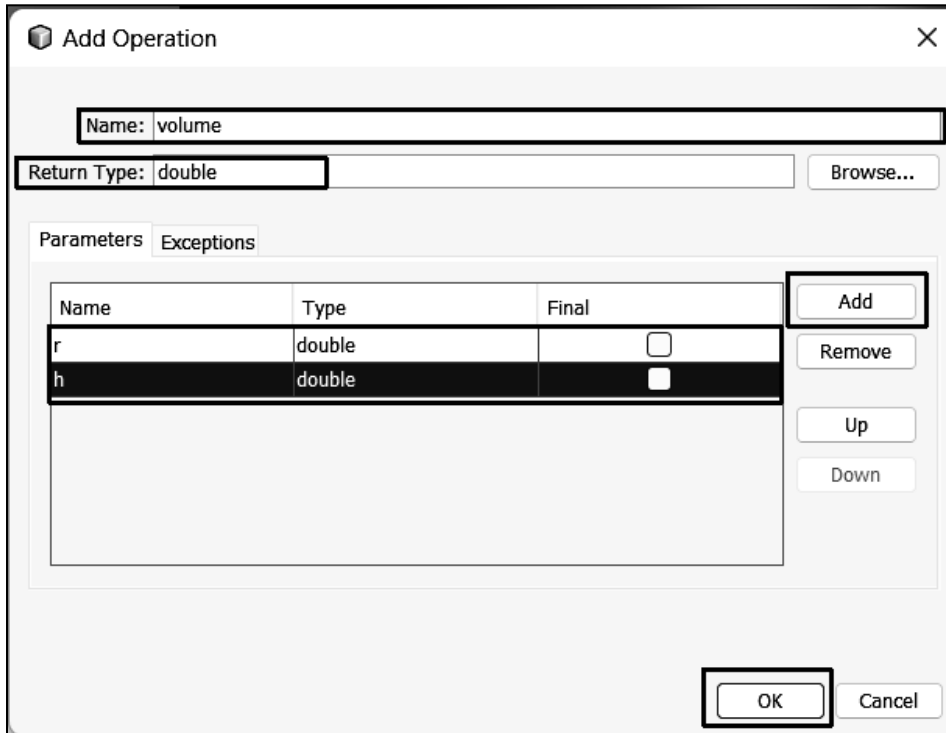


9) Add an operation inside the web service that you have created before.

[your web project name] > Web Services > [your web service name] > Right Click > Add Operation.



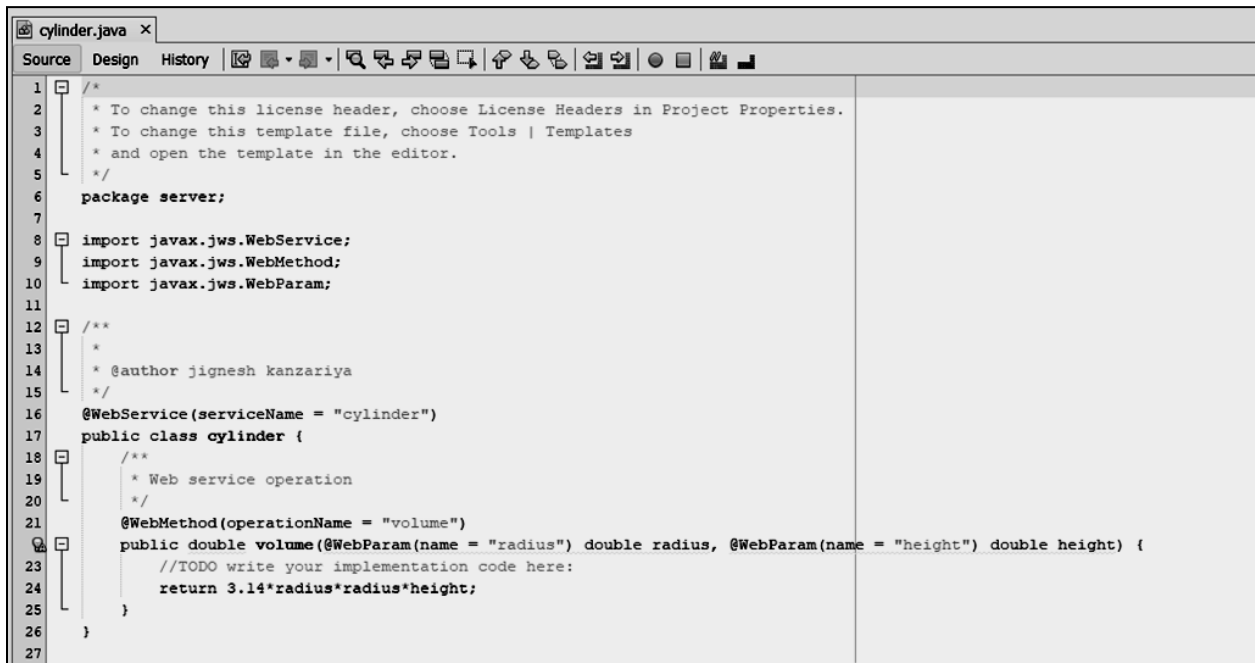
10) Add your Operation.



The "Add Operation" dialog box is shown. It has a title bar with a close button. The "Name" field contains "volume". The "Return Type" field contains "double" and has a "Browse..." button next to it. Below these fields are two tabs: "Parameters" and "Exceptions". The "Parameters" tab is active, showing a table with two columns: "Name" and "Type". The table has two rows: one with "r" and "double", and another with "h" and "double". To the right of the table are buttons: "Add", "Remove", "Up", and "Down". The "Add" button is highlighted with a red box. At the bottom of the dialog are "OK" and "Cancel" buttons, with the "OK" button highlighted by a red box.

Name	Type	Final
r	double	<input type="checkbox"/>
h	double	<input type="checkbox"/>

11) Modify your web service Java file as per your requirements.

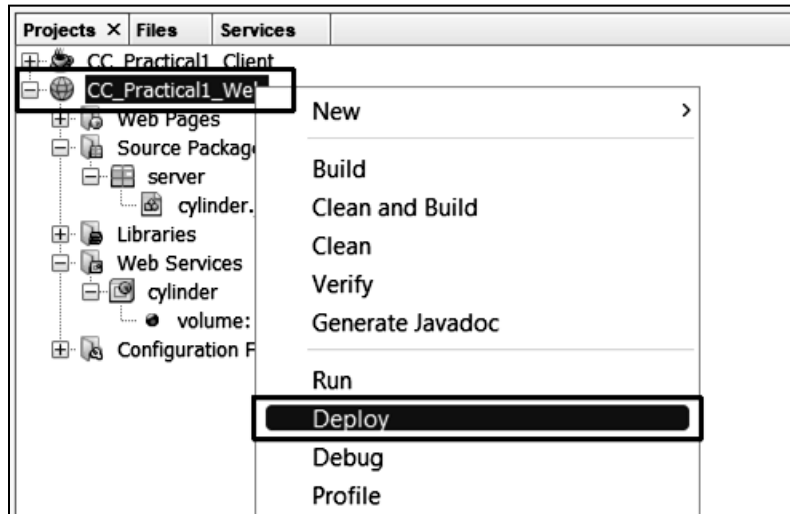


The "cylinder.java" code editor is shown. The code is as follows:

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package server;
7
8   import javax.jws.WebService;
9   import javax.jws.WebMethod;
10  import javax.jws.WebParam;
11
12  /**
13   *
14   * @author jignesh kanzariya
15   */
16  @WebService(serviceName = "cylinder")
17  public class cylinder {
18      /**
19       * Web service operation
20       */
21      @WebMethod(operationName = "volume")
22      public double volume(@WebParam(name = "radius") double radius, @WebParam(name = "height") double height) {
23          //TODO write your implementation code here:
24          return 3.14*radius*radius*height;
25      }
26  }
27
```

12) Deploy your Project.

[your web project name] > Right Click > Deploy.

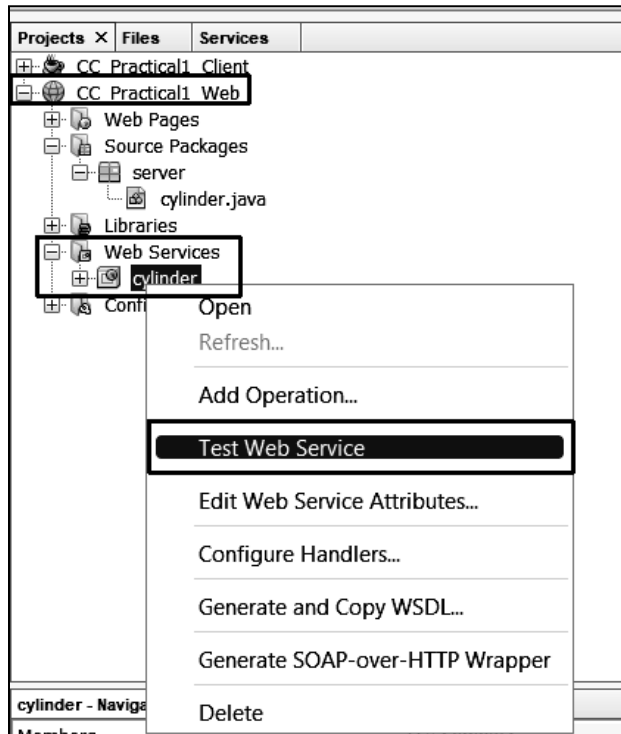


Output.



13) After that Test Web Service.

[your web project name] > Web Services > [your service name] > Right Click > Test Web Service.



14) After clicking "Test Web Service," the browser will open and you will be able to see your operation. Check to see if it is working or not. by applying input and resulting output.

A screenshot of a web browser displaying the 'cylinder Web Service Tester' application. The address bar shows 'localhost:8080/CC_Practical1_Web/cylinder?Tester'. The page title is 'cylinder Web Service Tester'. Below the title, there is a text area with the instruction: 'This form will allow you to test your web service implementation (WSDL File)'. Below this, another text area says: 'To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.' Under the heading 'Methods :', the WSDL method signature is shown: 'public abstract double server.Cylinder.volume(double,double)'. Below the signature, there are two input boxes, each containing the number '2', followed by a closing parenthesis ')'. There is a 'Test' button to the right of the input boxes.

Output:

volume Method invocation

Method parameter(s)

Type	Value
double	2
double	2

Method returned

double : "25.12"

15) Now it's time to create a client after testing your web service on a browser. For that, you need a WSDL file. So, click on the WSDL file link and copy the URL of WSDL file.

cylinder Web Service Tester

This form will allow you to test your web service implementation

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract double server.Cylinder.volume(double,double)

volume (,)

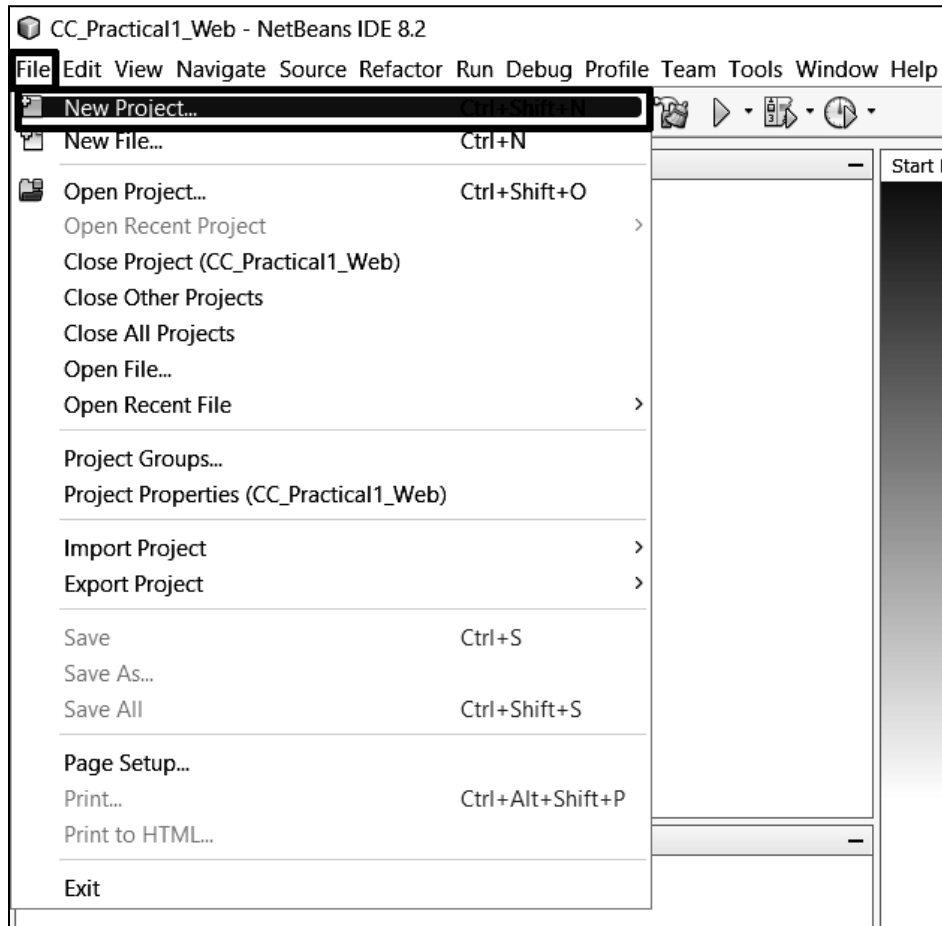
localhost:8080/CC_Practical1_Web/cylinder?WSDL

This XML file does not appear to have any style information associated with it. The document

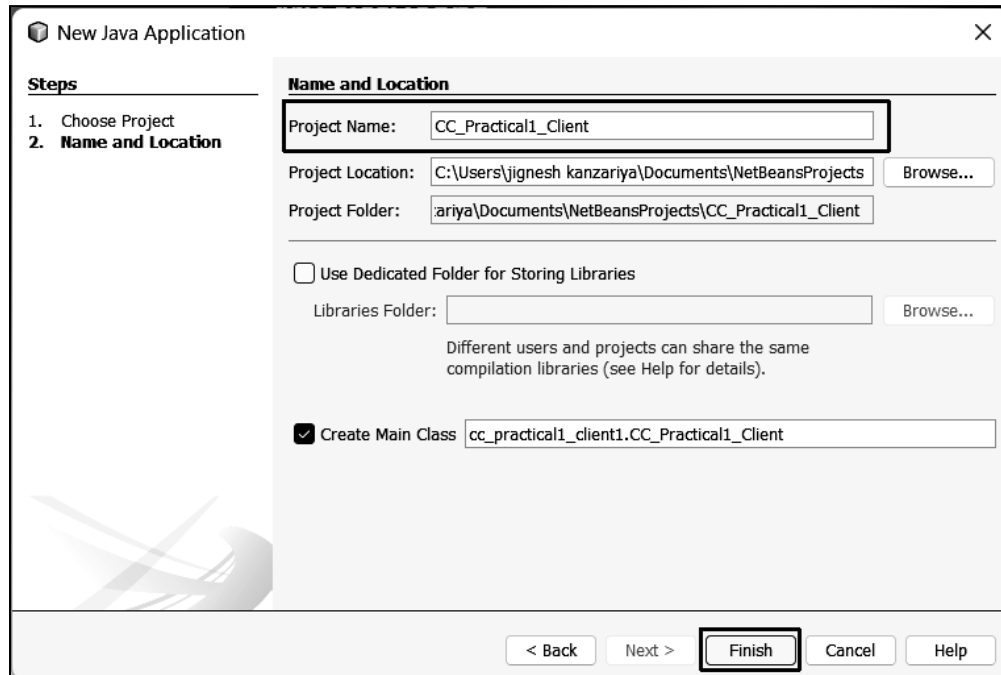
```
<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://sc
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://server/" name="
<types>
```

16) Now create java application it is work as client of our web application.

File > New Project.

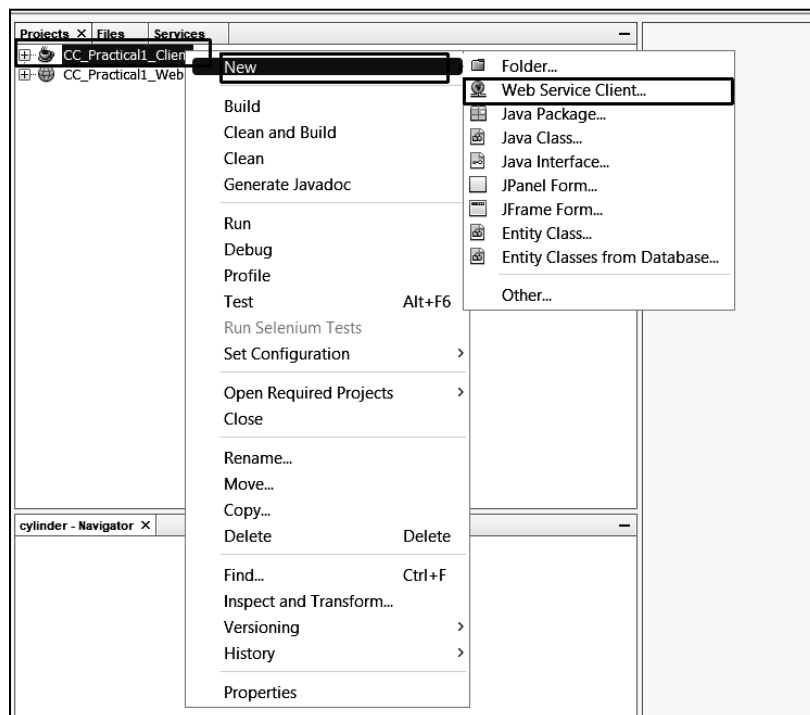


17) Enter your java project name and click “Finish”.

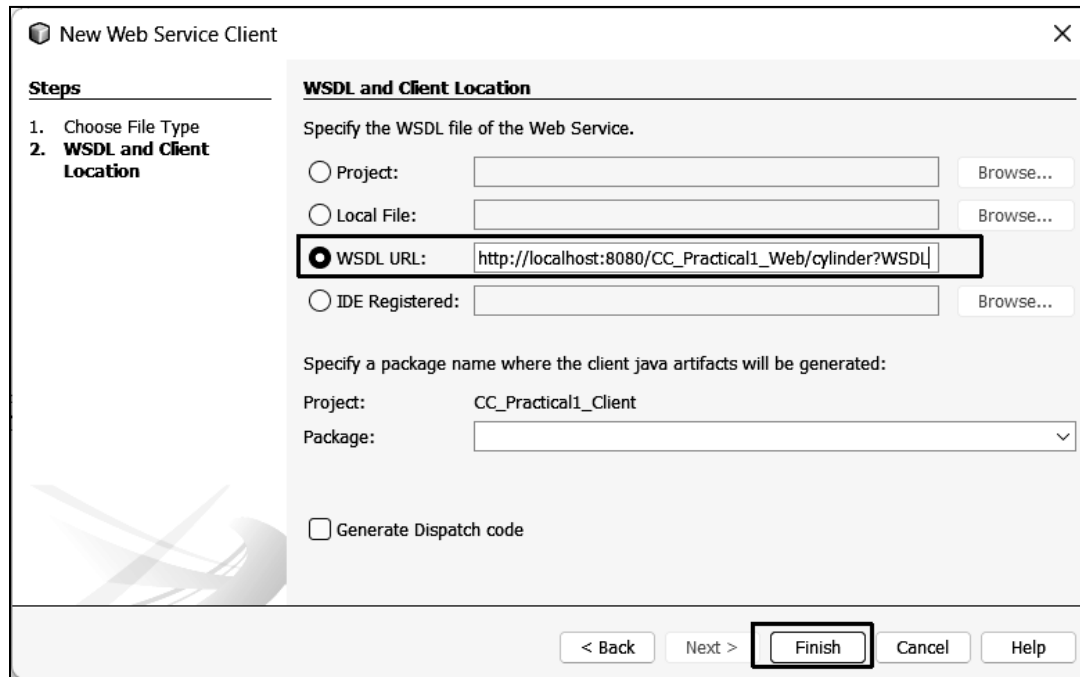


18) Add Web Service Client in your java application project.

[your java project] > Right Click > New > Web Service Client.



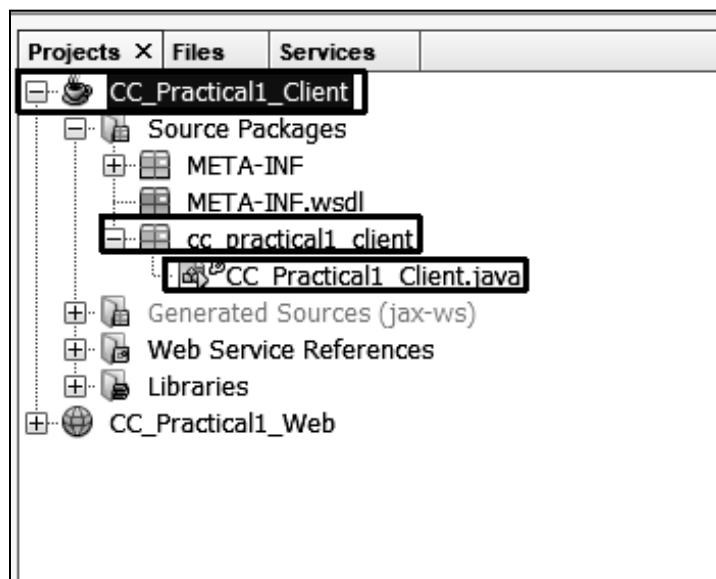
- 19) In the Web Service Client window, select the WSDL URL. Paste a previously copied WSDL URL into an inside WSDL URL field.

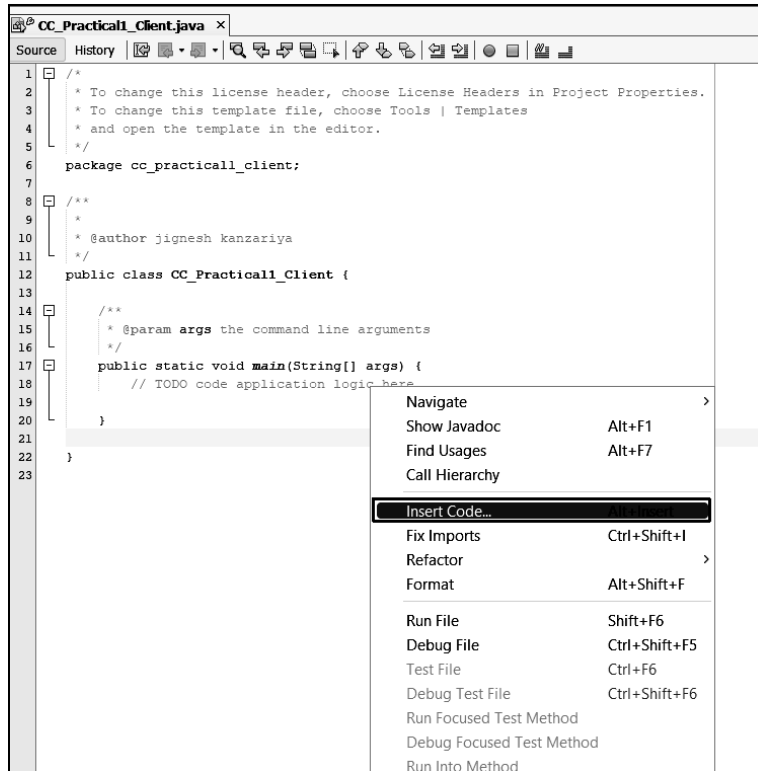


- 20) Insert Code of web service in java application project for that you need to open java file.

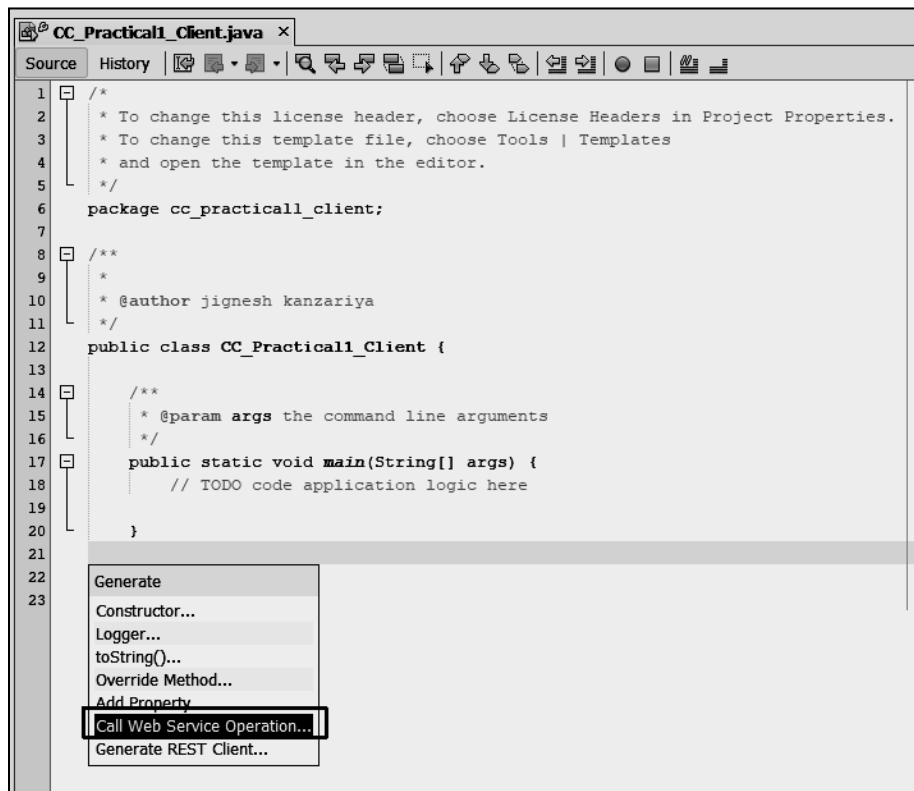
[your java project name] > Source Packages > [project name] > [project name].java.

Inside java file Right Click > Insert Code.

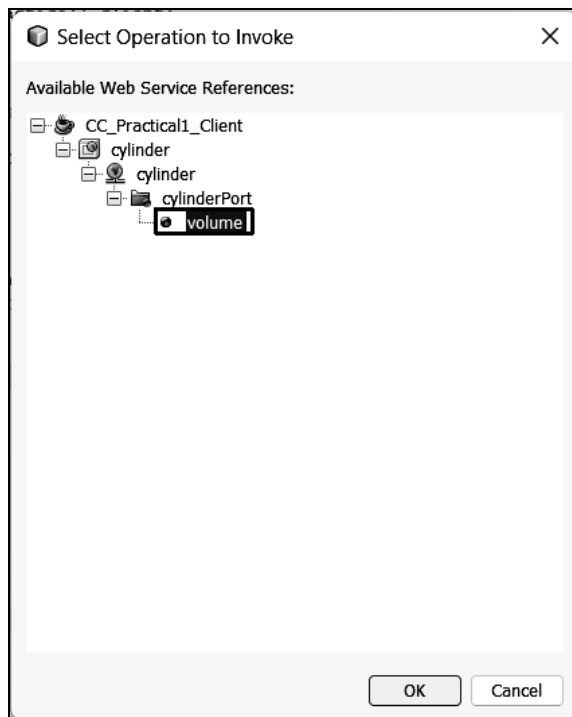




21) After that click “Call Web Service Operation”.



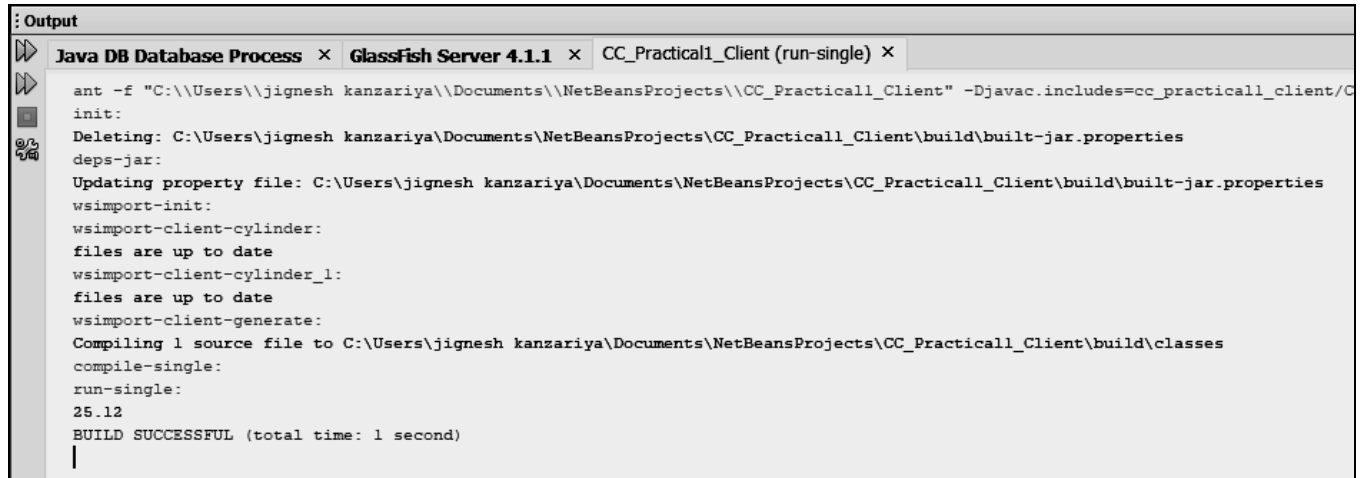
22) Select the operation you want to insert.



23) Modify code as per need.

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package cc_practical1_client;
7
8   /**
9    *
10   * @author jignesh kanzariya
11   */
12   public class CC_Practical1_Client {
13
14       /**
15        * @param args the command line arguments
16        */
17       public static void main(String[] args) {
18           // TODO code application logic here
19           System.out.println(volume(2.0,2.0));
20       }
21
22       private static double volume(double radius, double height) {
23           server.Cylinder_Service service = new server.Cylinder_Service();
24           server.Cylinder port = service.getCylinderPort();
25           return port.volume(radius, height);
26       }
27   }
```

24) Run the file and see output that comes from the server.



The screenshot shows the 'Output' window of an IDE with three tabs: 'Java DB Database Process', 'GlassFish Server 4.1.1', and 'CC_Practical1_Client (run-single)'. The 'CC_Practical1_Client (run-single)' tab is active, displaying the output of an Ant build script. The output shows the execution of various tasks including deleting old build files, updating the build-jar.properties file, and importing client modules. The build concludes with a successful compilation of 1 source file and a total build time of 1 second.

```
ant -f "C:\Users\jignesh.kanzariya\Documents\NetBeansProjects\CC_Practical1_Client" -Djavac.includes=cc_practical1_client/C
init:
Deleting: C:\Users\jignesh.kanzariya\Documents\NetBeansProjects\CC_Practical1_Client\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\jignesh.kanzariya\Documents\NetBeansProjects\CC_Practical1_Client\build\build-jar.properties
wsimport-init:
wsimport-client-cylinder:
files are up to date
wsimport-client-cylinder_1:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\Users\jignesh.kanzariya\Documents\NetBeansProjects\CC_Practical1_Client\build\classes
compile-single:
run-single:
25.12
BUILD SUCCESSFUL (total time: 1 second)
|
```