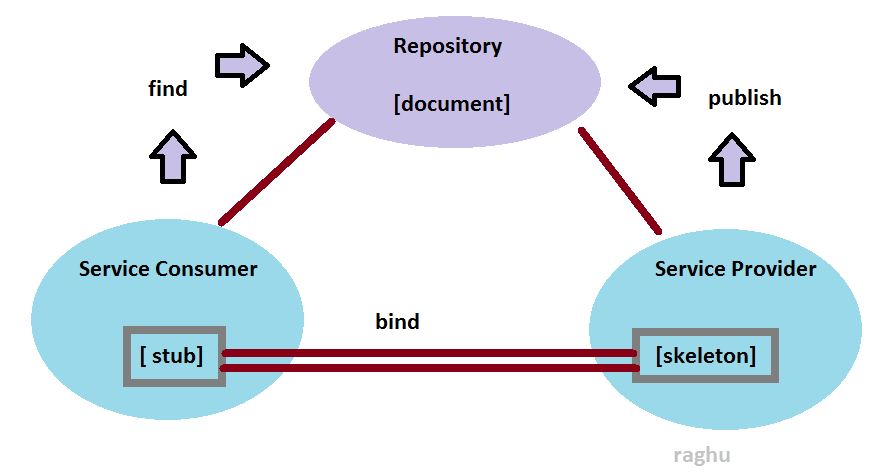
SOAP:- SAOP Stands for Simple Object Access Protocol.

It is a process of sending XML data over HTTP to make communication between two application for integration process.

SOAP Follows SOA (Service Oriented Architecture) Design. (WIKI Def) Service-oriented architecture (SOA) is an architectural pattern in computer software design in which application components provide services to other components via communications, typically over a network.



SOA contains 3 components and 3 operations.

1. Components:
2. Service provider
3. Repository
4. Service consumer
5. Operations:
6. Publish : Generate Document from Skeleton
7. Find :- Check for WSDL and generate classes from WSDL
8. Bind: - Make Communication link between Consumer and Provider.

Initially we need to design Skeleton which is a part of service provider, it provides service for consumer. Once service provider component is defined then using publish operation skeleton need to be converted to document format. As per SOA document should be a readable text format.

As per SOAP document is WSDL File (Web Service Description Language).

This WSDL/Document file is placed under Repository. Repository also known as UDDI (Universal Description, Discovery, and Integration). WSDL/Document mainly provides details about skeleton. Like class name/service name, list of operations and their parameters and return type details.

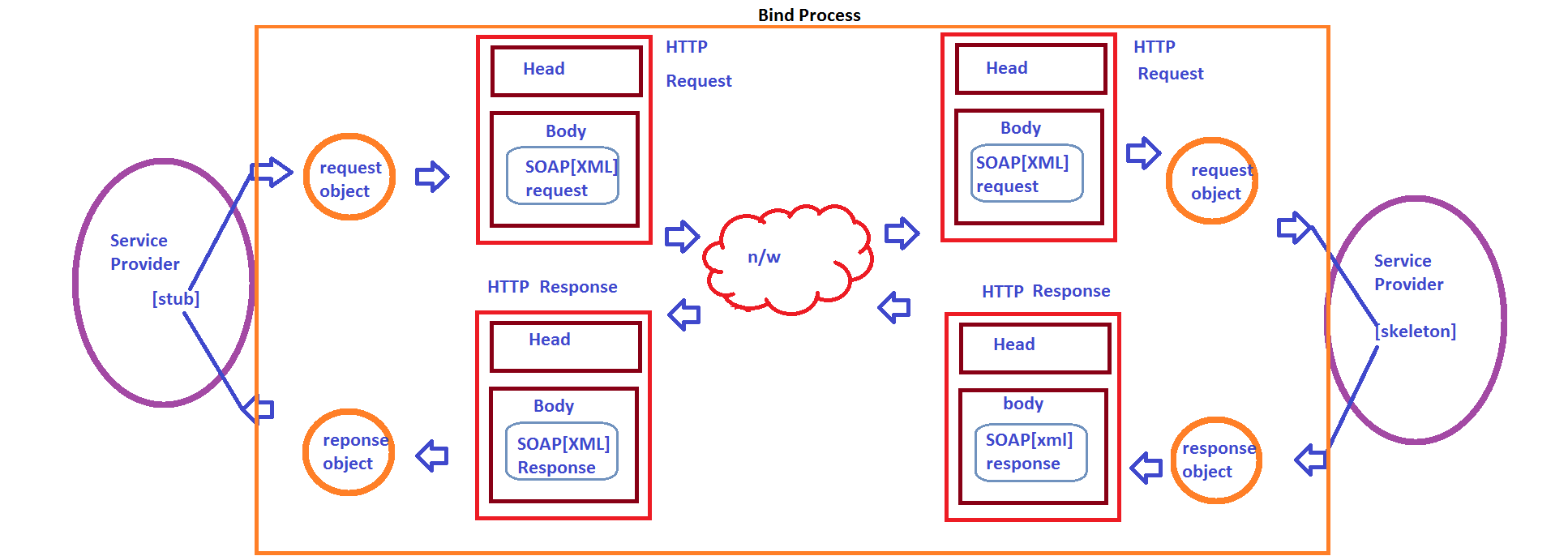
On publishing Document, service consumer can make a request for document, and generate classes

This operation is known as find. In case of java wsimport tool is used to generate stubs from WSDL file.

After generating stubs we need to define bind programming also known as connection/client program. This should be defined at service consumer side. It is used to make request from consumer to provider.

Writing of client Program should follow process of WSDL service, operations and messages.

**SOAP Binding Process in SOA:**

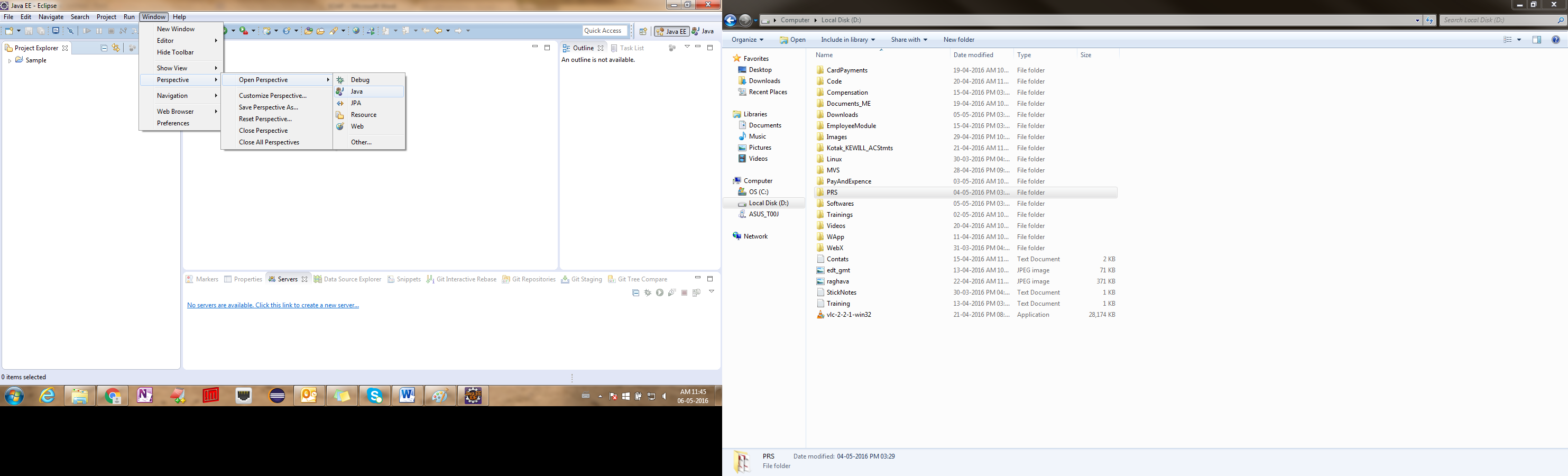


**On running Client Program, It generates a request object, which is converted into XML format of SOAP Request Format. SOAP Request is a XML file, it cannot be transferred over network as it is, so transfer over network we use HTTP (Transfer Protocol), here XML request will be wrapped into HTTP Body. And this will be sent over network. Service Provider Receives this request and parse the SOAP request and converts into its object format.**

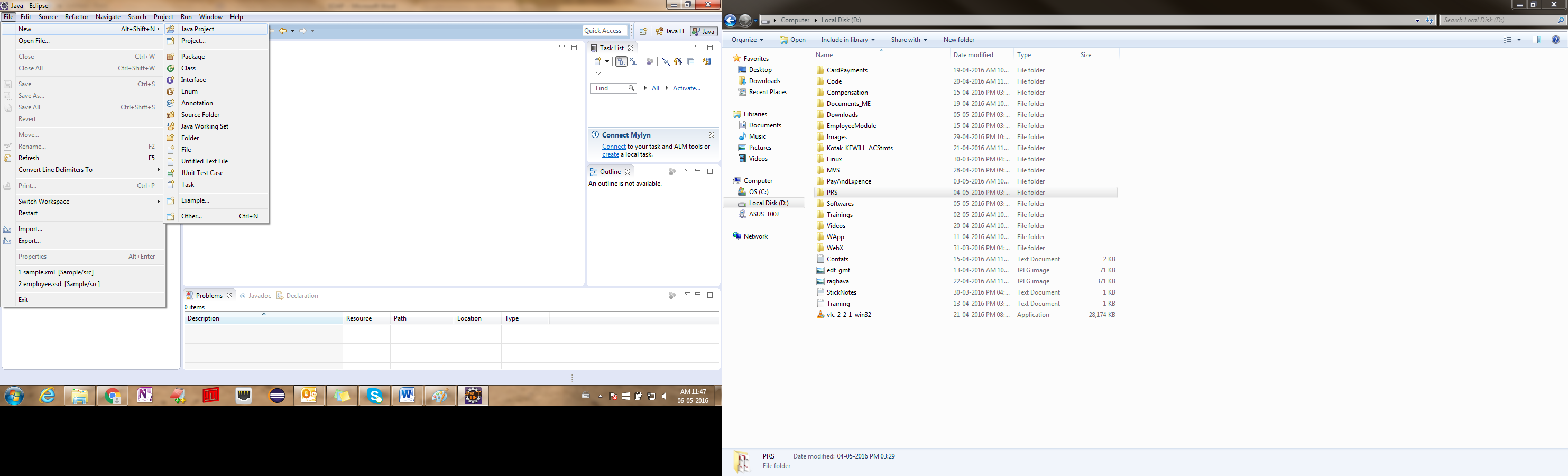
**Using input request object service is executed and returns Response Object. This time service provider converts response Object (SOAP Response Object) to XML format and wraps into HTTP Response and Service consumer receives this and reads SOAP Response XML into Object format for further operations.**

**Writing SOAP Program Using Eclipse:**

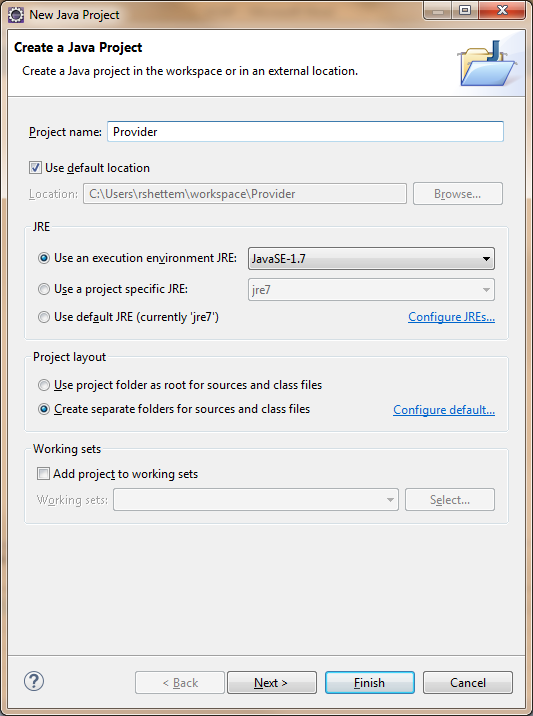
**\*\*) Open eclipse in Java Perspective. (If Java Option, unable to find click on other and select Java)**



**\*\*) create a new Java Project for Service Provider:**

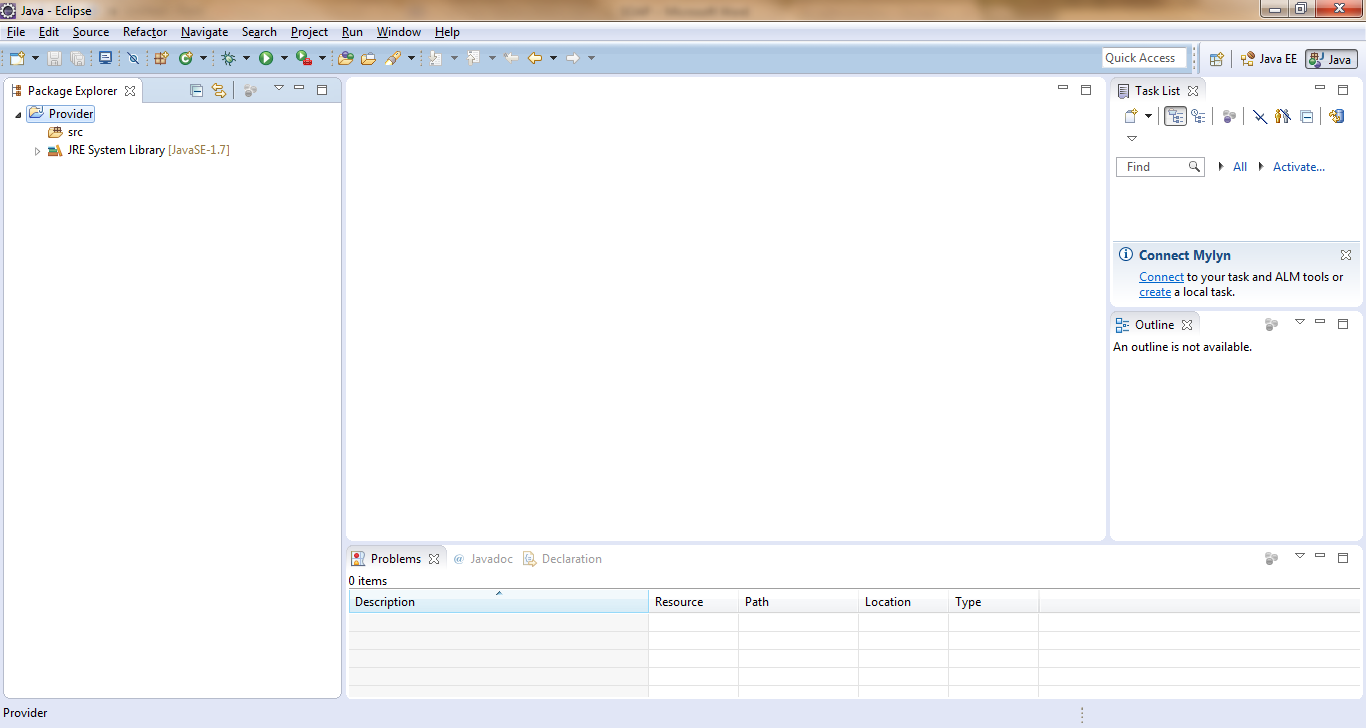


**Enter Project name. Ex: Provider**



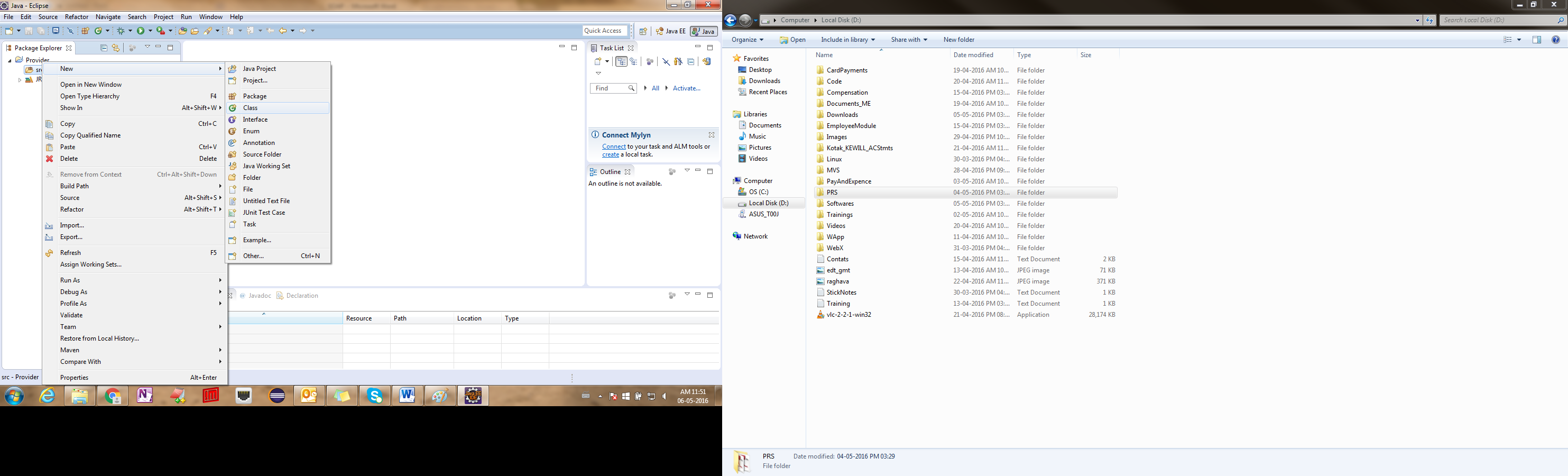
**And click on Finish Button.**

**Observe Project created in Package/Project Explorer.**

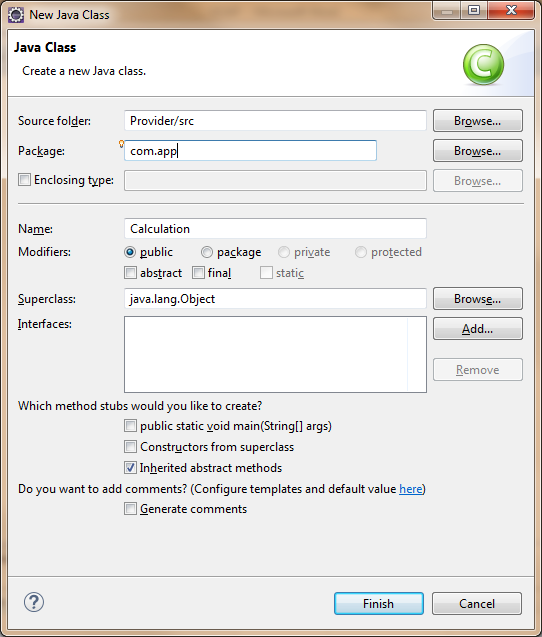


**\*\*) Create a new Java Class for Skeleton code.**

**Right click on sac -> new -> class**



**Enter class name and package name:**



**And click on finish button.**

**Define a method with sample logic and also apply @WebService and @WebMethod annotations at class and method level to Define Complete skeleton (Provider Code).**

**Ex:**

**package** com.app;

**import** javax.jws.WebMethod;

**import** javax.jws.WebService;

@WebService

**public** **class** Calculation {

@WebMethod

**public** **int** doSum(**int** x,**int** y){

**return** x+y;

}

}

**\*\*) Publish above code by creating publisher class.**

**(Create one more class as shown above.)**

**Ex:**

**package** com.app;

**import** javax.xml.ws.Endpoint;

**public** **class** Publisher {

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

//where to publish

String address="http://localhost:9999/sample";

//what to publish

Calculation calOb=**new** Calculation();

Endpoint.*publish*(address, calOb);

System.***out***.println("done");

}

}

**Here Endpoint is abstract class which is having static method publish. It is used to generate WSDL from Skeleton Object. Publish operation needs two parameter 1st one address of WSDL file to be published.**

**And 2nd one is Object of Skeleton.**

**Address should contain format of**

[**http://IPAddress:portNumber/context**](http://IPAddress:portNumber/context)**..**

**Ex:**

[**http://201.88.0.08:9009/weatherReportService**](http://201.88.0.08:9009/weatherReportService)

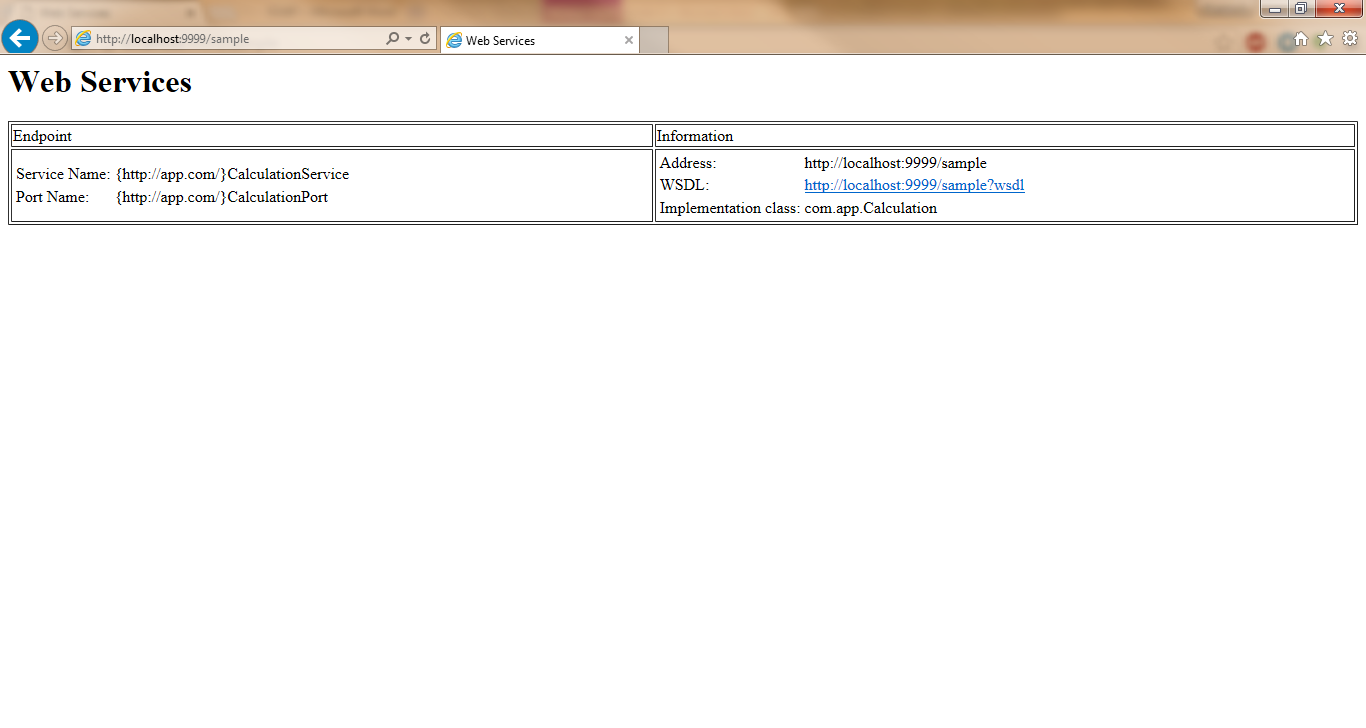
**After Executing publisher class. It generates WSDL file. To view generated WSDL file**

**Open browser and type address?WSDL.**

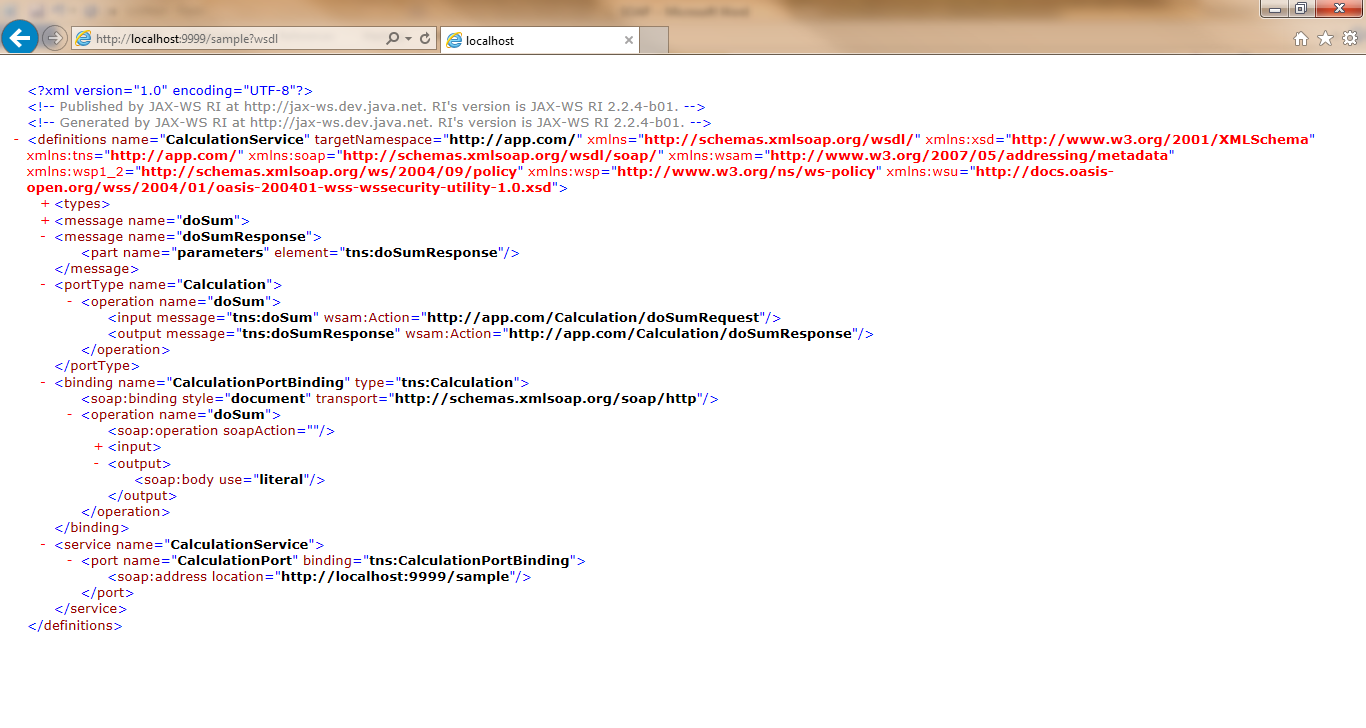
**Ex:** <http://localhost:9999/sample?WSDL>

If you type only address without ?WSDL , you can observer some extra details like service name, Port name

**WSDL Link Details etc…**

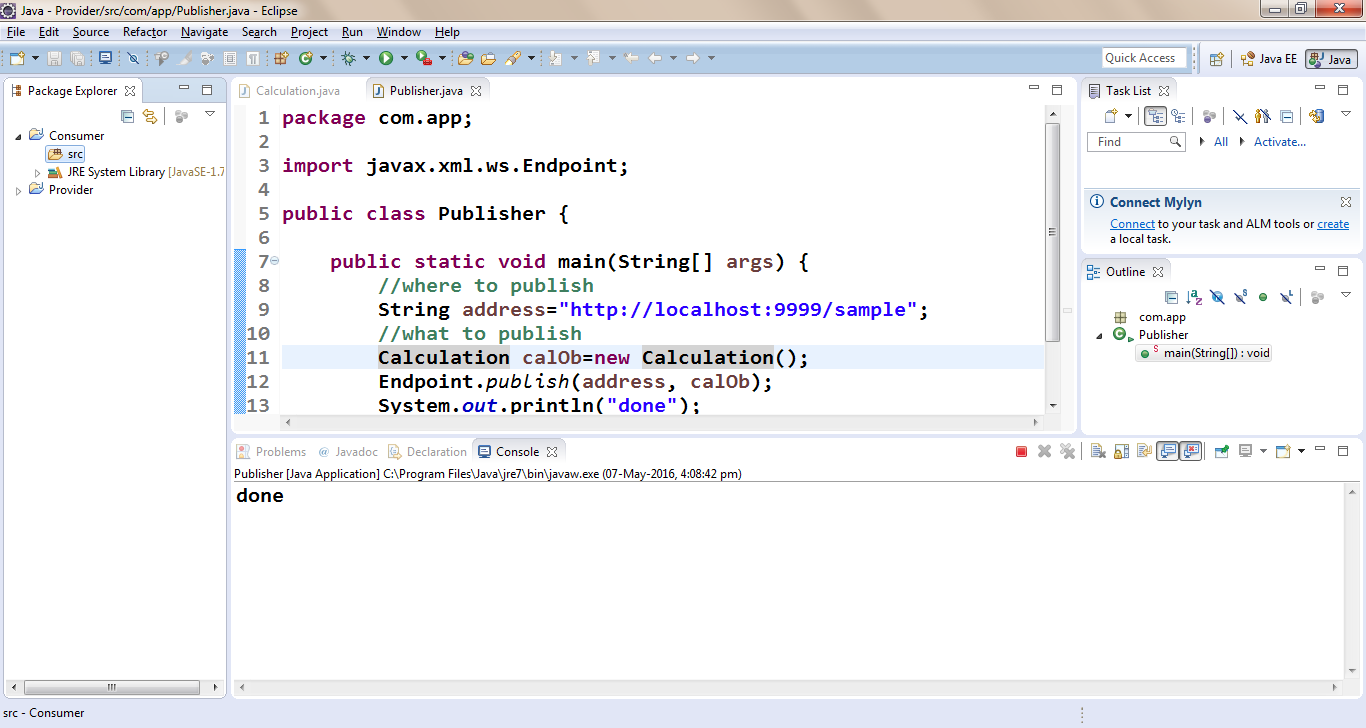


**Or WSDL :**

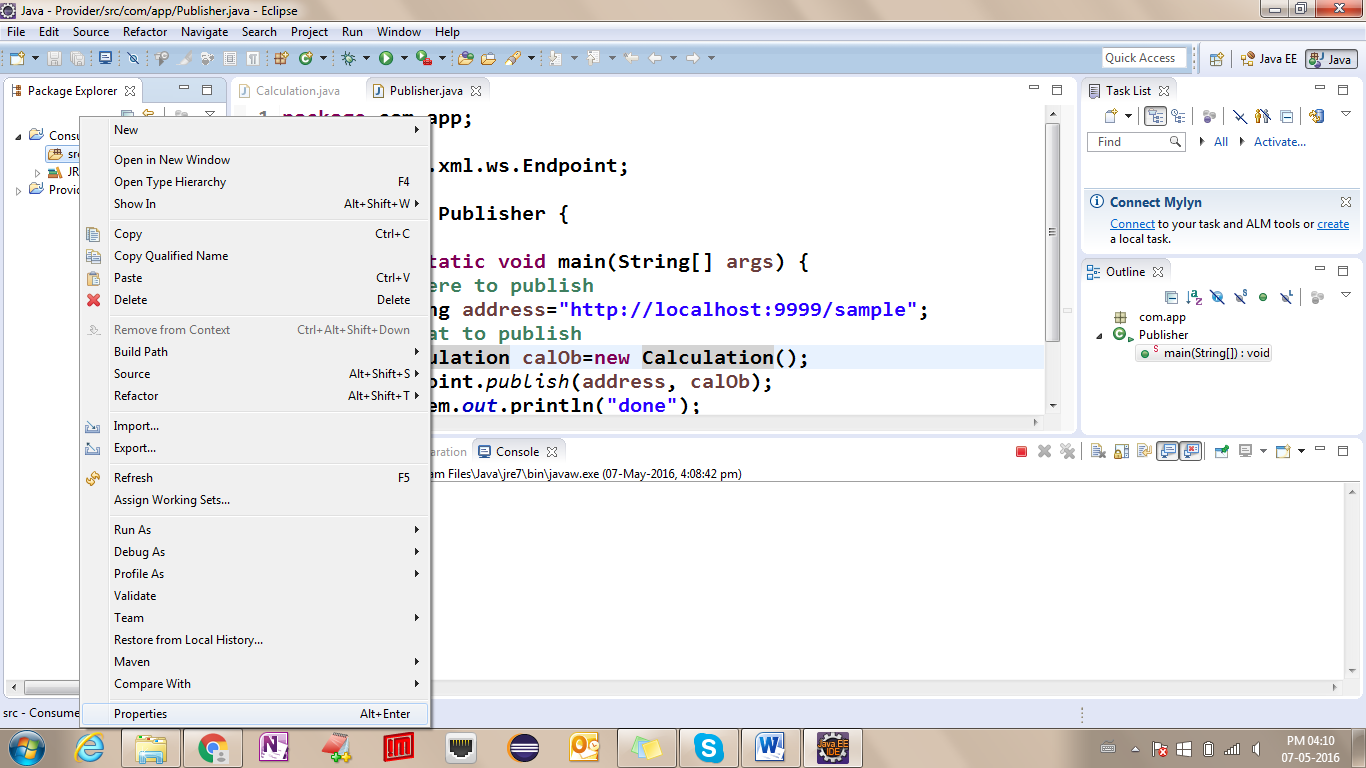


**Create another project for service consumer:**

**File->new->java Project -> Enter Name as Consumer -> finish button.**



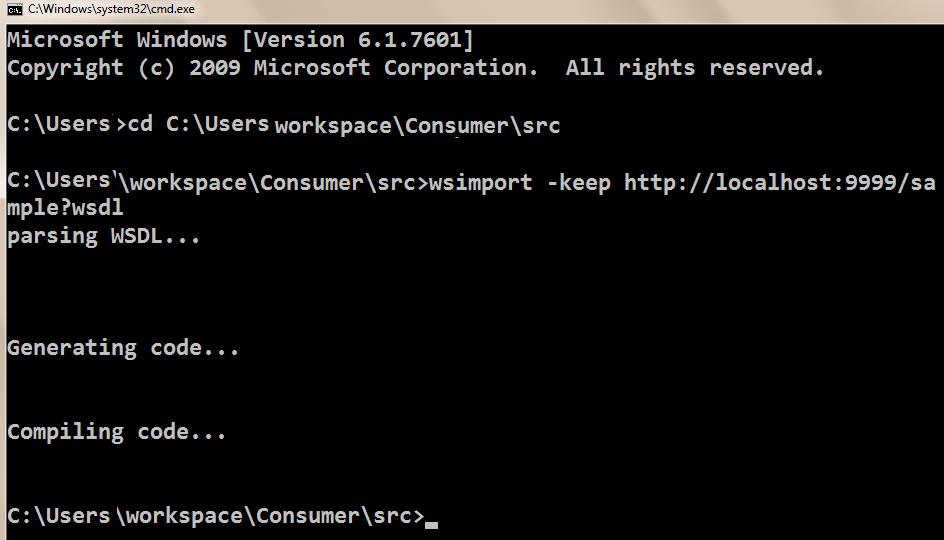
**Right click on src-> properties -> copy location: (Atl+Enter is shortcut)**



**Copy location:**

**And open cmd prompt for same location as shown, also generates stubs using wsimport**

**(ex: wsimport –keep http://localhost:9999/sample?WSDL)**

****

**Define Client class or bind programs as below steps:**

1. **Create Object for service class (find service name under WSDL file**

**As <service name=”SampleService”>**

1. **Using Service Object, get Port Type Object. Example (serv.get[PortName]).**

**Observe <port name=”SamplePort”**

1. **Once You get PortType Object call operation by providing inputs.**

**Example Program for Bind:**

**package** com.app;

**public** **class** BindProgram {

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

//step 1: create Object for service

CalculationService cs=**new** CalculationService();

//step 2: create object for PortType. use get[PortName]() method

Calculation c=cs.getCalculationPort();

//step 3: call operation, pass parameters and get Return Type

**int** res=c.doSum(10, 200);

//display result

System.***out***.println("Sum result from service is:"+res);

}

}

Result:

Sum result from service is:210

**Final Package Explorer:**

