Java Web Services

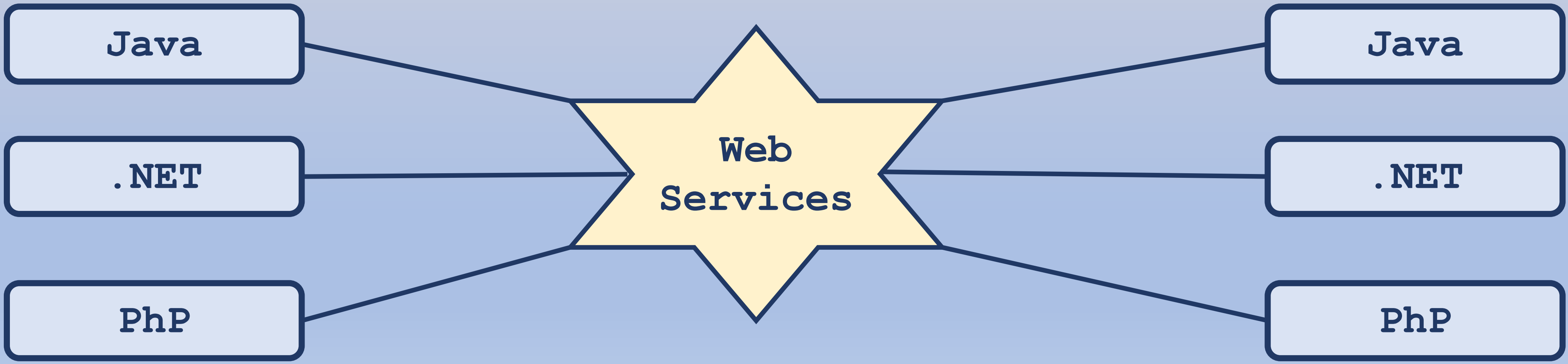
Unit - 6

Topics to be covered

- Introduction to web-services
- Web service architecture
- Functions of web services
- Web services Protocol stack
- Components of web services
  - SOAP
  - UDDI
  - WSDL
- Implement Hello World SOAP web service using eclipse.

Java Web services

- Web services are client and server applications that communicate over the World Wide Web's (WWW) through HyperText Transfer Protocol (HTTP).
- Java Web Services is a technology that allows applications to communicate over a network regardless of their programming languages/platform.



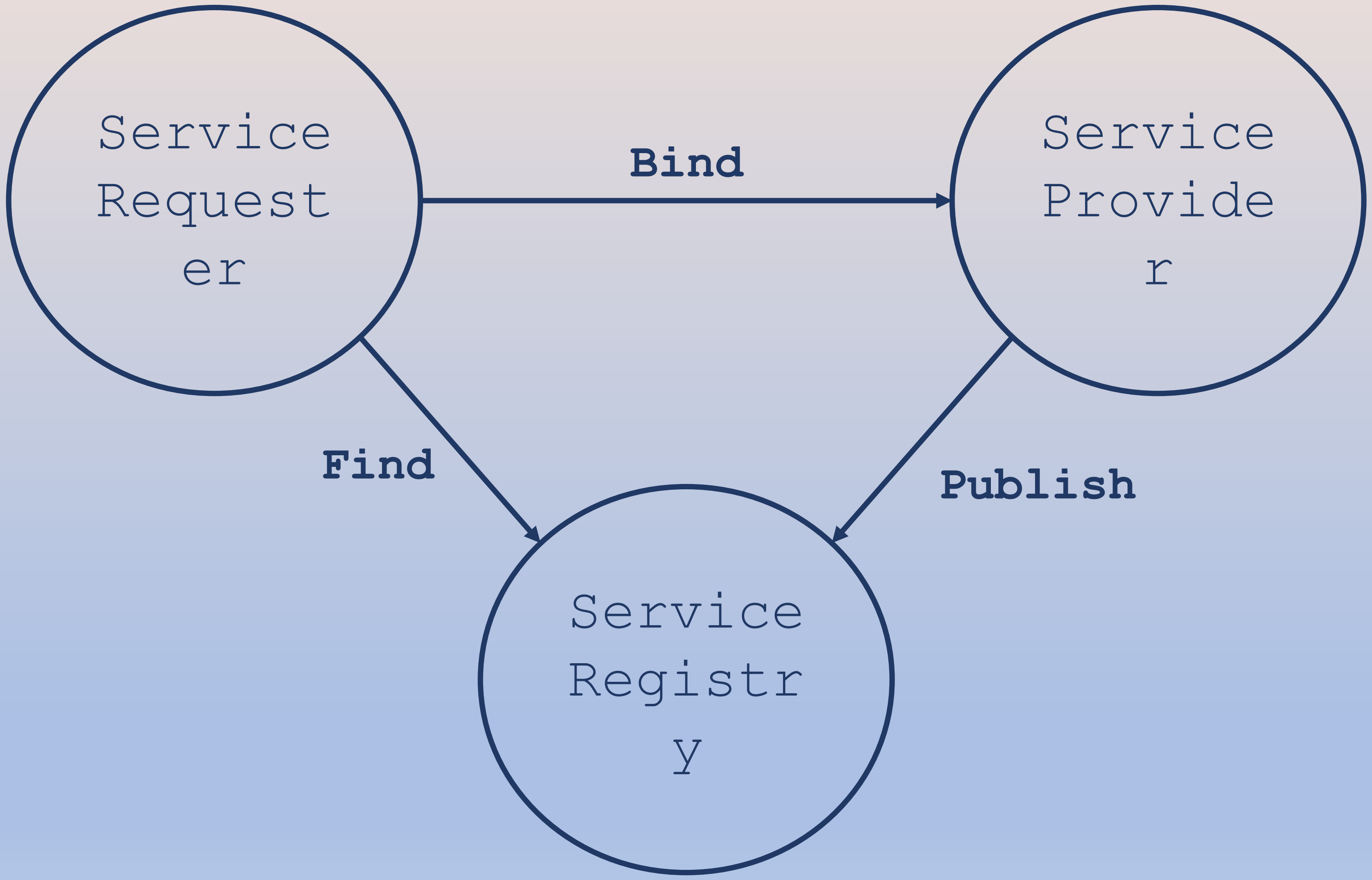
Web Service Architecture

- The architecture of web service interacts among three **components:**
  - **Service Provider:** It is the platform that hosts the services.
  - **Service Requester:** it’s the client system application that consumes the web services.
  - **Service Registry:** It is the repository where web service providers can publish their service description such as information about service, its endpoints and other metadata. Clients query the registry to locate the appropriate services.
- There are various operations that are performed in this architecture
  - Publish
  - Find

Operations on Web Service Architecture

- **Publish:** In the publish operation, a service description must be published so that a service requester can find the service.
- **Find:** In the find operation, the service requestor retrieves the service description directly. It can be involved in two different lifecycle phases for the service requestor:
  - **At design time** to retrieve the service's interface description for program development.
  - **At the runtime** to retrieve the service's binding and location description for invocation.
- **Bind:** In the bind operation, the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service

Web Service Components & Their Interaction

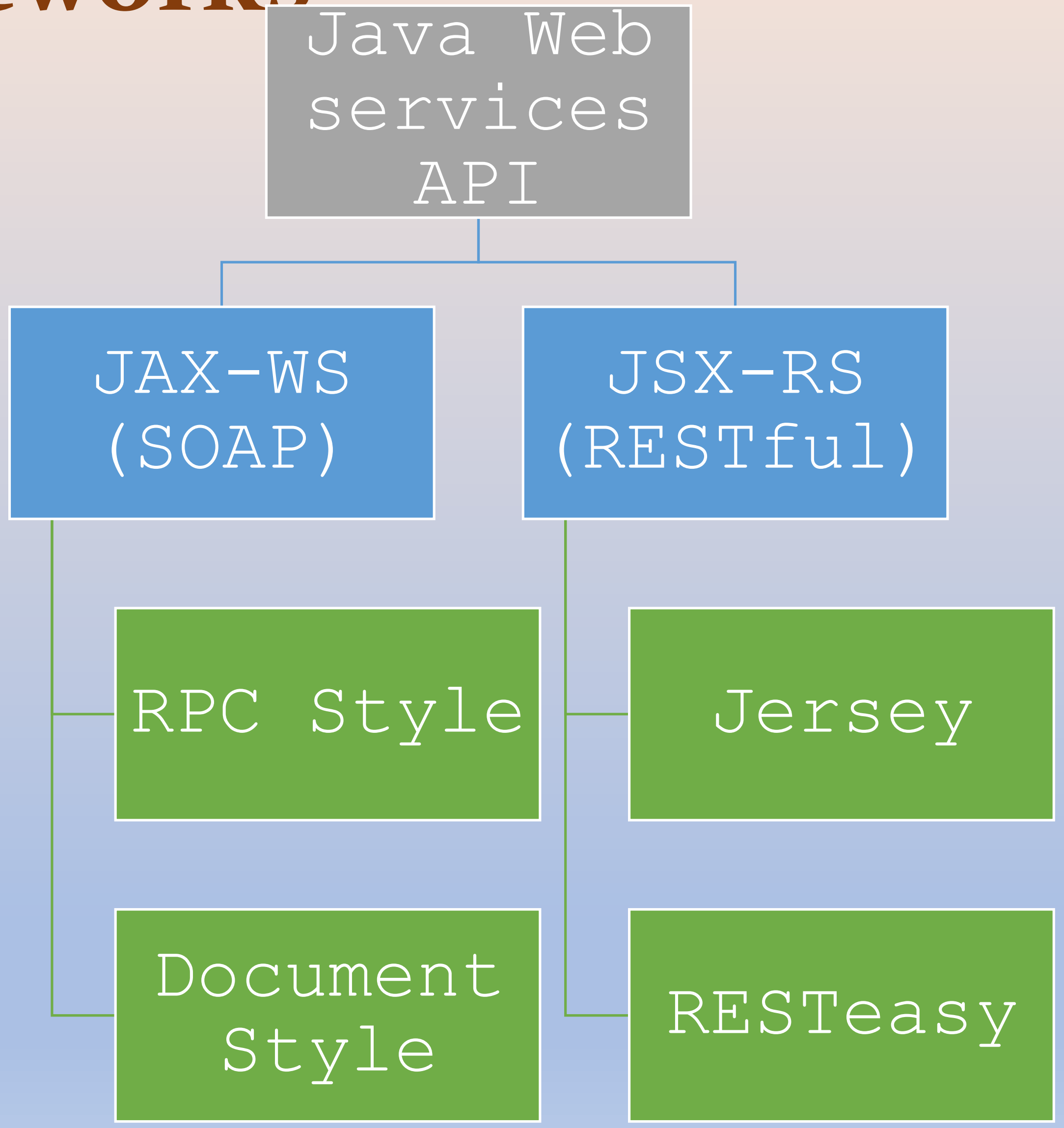


Functions of Web services

Interoperability	• Webservices facilitate communication of different applications regardless of their programming language and platform.
Communication	• It allows a way for applications to send request and receive response over network.
Service Discovery	• It allows applications to find and locate available services through registries and directories.
Service description	• Web services use descriptive language like WSDL (Web service descriptive language) to define the structure and functionality of a service.
Service Invocation	• Web services let the applications invoke remote services and access their functionality.
Data Exchange	• Applications can share and exchange data.
Security	• Web services incorporate security mechanisms to protect data during transmission and storage.
Scalability	• Can handle large number of requests.
Loose coupling	• Applications are independent to be executing separately.
Reusability	• Web services encourage the reuse of existing components and functionality to improve productivity.

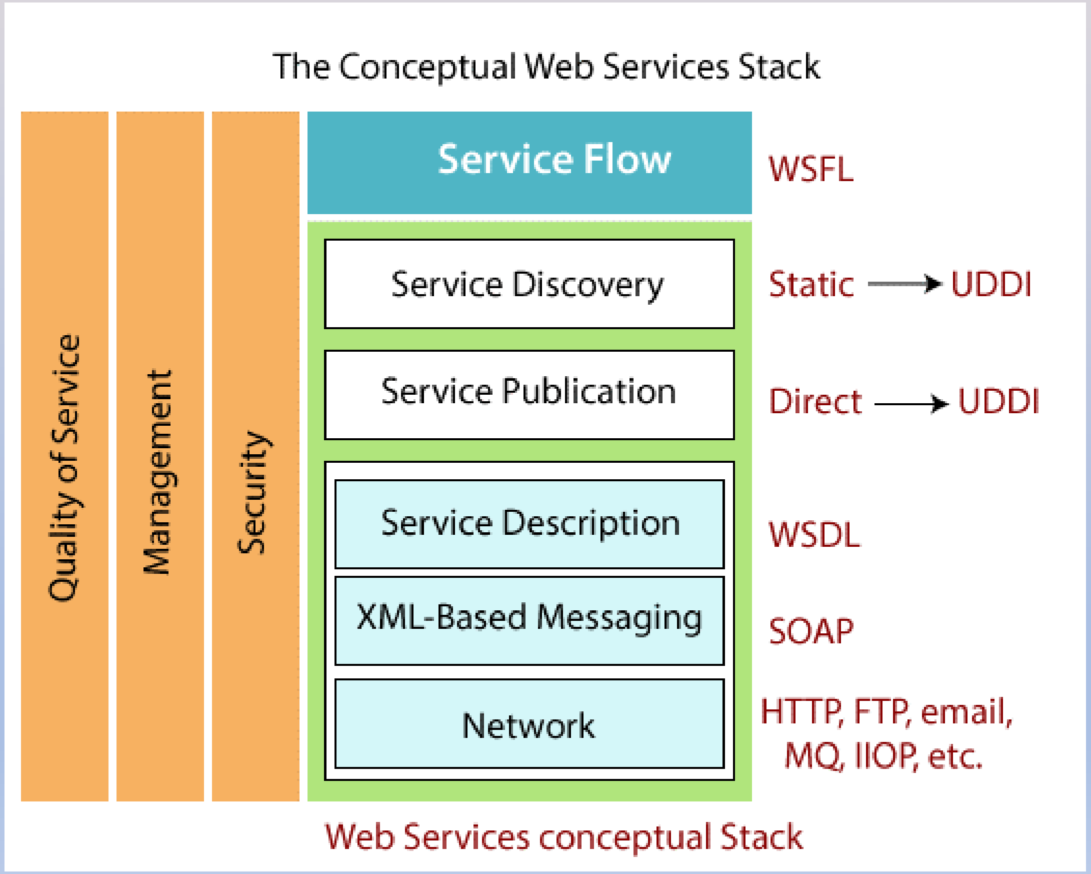
Java Web Services Frameworks

- Spring Boot
- Jakarta RESTful Web Services
- **Java SOAP**
- JAX-WS



Web Services Protocol Stack

- To perform the three operations: publish, find, and bind in an interoperable manner, we need a web service stack.



Web Services conceptual Stack

Protocols in Stack

- A web service protocol stack typically contains four protocols:
  - **Transport Protocol**
    - its responsible for the actual transportation of messages between the client and the service provider.
    - Http, FTP, SMTP are few protocols among those used for actual transportation.
  - **Messaging Protocol**
    - This protocol is responsible for encoding the messages in a common XML format so that they are understood at either end of a network connection. SOAP is the XML messaging protocol because it supports all the three operations: publish, find, and bind operation.
  - **Description Protocol**
    - This protocol is used for describing the public interface to a specific web service. WSDL is the standard used for the XML-based service description. WSDL describes the interface and the mechanics of service interaction. The description is essential to specify the business context, quality of service, and service-to-service relationship.
  - **Discovery Protocol**
    - This protocol is a centralized service in a common registry so that network Web services can publish their location as well as the description, and it becomes easy to find those services that are available on the network.

Protocols in Stack

- The first three layers in the stack are needed to provide or use any web service. The simplest stack consists of HTTP for the network layer, SOAP protocol for the XML-based messaging, and WSDL for the service description layer. This three-layer provides interoperability and enables the web service to control the existing internet infrastructure. It also creates a low cost of entry to a global environment.
- The bottom three layers of the stack identify technologies for compliance and interoperability; the next two layers – Service Publication and Service Discovery are implemented with a range of solutions.

SOAP: Simple Object Access Protocol

- SOAP **defines standardized format in XML for communication in web services over HTTP or other transport protocols.**
- SOAP provides **mechanism for exchanging messages** containing method calls, parameters and responses.
- Interaction
  - The **web service client sends a SOAP request** message encoded in **XML format** to the web service provider, containing the method call and any required parameters over a communication protocol i.e: HTTP.
  - The **web service provider receives** the SOAP request, processes it and **prepares a SOAP response** message.
  - This **response is sent to the client** and extracted to get the desired response.

UDDI: Universal Description, Discovery an Integration

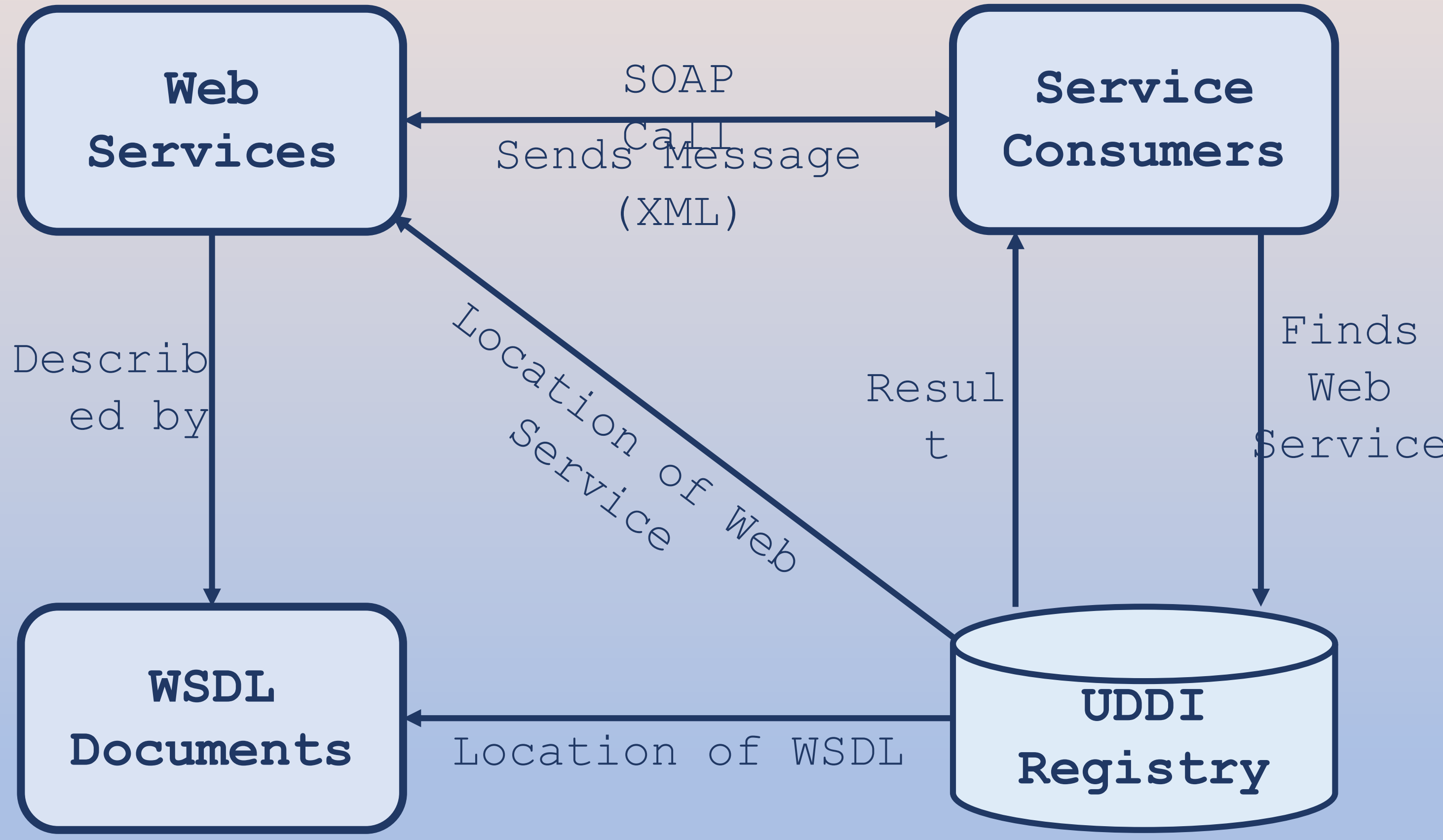
- It is a directory service that **allows web service providers to publish their service descriptions and locate these services.**
- It provides a **standardized way to describe, discover and integrate** web services within the network.
- Interaction:
  - Web service providers **register their service description** in the UDDI registory.
  - Clients **query the registry to search** for specific services based on criteria such as service type, keyword, location.
  - The UDDI registry **returns relevant service description** to the client such as service's interface, location and supported protocols.
  - The **client uses this obtained description to interact** with the desired web service provider.

WSDL: Web Services Description Language

- WSDL is an **XML-based language used to describe the interface of a web service.**
- It specifies the **operations that the service provides**, the **parameters** that can be passed to each operation, and the **data types** used by the service.
- Interaction:
  - The web service provider **creates a WSDL document** and **made available** to the client, which is parsed to obtain information about service's operations, input/output parameters and message formats
  - Based on this information the **client constructs a SOAP message to send requests and receive responses** from the web service provider.

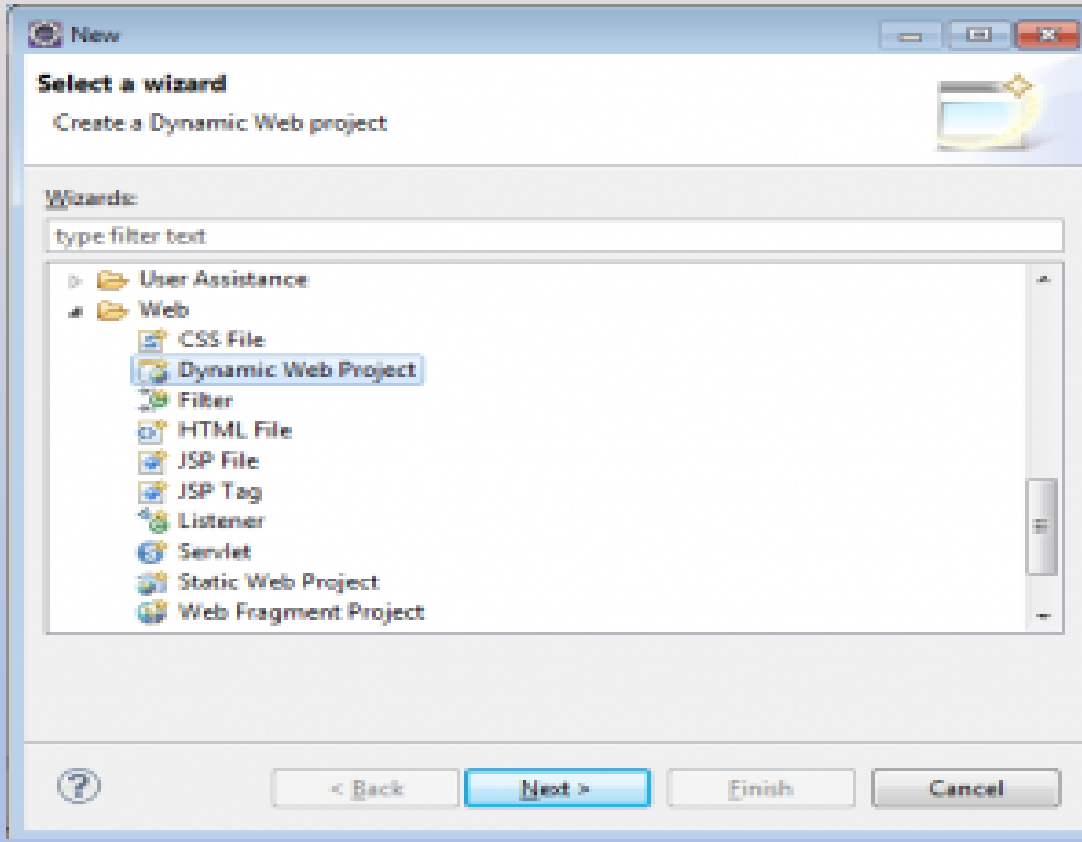
Interaction of Web Service Components

- The consumer contacts the UDDI registry and finds the service description (in WSDL)
- Using WSDL document consumer finds the location of the service and "binds" to it.
- Then, using the interface from WSDL document, consumer



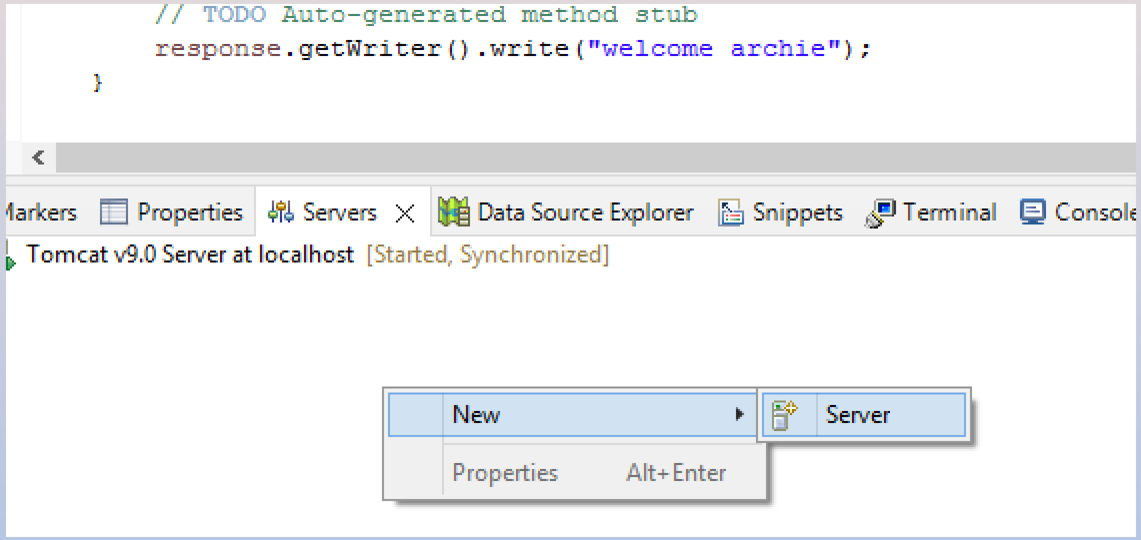
Setting up Eclipse

- Install Eclipse:
  - Download the eclipse version appropriate for your JDK version.
  - Also consider your tomcat server while selecting eclipse version.
  - Example: JDK 17 with tomcat 9 is supported by eclipse 4.26
- Creating a web service:
  - Create a **Dynamic Web Project**
    - Use the **new project** from menu and **open project wizard. Select 'Dynamic Web Project'** and **click next.** Then **give a project name** and **select a target runtime** (Server



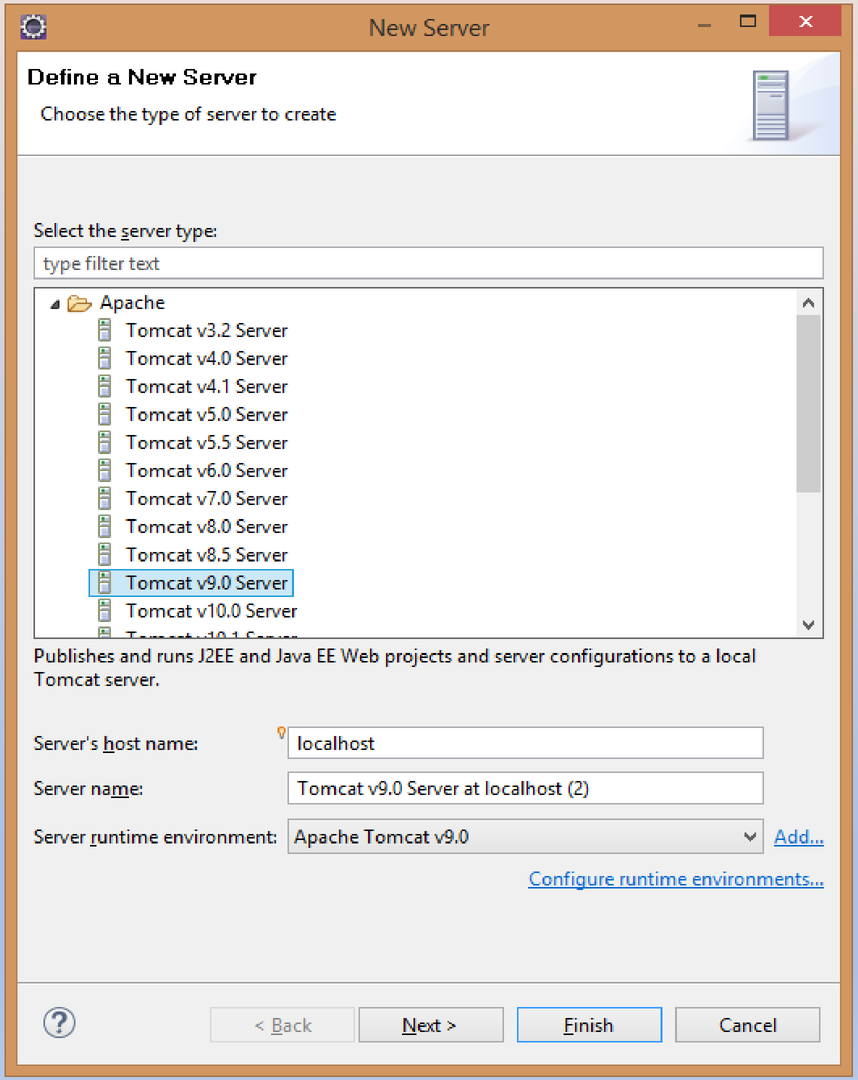
Setting up Eclipse

- Create a new Runtime



```
        // TODO Auto-generated method stub
        response.getWriter().write("welcome archie");
    }
```

Tomcat v9.0 Server at localhost [Started, Synchronized]



New Server

Define a New Server

Choose the type of server to create

Select the server type:

type filter text

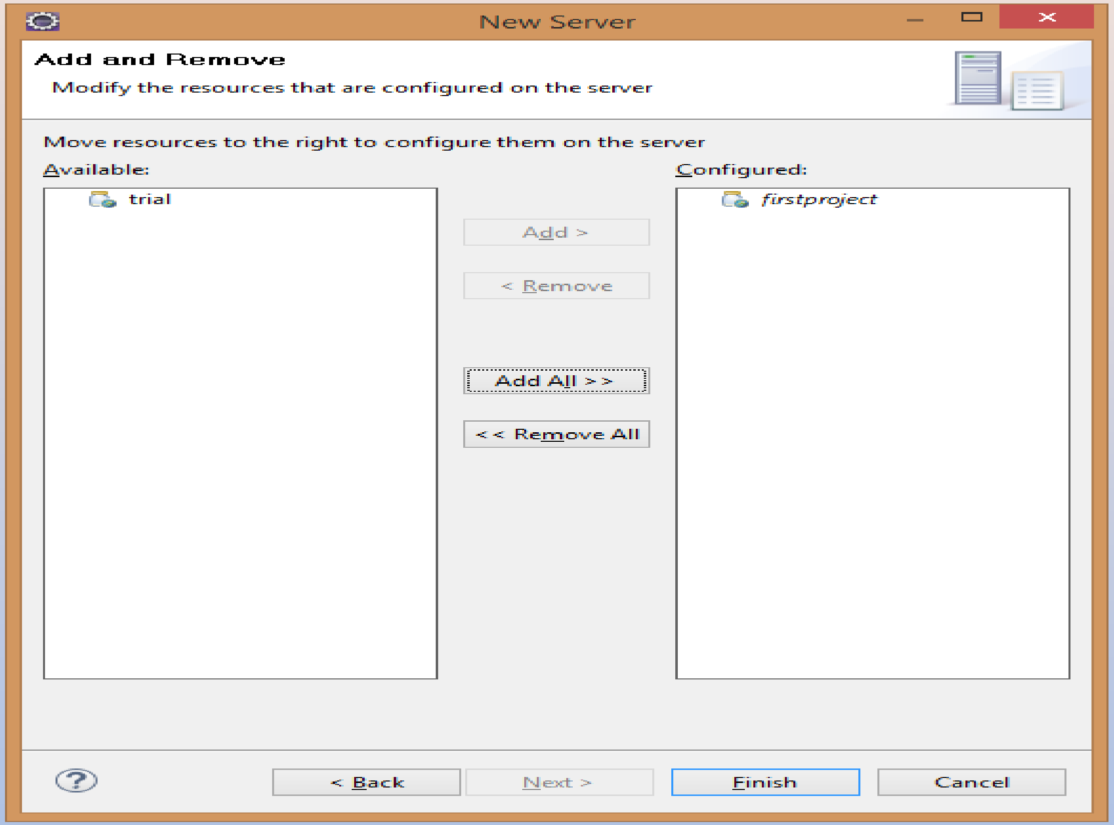
- Apache
  - Tomcat v3.2 Server
  - Tomcat v4.0 Server
  - Tomcat v4.1 Server
  - Tomcat v5.0 Server
  - Tomcat v5.5 Server
  - Tomcat v6.0 Server
  - Tomcat v7.0 Server
  - Tomcat v8.0 Server
  - Tomcat v8.5 Server
  - Tomcat v9.0 Server
  - Tomcat v10.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name:	localhost
Server name:	Tomcat v9.0 Server at localhost (2)
Server runtime environment:	Apache Tomcat v9.0

Setting up Eclipse

- **Add the resource** in Configuration
- Then **Browse for path of Tomcat** and click Finish.



Setting up Eclipse

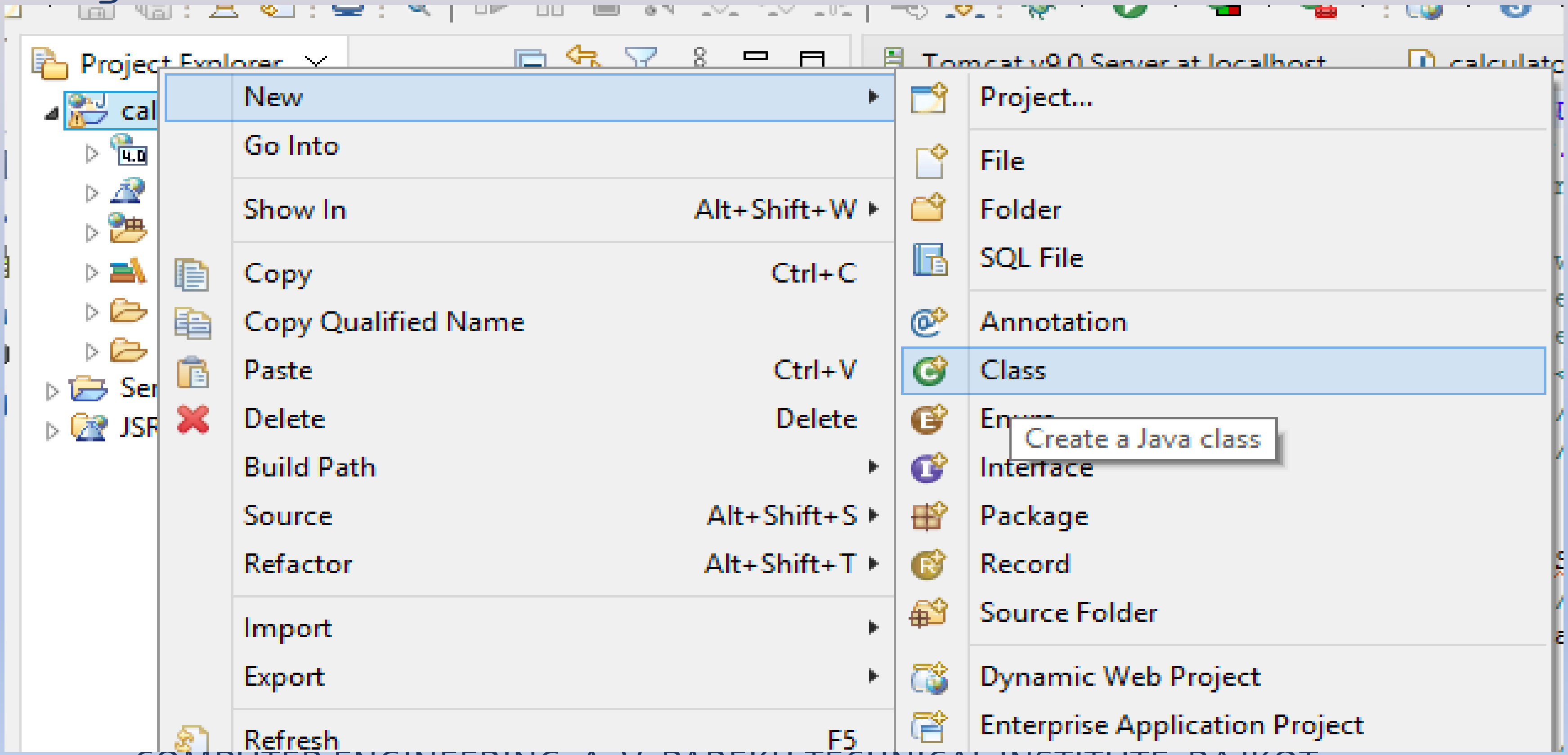
```
package firstproject;
import java.io.IOException; import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/serv")
public class Serv extends HttpServlet {
	private static final long serialVersionUID = 1L;
	protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
			response.getWriter().write("Hello World from Web Service.");
	}

}
```

Setting up Eclipse

- Right click on the project and select “Run on”
- Then select “Run on Server”.
- If the output is generated then our set-up for eclipse is completed.

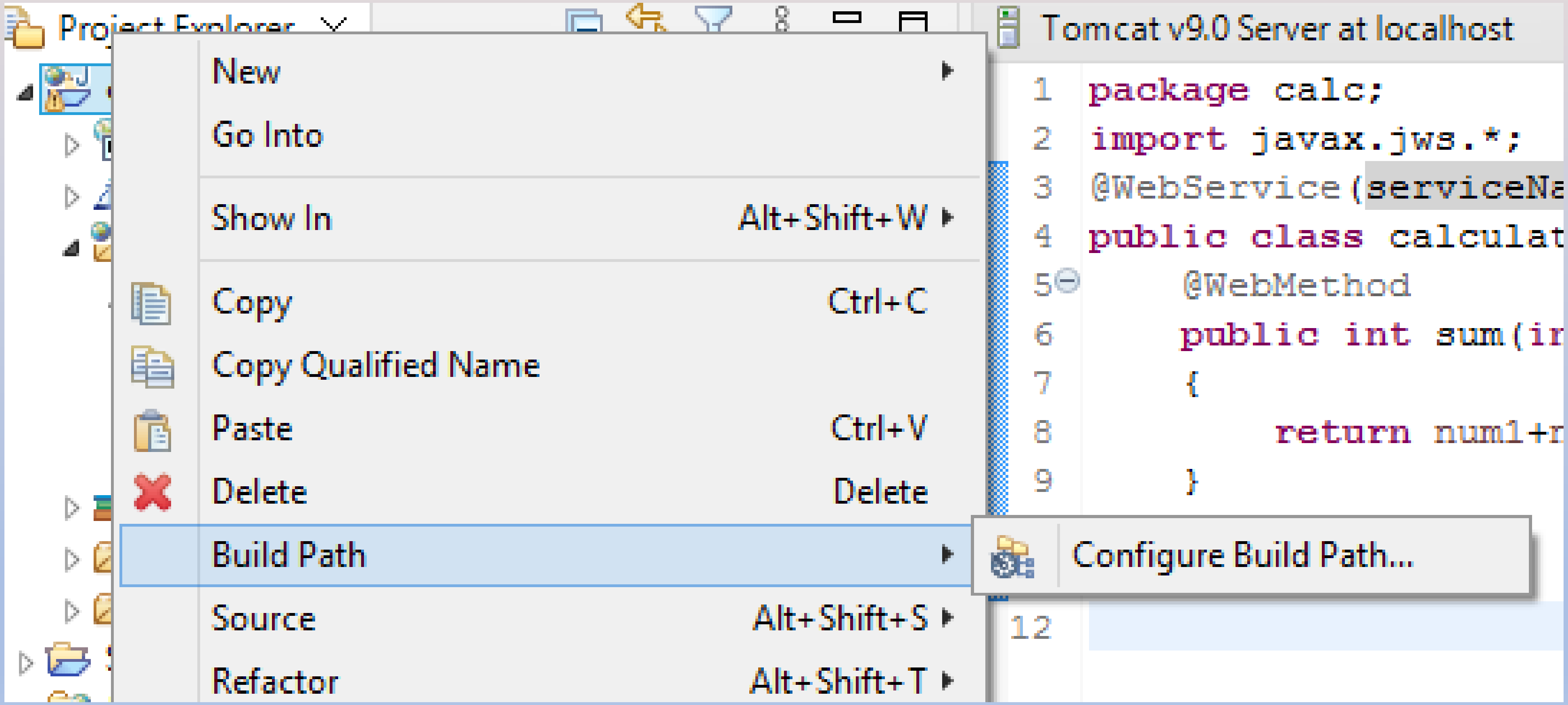
Simple Calculator SOAP Web service

- Create a "Dynamic web project" named "calc"
- Create a class named "Calculator.java" in the "src/main/java"



Simple Calculator SOAP Web service

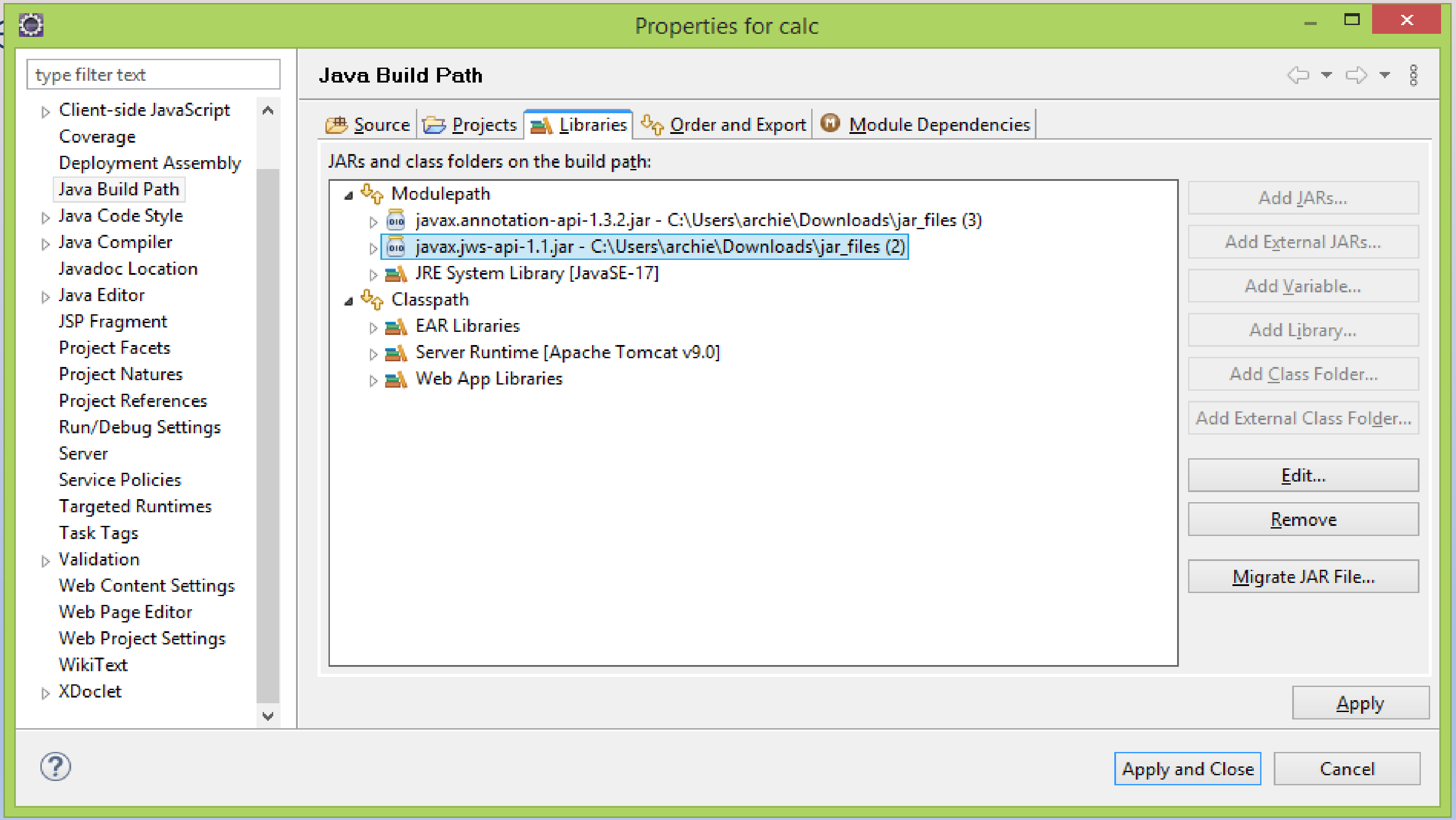
- Add the jar file for web services using "Build path"



```
package calc;
import javax.jws.*;
@WebService(serviceNa
public class calculat
    @WebMethod
    public int sum(ir
    {
        return num1+r
    }
```

Simple Calculator SOAP Web service

- Select the path for external jar file for “javax.jws” package



Properties for calc

type filter text

Client-side JavaScript
Coverage
Deployment Assembly
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
JSP Fragment
Project Facets
Project Natures
Project References
Run/Debug Settings
Server
Service Policies
Targeted Runtimes
Task Tags
Validation
Web Content Settings
Web Page Editor
Web Project Settings
WikiText
XDoclet

Java Build Path

Source | Projects | Libraries | Order and Export | Module Dependencies

JARs and class folders on the build path:

```
Modulepath
    javax.annotation-api-1.3.2.jar - C:\Users\archie\Downloads\jar_files (3)
    javax.jws-api-1.1.jar - C:\Users\archie\Downloads\jar_files (2)
    JRE System Library [JavaSE-17]
Classpath
    EAR Libraries
    Server Runtime [Apache Tomcat v9.0]
    Web App Libraries
```

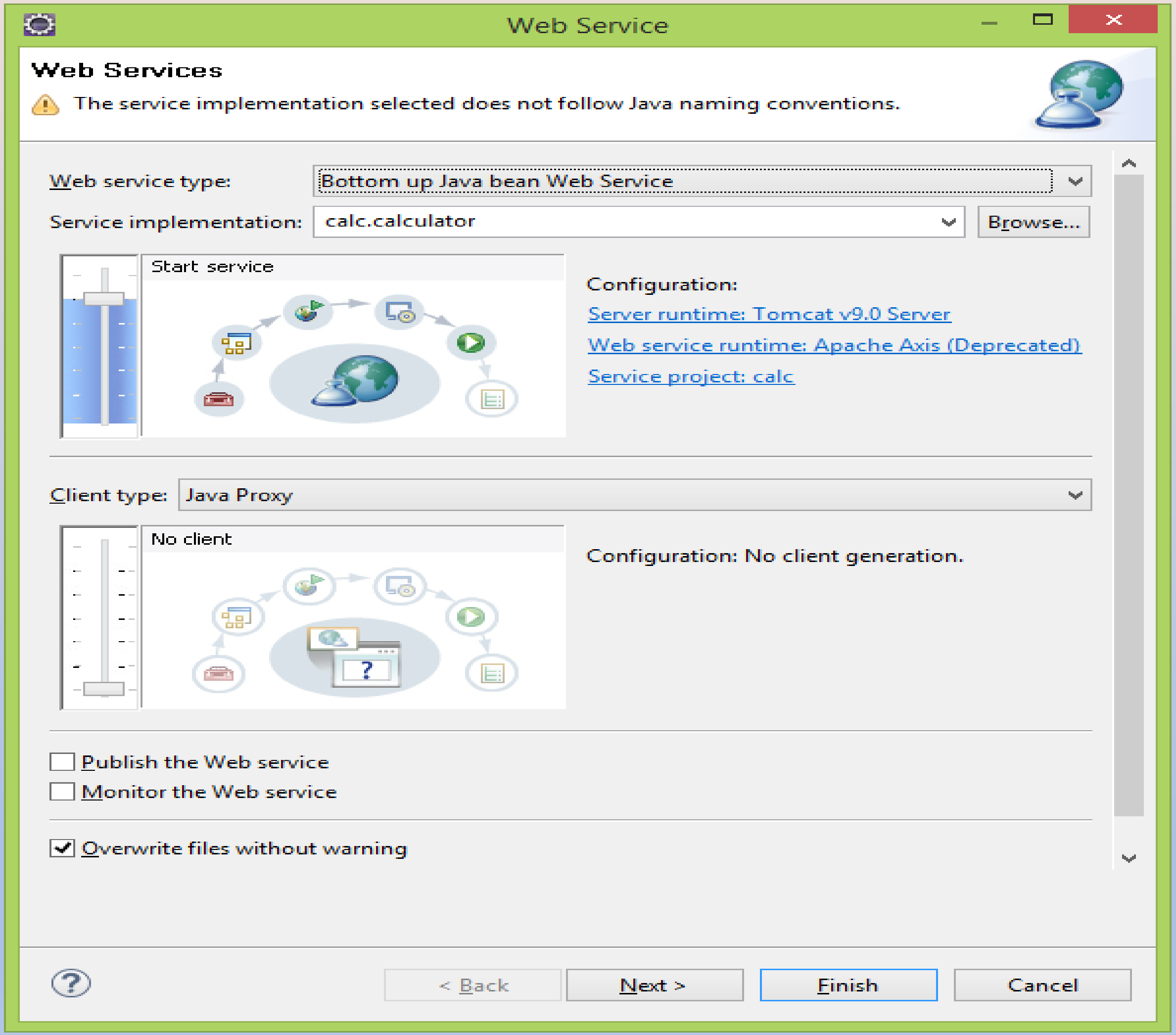
Simple Calculator SOAP Web service

- Create a sum method inside calculator class

```
package calc;
import javax.jws.*;

@WebService(serviceName="calcprogram")
public class Calculator {
    @WebMethod
    public int sum(int num1,int num2) {
        return _num1 + num2);
    }
}
```

Simple Calculator SOAP Web service

- Create a web service for "Calculator" class
  - Right click on the Calculator.java under projects
  - Under "web services" select "create web service".



Web Service

Web Services

The service implementation selected does not follow Java naming conventions.

Web service type: Bottom up Java bean Web Service

Service implementation: calc.calculator

Configuration:
Server runtime: Tomcat v9.0 Server
Web service runtime: Apache Axis (Deprecated)
Service project: calc

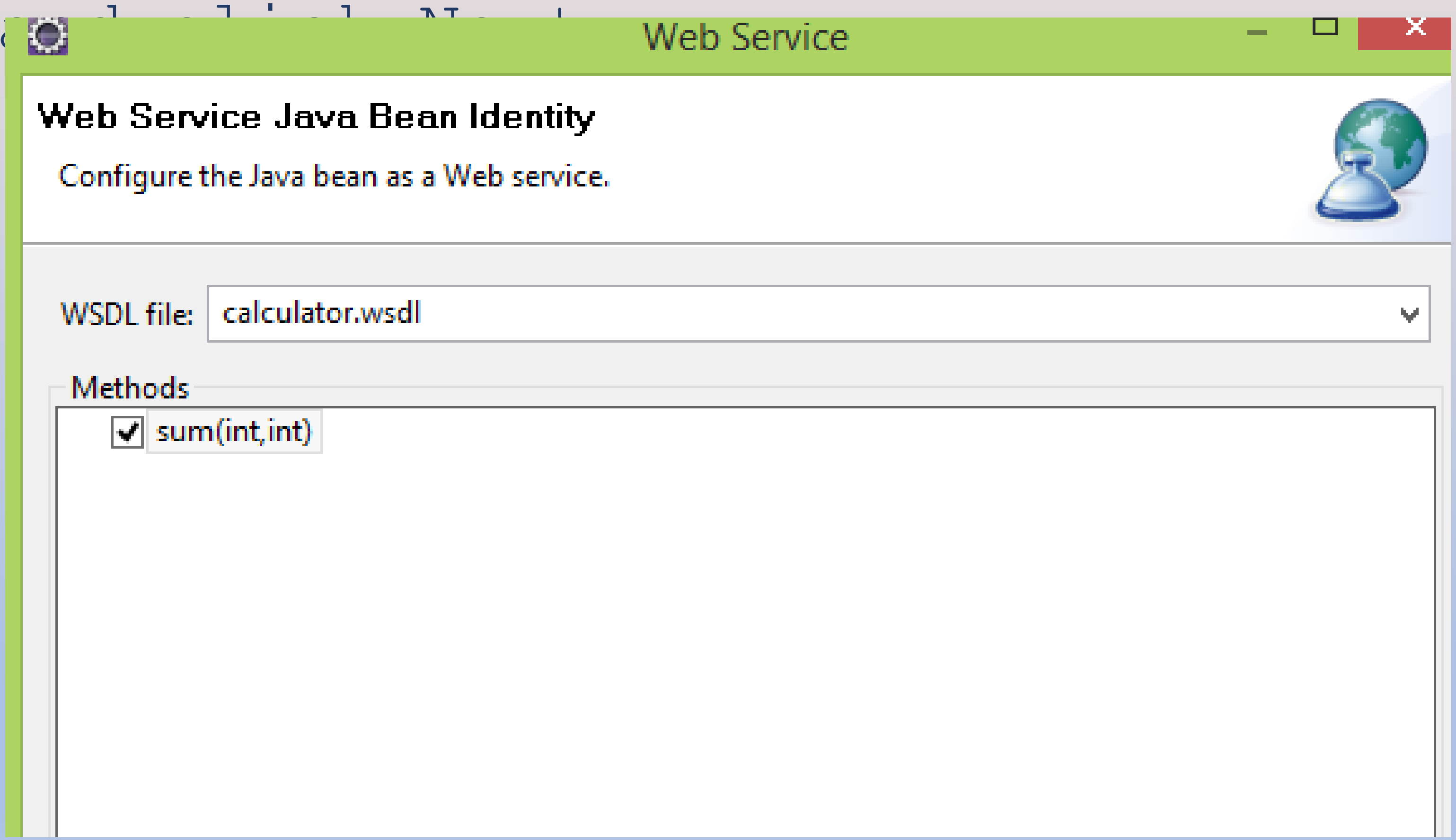
Client type: Java Proxy

Configuration: No client generation.

- [ ] Publish the Web service
- [ ] Monitor the Web service
- [x] Overwrite files without warning

Simple Calculator SOAP Web service

- Click Next and select the methods to be included in service



Web Service Java Bean Identity

Configure the Java bean as a Web service.

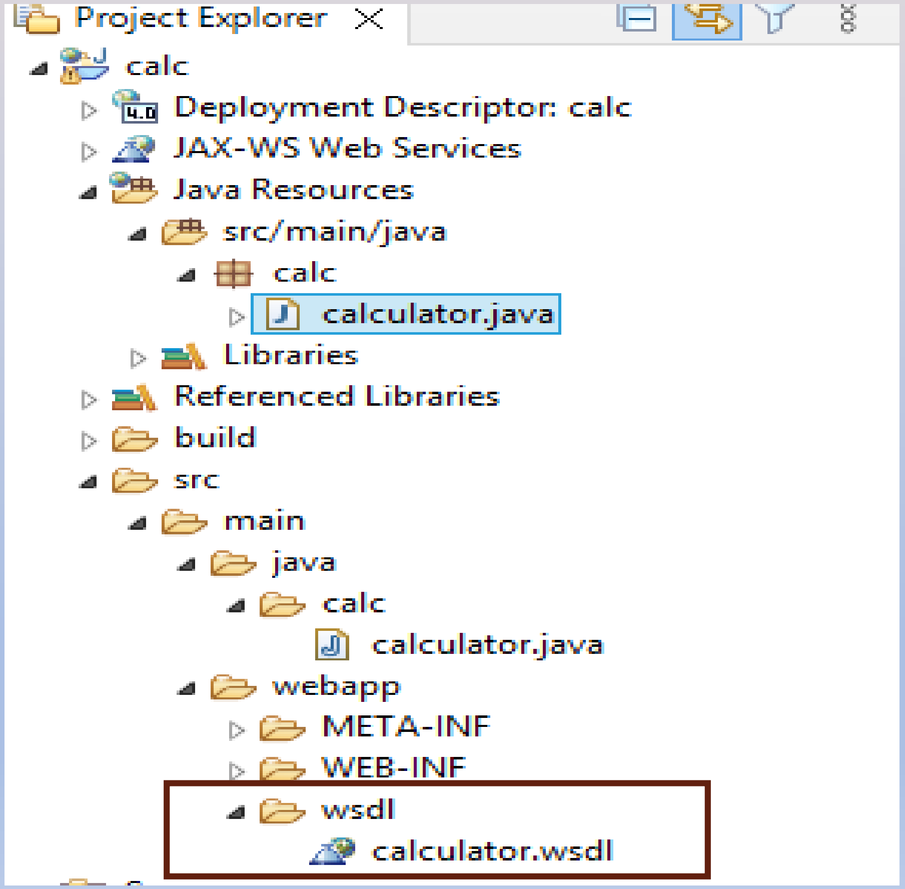
WSDL file: calculator.wsdl

Methods

- [✓] sum(int,int)

Simple Calculator SOAP Web service

- Turn on the servcer and Click Finish
- Under the "webapps" directory, expand wsdl to find "Calculator.wsdl"

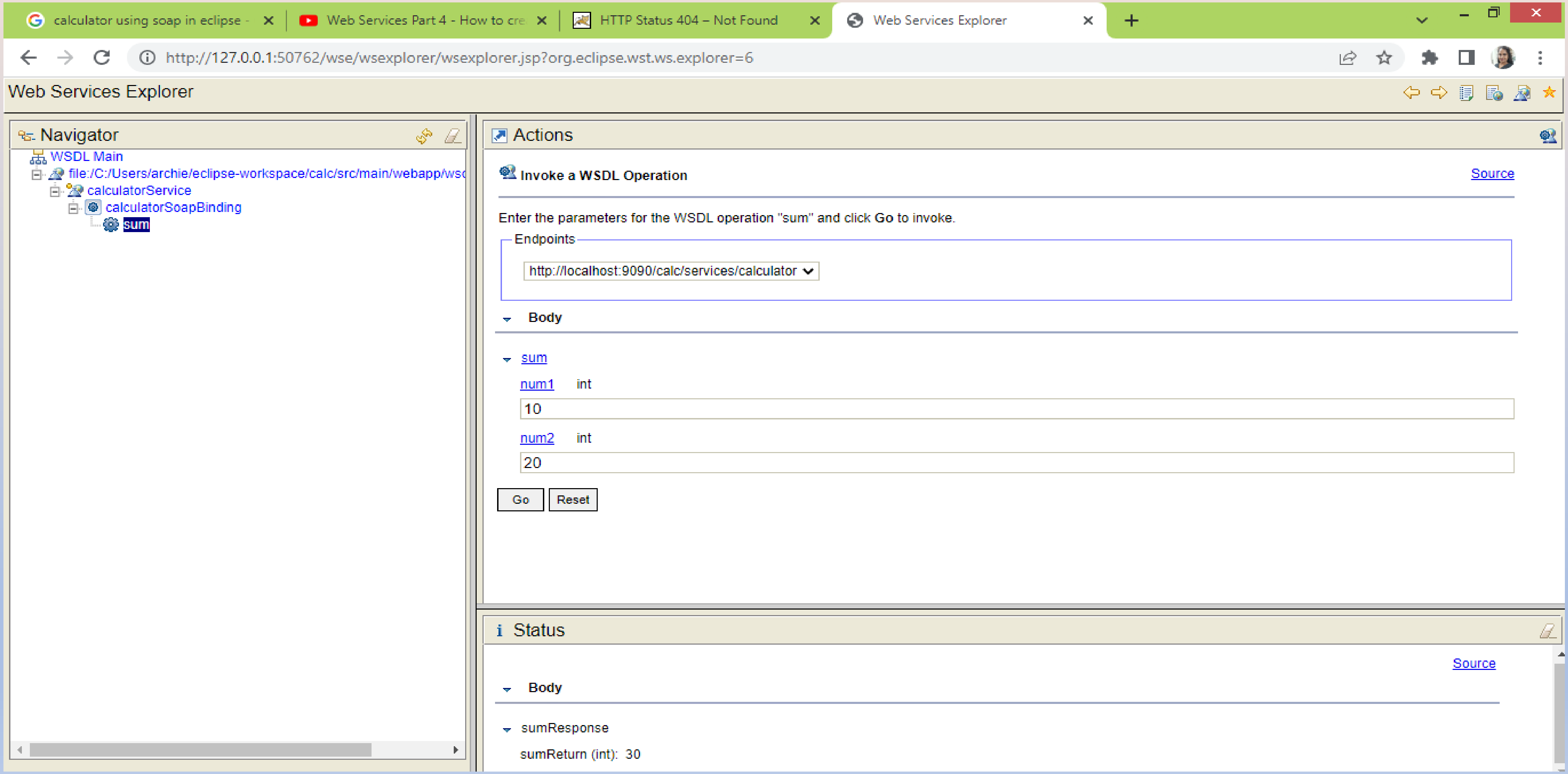


```
Project Explorer
◢ calc
  ▷ Deployment Descriptor: calc
  ▷ JAX-WS Web Services
  ◢ Java Resources
    ◢ src/main/java
      ◢ calc
        ▷ calculator.java
    ▷ Libraries
  ▷ Referenced Libraries
  ▷ build
  ◢ src
    ◢ main
      ◢ java
        ◢ calc
            calculator.java
      ◢ webapp
        ▷ META-INF
        ▷ WEB-INF
        ◢ wsdl
            calculator.wsdl
```

Simple Calculator SOAP Web service

- Right click the “calculator.wsdl”, select web services.
- Further select “Test with Web Services explorer”.
- Under the wsdl operations, enter parameters (two numbers to be added) in the given textboxes.

Simple Calculator SOAP Web service


calculator using soap in eclipse -
Web Services Part 4 - How to cre
HTTP Status 404 – Not Found
Web Services Explorer
http://127.0.0.1:50762/wse/wsexplorer/wsexplorer.jsp?org.eclipse.wst.ws.explorer=6
Web Services Explorer
Navigator
WSDL Main
file:/C:/Users/archie/eclipse-workspace/calc/src/main/webapp/wsc
calculatorService
calculatorSoapBinding
sum
Actions
Go
Reset
Status
Source

Invoke a WSDL Operation

Enter the parameters for the WSDL operation "sum" and click Go to invoke.

Endpoints

http://localhost:9090/calc/services/calculator

Body

sum

num1 int

10

num2 int

20

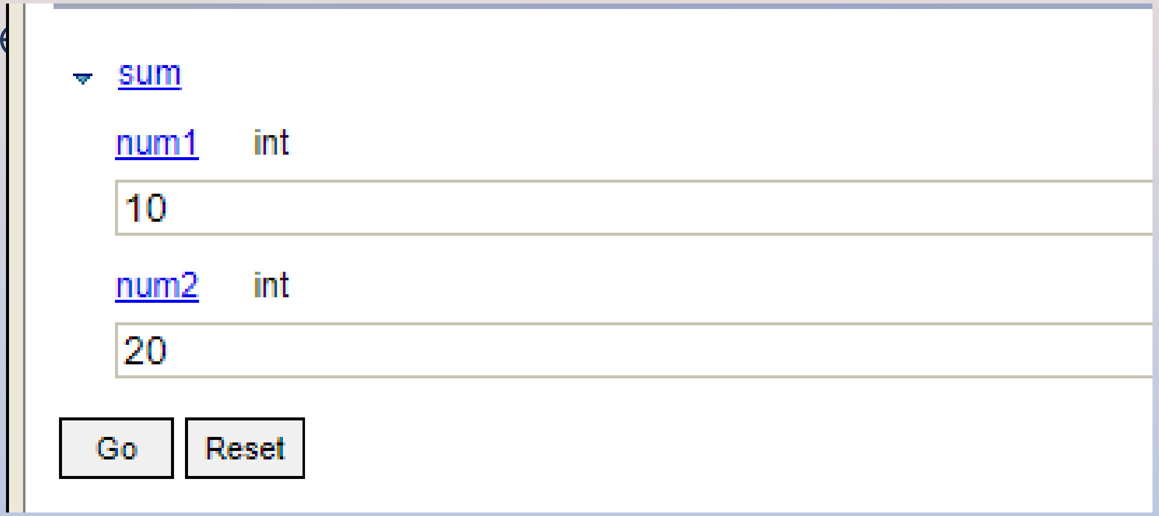
Body

sumResponse

sumReturn (int): 30

Simple Calculator SOAP Web service

- Input in Web service explore



sum

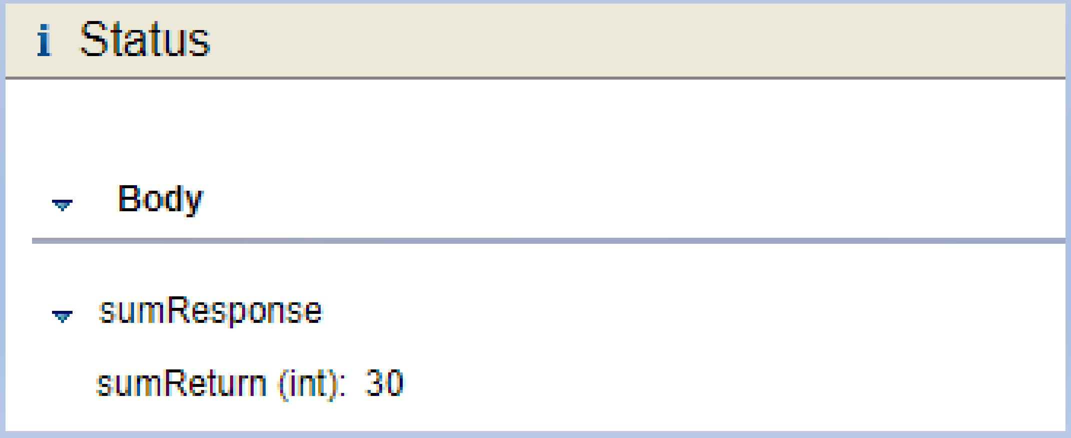
num1 int

10

num2 int

20

- Response



Status

Body

sumResponse

sumReturn (int): 30