Module 3
SOAP Web Service

What are SOAP web services?

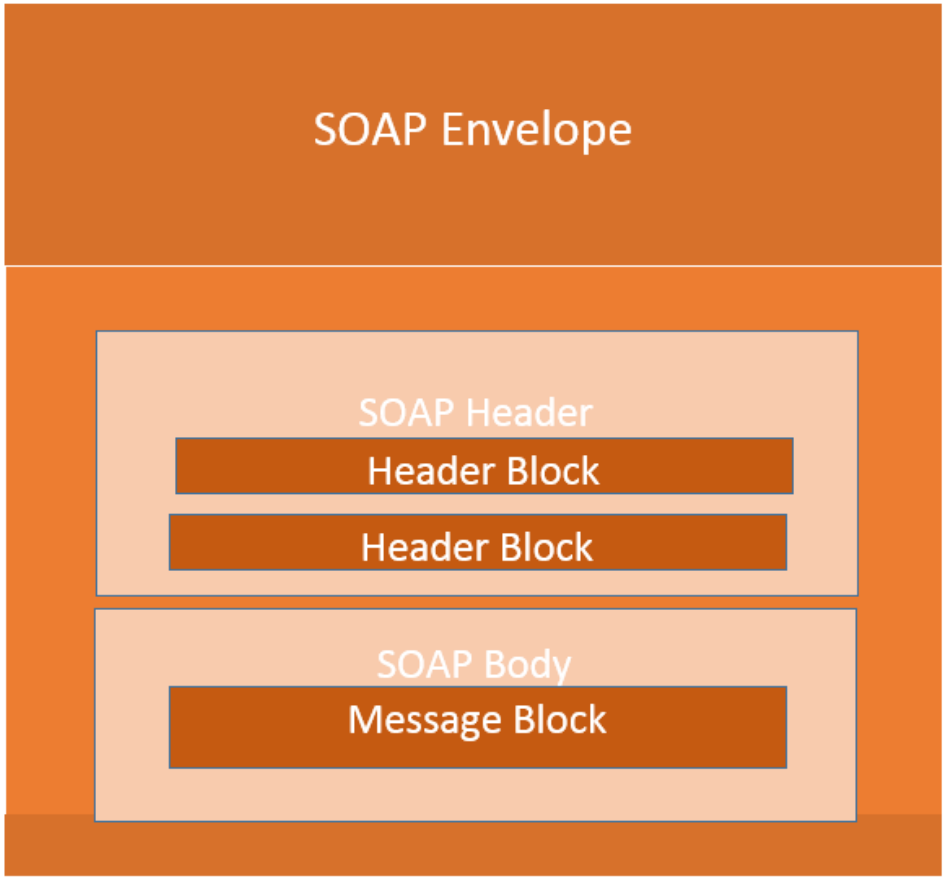
- A web service that complies to **SOAP web services specification** is a SOAP web services
- SOAP web service specifications:
- **Basic:**
  - **SOAP**
  - **WSDL –** Web Service Description Language
  - **UDDI –** Universal Description Discovery Integration

SOAP web services

- SOAP stand for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an **application communication** protocol
- SOAP is a **format** for sending and receiving messages
- SOAP is **platform independent**
- SOAP is based on **XML**
- SOAP is a W3C recommendation

SOAP Building blocks

- The SOAP specification defines something known as a "**SOAP message**" which is what is sent to the web service and the client application.
- The below diagram of SOAP architecture shows the various building blocks of a SOAP Message.



SOAP Message structure

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

SOAP Message

- The required SOAP Envelope element is the root element of a SOAP message. This element defines the **XML document as a SOAP message.**
- namespace defines the Envelope as a SOAP Envelope.
- A SOAP message has no default encoding.

- ```
  <?xml version="1.0"?>

  <soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
    ...
    Message information goes here
    ...
  </soap:Envelope>
  ```
  ```

SOAP Message

- The optional SOAP Header element contains application-specific information (like authentication, payment, etc) about the SOAP message.
- If the Header element is present, it must be the first child element of the Envelope element.
- Suppose we wanted to send a structured data type (complex type)

```
<soapenv:Header>
</e:Book>
  <element name="Book">
  <complexType>
 <element name="author" type="xsd:string"/>
 <element name="preface" type="xsd:string"/>
  <element name="intro" type="xsd:string"/>
  </complexType>
  </e:Book>
 </soapenv:Header>›
```

SOAP Message Request

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-
envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap
-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/p
rices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

SOAP Message Response

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-
encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/
prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

SOAP Message

- One thing to note is that SOAP messages are normally auto-generated by the web service when it is called.
- Whenever a client application calls a method in the web service, the web service will **automatically generate a SOAP message** which will have the necessary details of the data which will be sent from the web service to the client application.

The Fault message

- When a request is made to a SOAP web service, the response returned can be of either 2 forms which are a **successful response or an error response.**
- When a success is generated, the response from the server will always be a SOAP message. But if SOAP faults are generated, they are returned as **“HTTP 500” errors**.

WSDL

- WSDL stands for **Web Services Description Language**
- WSDL is used to describe web services
- **WSDL is written in XML**
- WSDL Documents
  - An WSDL document describes a web service. It specifies the **location of the service, and the methods of the service**, using these major elements.

WSDL document

Element	Description
<types>	Defines the (XML Schema) data types used by the web service
<message>	Defines the data elements for each operation
<portType>	Describes the operations that can be performed and the messages involved.
<binding>	Defines the protocol and data format for each port type

Structure of a WSDL document

```
<definitions>

 <types>
   data type definitions.....
 </types>

 <message>
   definition of the data being communicated....
 </message>

 <portType>
   set of operations.....
 </portType>

 <binding>
   protocol and data format specification....
 </binding>


 <service> service name and binding information  </service>

 </definitions>
```

WSDL document

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

WSDL document

- In this example the **<portType>** element defines "glossaryTerms" as the name of a **port**, and "getTerm" as the name of an **operation**.
- The "getTerm" operation has an **input message** called "getTermRequest" and an **output message** called "getTermResponse".
- The **<message>** elements define the **parts** of each message and the associated data types.
- The <portType> element defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.
- The request-response type is the most common operation type

Port Type element

Type	Definition
One-way	The operation can receive a message but will not return a response
Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

WSDL One-Way Operation

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value".
However**, no output is defined for the operation**.

WSDL Request-Response Operation

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".

Building SOAP Web Service

- **Step 1: Create Web service Project**
- Step 2: Add Web service
- Step 3: Deploy the web service
- Step 4: Test the Web methods
- **Step 5: Creating Web service Client**
- Step 6: Consuming web methods

Creating Web Methods

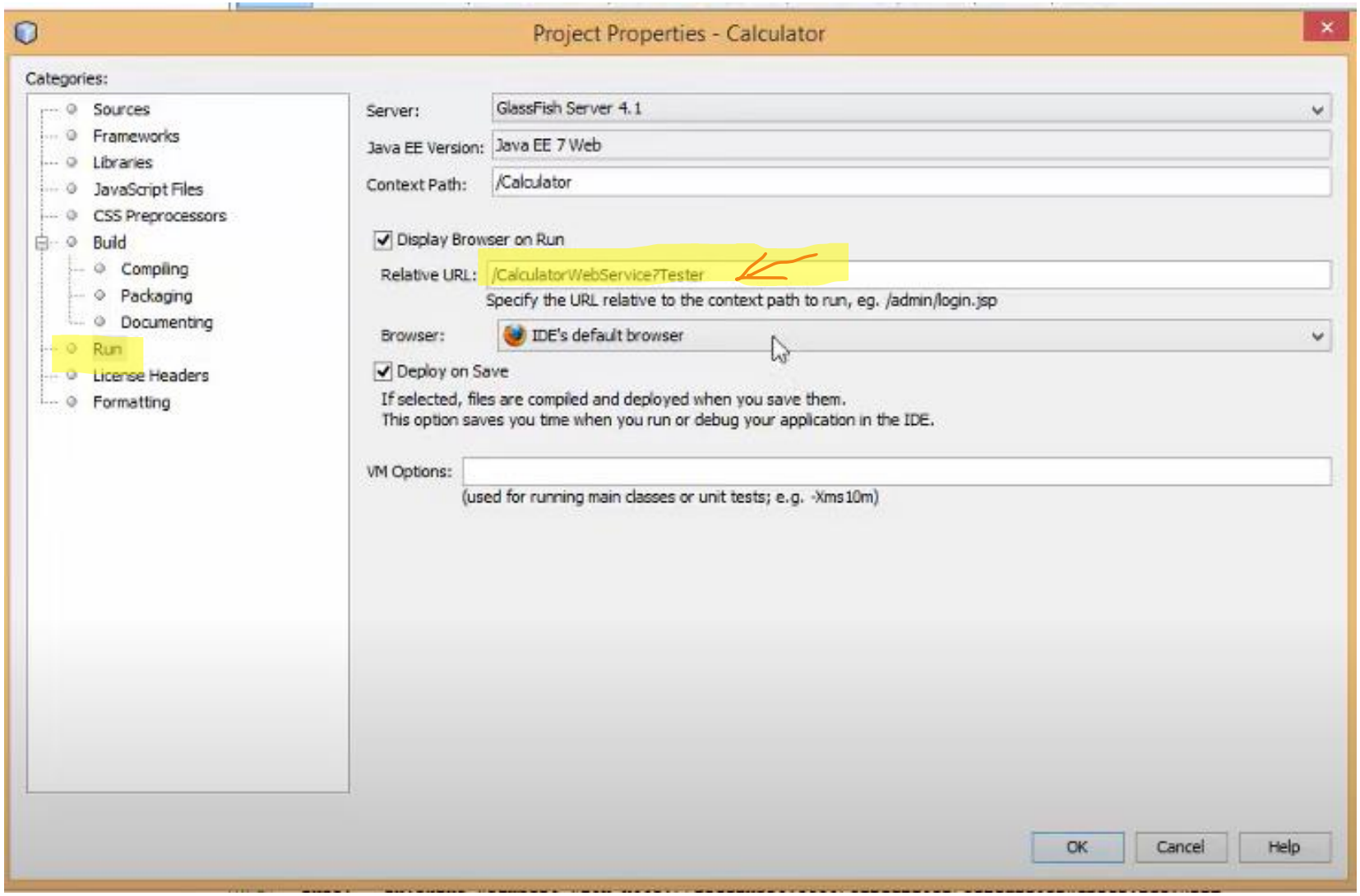
```
*/
@WebService(serviceName = "CalculatorWebService")
public class CalculatorWebService {

    @WebMethod(operationName = "AddIntegers")
    public int add(@WebParam(name = "firstNum") int num1,
            @WebParam(name = "secondNum") int num2) {
        return num1 + num2;
    }

    @WebMethod(operationName = "SubIntegers")
    public int sub(@WebParam(name = "firstNum") int num1,
            @WebParam(name = "secondNum") int num2) {
        return num1 - num2;
    }

    @WebMethod(operationName = "MulIntegers")
    public int mul(@WebParam(name = "firstNum") int num1,
            @WebParam(name = "secondNum") int num2) {
        return num1 * num2;
    }
```

Testing Web Methods



Testing Web Methods

Testing Web Methods

subIntegers Method invocation

Method parameter(s)

Type	Value
int	8
int	4

Method returned

int : "4"

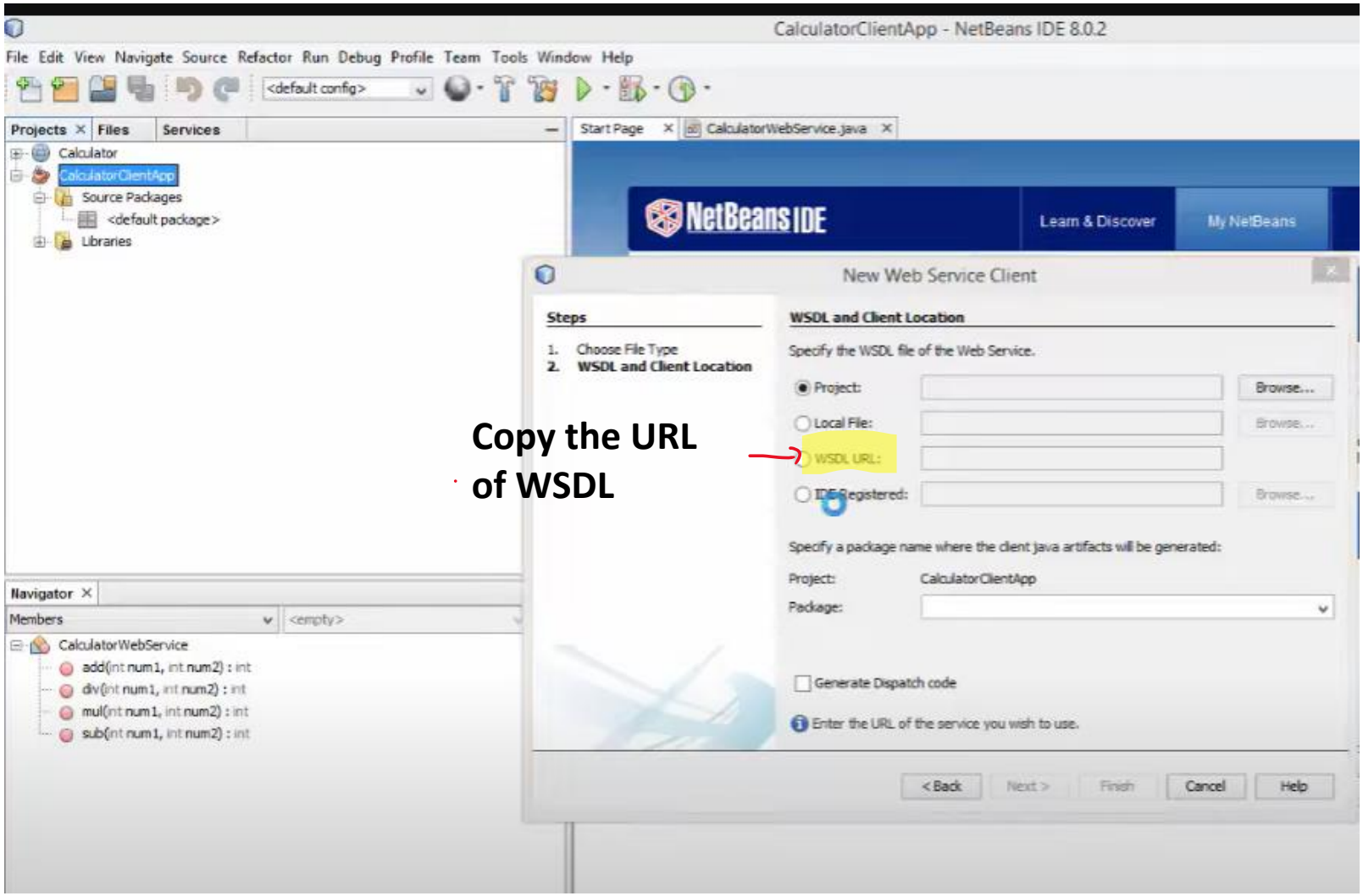
SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:SubIntegers xmlns:ns2="http://javawebservice.beingzero.in/">
            <firstNum>8</firstNum>
            <secondNum>4</secondNum>
        </ns2:SubIntegers>
    </S:Body>
</S:Envelope>
```

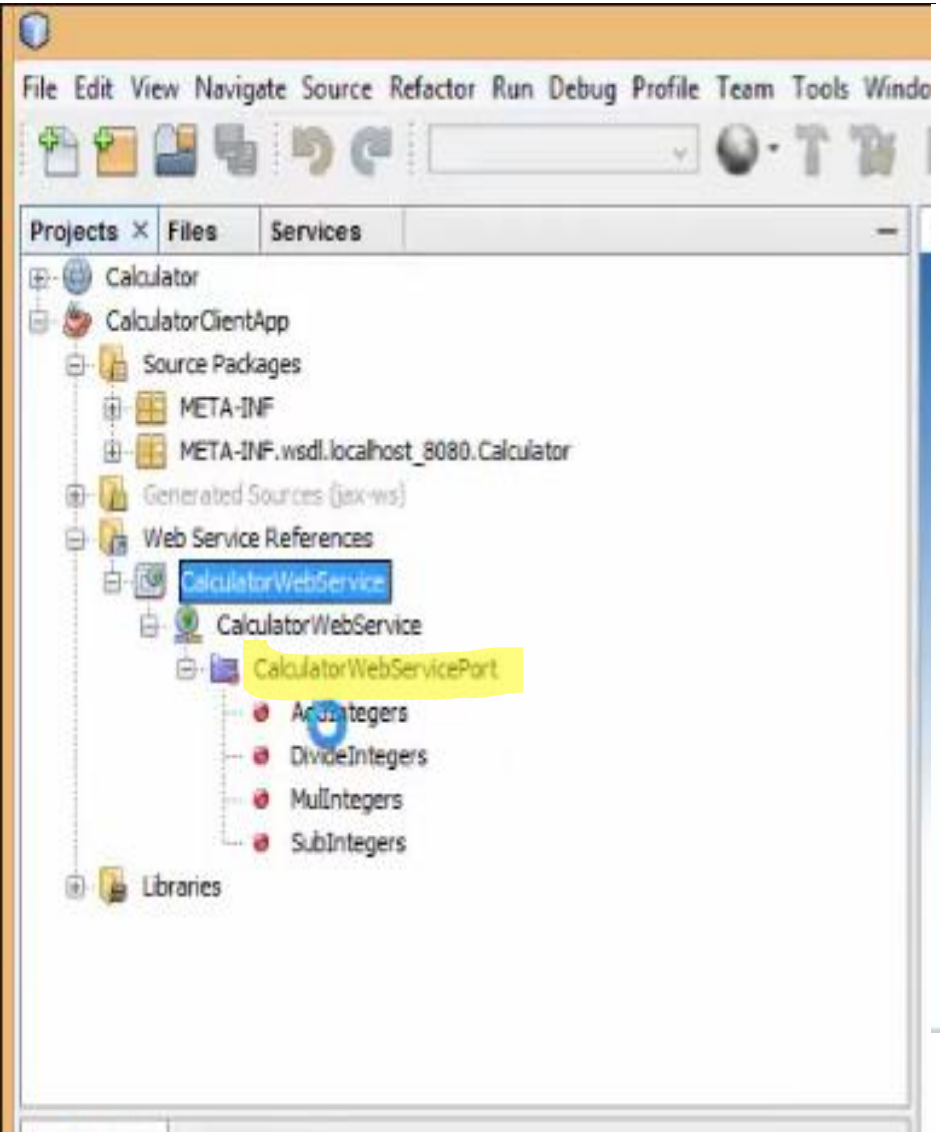
SOAP Response

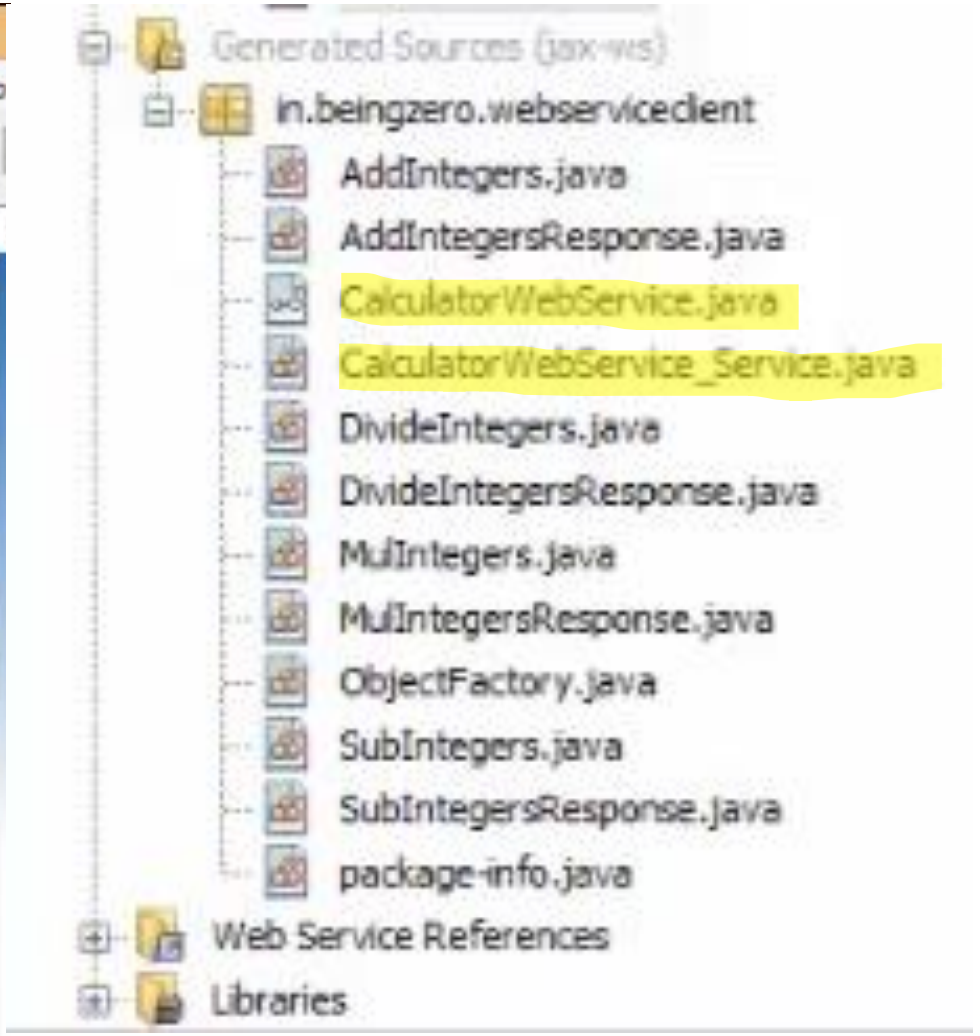
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
```

Web Service Client Application



Web Service Client Application





Consuming web Methods

```
Public class CalculatorClient {
public static void main(Strings arg[]){

CalculatorWebService proxy;
 proxy= (new
CalculatorwebService_Service()).getCalculatorServicePort();

int result= proxy.AddIntegers(10,20);

System.out.println("The addition is " + result);
}
}
```

Applications

- used for enterprise-level web services that require high security and complex transactions
  - APIs for financial services, payment gateways, CRM software, identity management, and telecommunication services are commonly used examples of SOAP
- need to process stateful operations
  - SOAP APIs are stateless by default, SOAP does support stateful operations that can be implemented using the WS (Web Services) Specifications
  - built on top of the core XML and SOAP standards.

SOAP Limitations

- As SOAP can only transfer messages as XML files
- In SOAP, the client-server communication depends on WSDL (Web Service Description Language) contracts, which implies tight coupling
  - it's not recommended for loosely coupled applications
- SOAP also has a higher learning curve, is harder to code and can't be tested in the web browser
  - generate contracts in WSDL, create client stubs, follow strict specifications

Thank you