



# Outline

---

- ▶ Apache Axis
- ▶ Web Service Clients
- ▶ Creating Web Services



# WS basics (I)

---

- ▶ Web services are described by their WSDL file
- ▶ Starting from the information in the WSDL, a WS client must:
  - ▶ Prepare call data parameters
  - ▶ Encapsulate them in a SOAP XML message
  - ▶ Send the message to the specified endpoint, using the right encoding and protocol
  - ▶ Wait... (remotely)
  - ▶ Receive the SOAP XML response message
  - ▶ Decode the message and extract the response data values
  - ▶ Handle possible errors and exceptions, and report them to the caller
  - ▶ Return the response data to the caller

# WS basics (II)

---

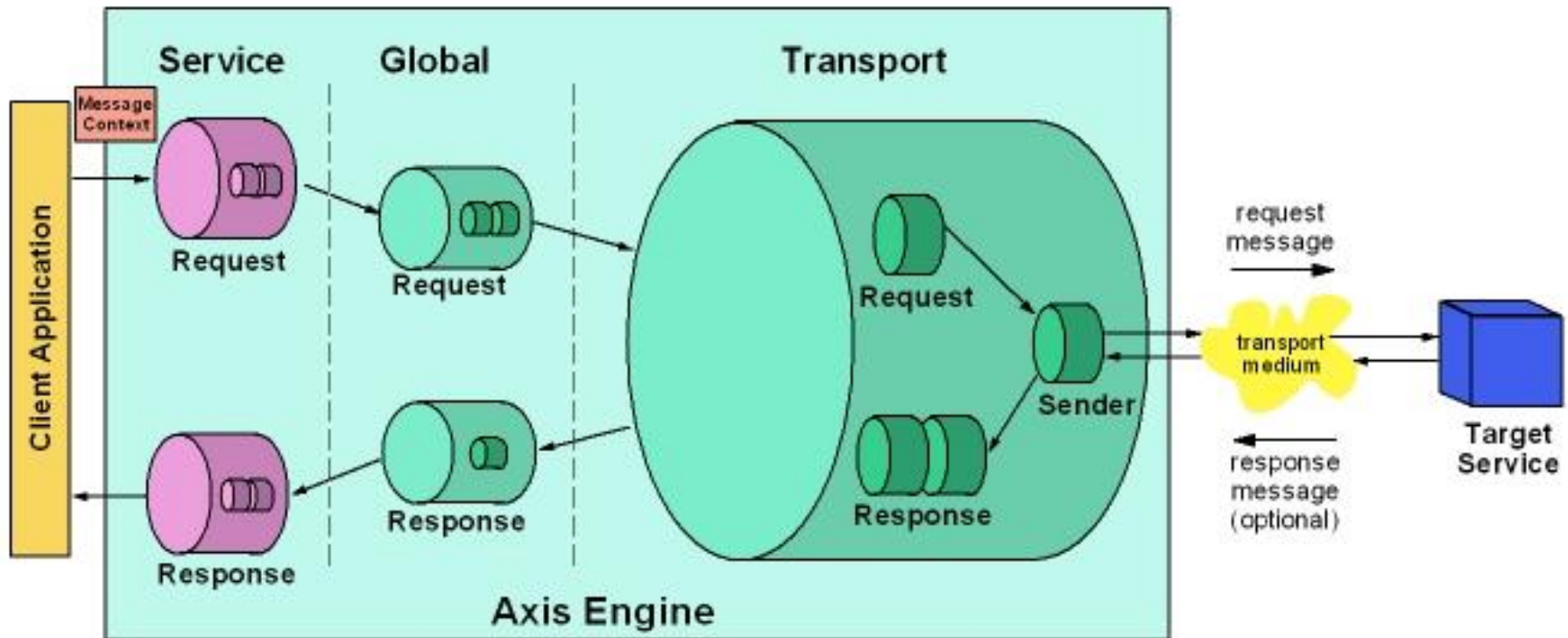
- ▶ **The provider of the WS (the server side) must:**
  - ▶ Publish the WSDL corresponding to its calling methods
  - ▶ Wait for incoming requests as SOAP XML messages
  - ▶ Decode the request parameters and validate their data types
  - ▶ Identify the called method, and invoke the right class for handling it (business method)
  - ▶ Wait... (locally)
  - ▶ Get the response from the business method
  - ▶ Catch any exception and possibly create an XML exception message
  - ▶ Encapsulate the response values in a SOAP XML message
  - ▶ Return the response message to the caller

# Welcome, Axis

---

- ▶ **Project Axis** ([axis.apache.org](http://axis.apache.org)) aims at simplifying and automating all the above steps, as much as possible
- ▶ **Automatically creates Java «proxy» classes for calling web services**
  - ▶ You call a method on the local object
  - ▶ The method call is translated into a WS request/response
- ▶ **Helps transforming business methods into Web Services**
  - ▶ Starting from a Java class, creates the WSDL and the Servlets to handle the WS request (bottom-up creation)
  - ▶ Starting from a WSDL, create the servlet and Java interfaces to handle the requests (top-down)

# Axis architecture (client pipeline)



# Axis architecture (client pipeline)

```
import Localhost.axis.Bing_jws.*;

public class MyBingClient {
    public static void main( String args[] ) throws Exception
    {
        BingService service = new BingServiceLocator();
        Bing myBing = service.getBing();
        // call instance methods on myBing, ex., say, myBing.foo()
        myBing.doSomething("hello") ;
    }
}
```

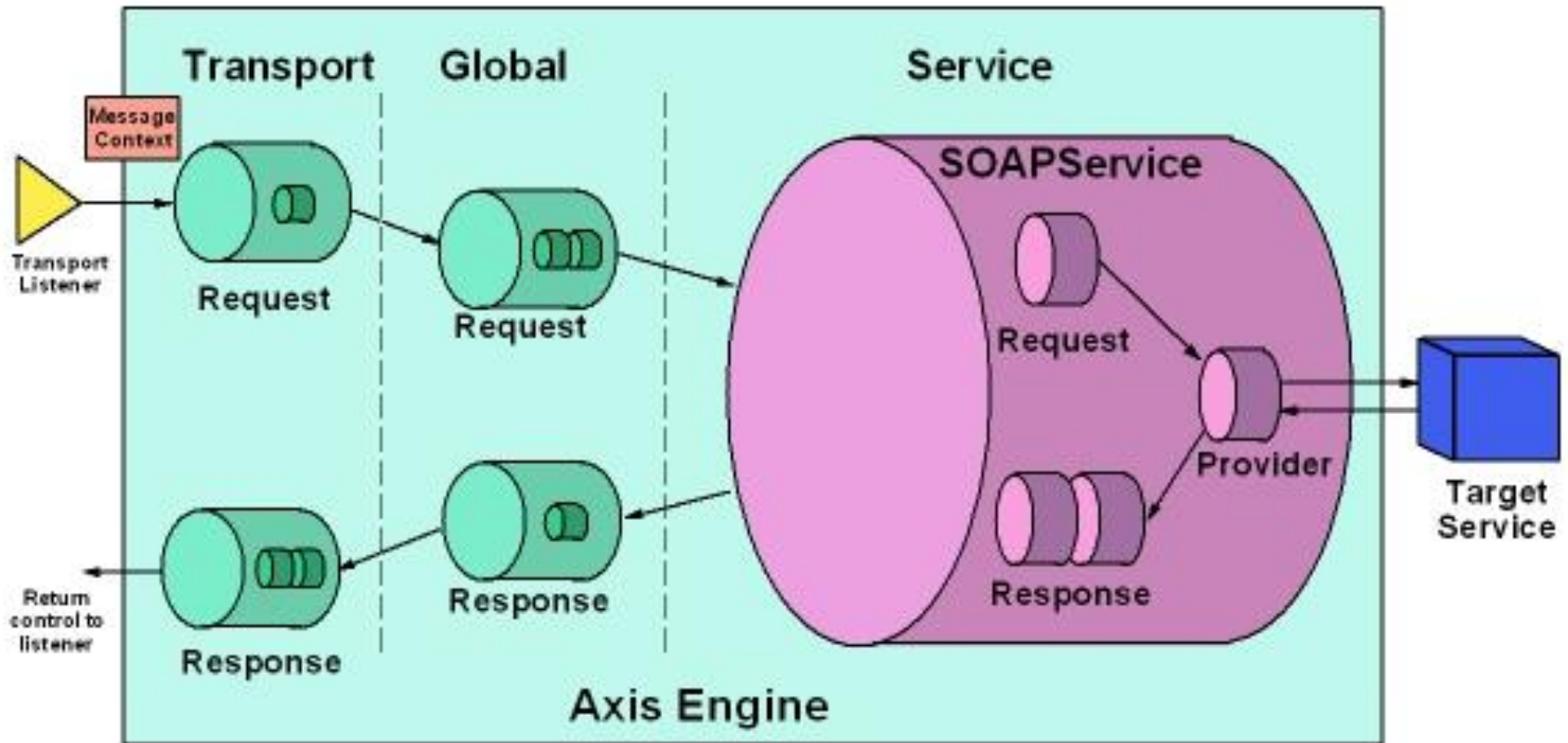
Client Application

Response

Axis Engine

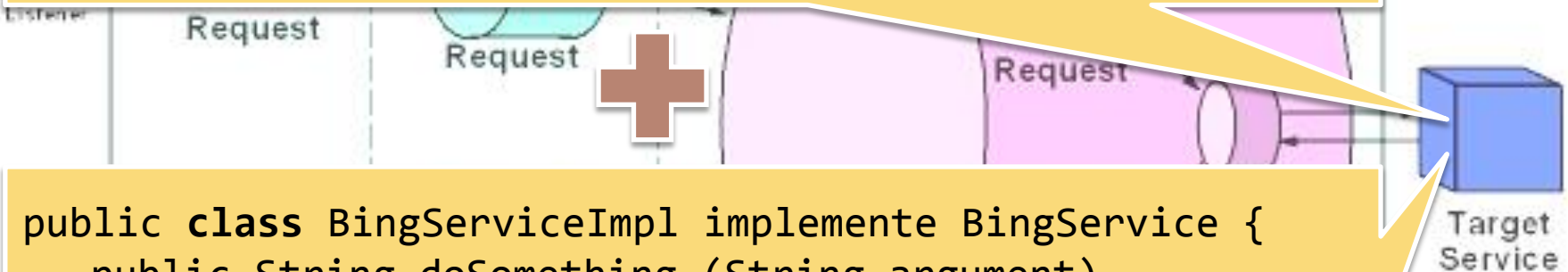
<http://wiki.apache.org/ws/FrontPage/Axis/WritingYourClient>

# Axis architecture (server pipeline)



# Axis architecture (server pipeline)

```
public interface BingService {  
    public String doSomething (String argument);  
    public ComplexTO retrieveResult(OtherTO value);  
}
```



```
public class BingServiceImpl implements BingService {  
    public String doSomething (String argument)  
    { ... }  
  
    public ComplexTO retrieveResult(OtherTO value)  
    { ... }  
}
```



# Guided example

---

- ▶ Currency Converter Web Service, hosted on <http://www.websvcex.net>
  - ▶ <http://www.websvcex.net/ws/wsdetails.aspx?wsid=10>
- ▶ Offers one operation: **ConversionRate**
  - ▶ Two input parameters: FromCurrency and ToCurrency (3-letter strings)
  - ▶ One output parameter: the conversion rate (double)

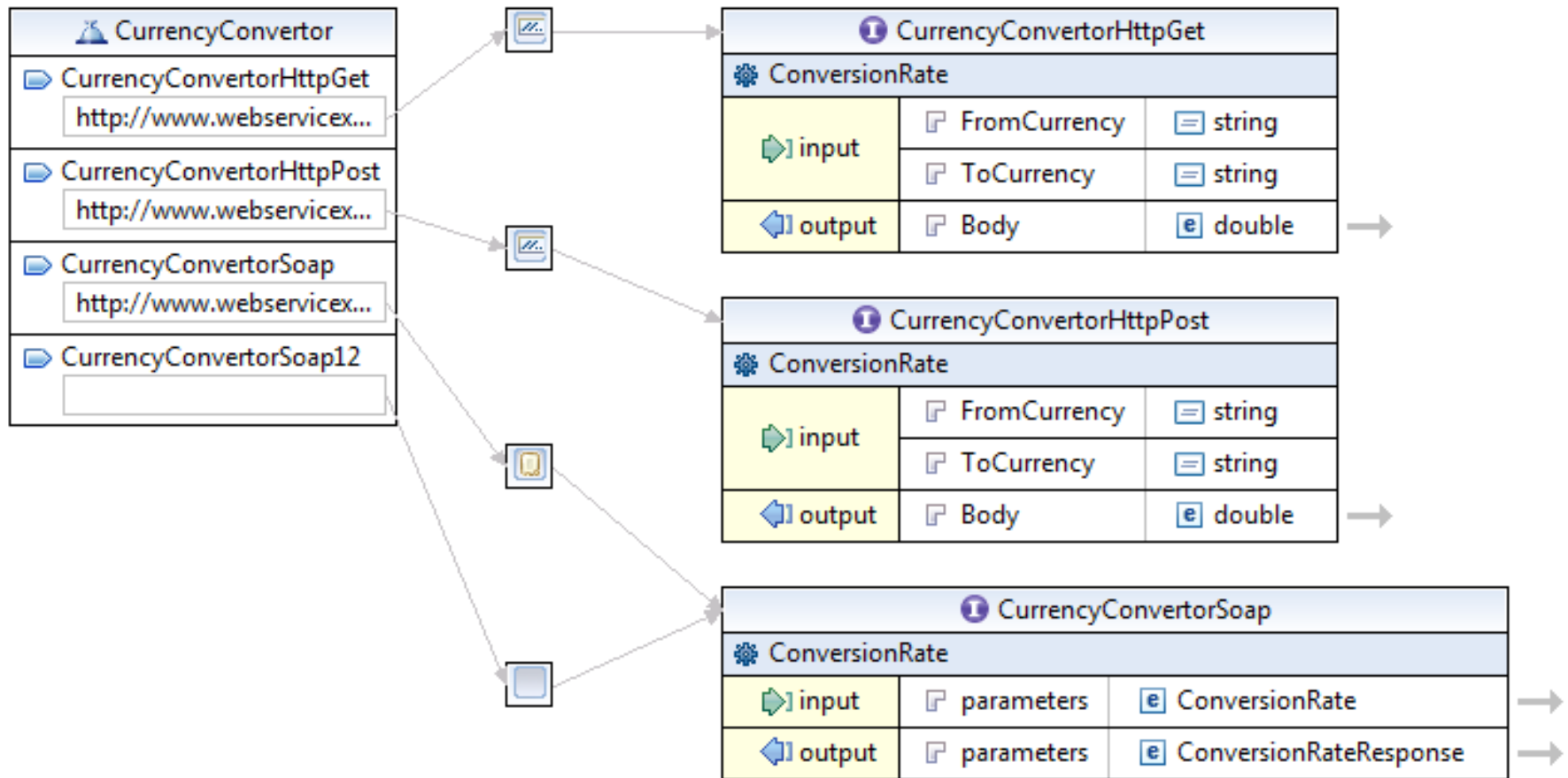
# Currency Converter WSDL

---

<http://www.websvcex.net/CurrencyConverter.asmx?WSDL>

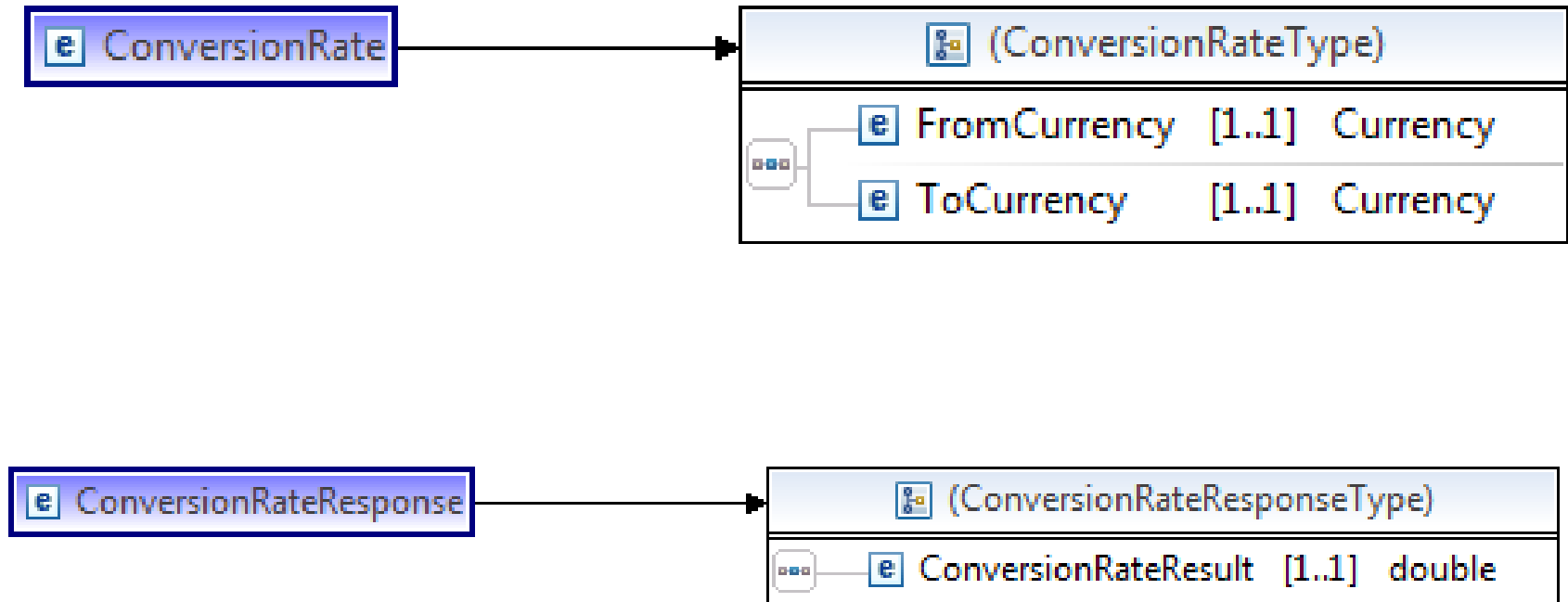


# Currency Converter WSDL (Eclipse WSDL)



# Parameter data types

---



# SOAP messages: request

---

**POST /CurrencyConvertor.asmx HTTP/1.1**

Host: www.webservicex.net

Content-Type: application/soap+xml; charset=utf-8

Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap12:Envelope
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
```

```
<soap12:Body>
```

```
  <ConversionRate xmlns="http://www.webserviceX.NET/">
```

```
    <FromCurrency>EUR</FromCurrency>
```

```
    <ToCurrency>USD</ToCurrency>
```

```
  </ConversionRate>
```

```
</soap12:Body>
```

```
</soap12:Envelope>
```



# SOAP messages: response

---

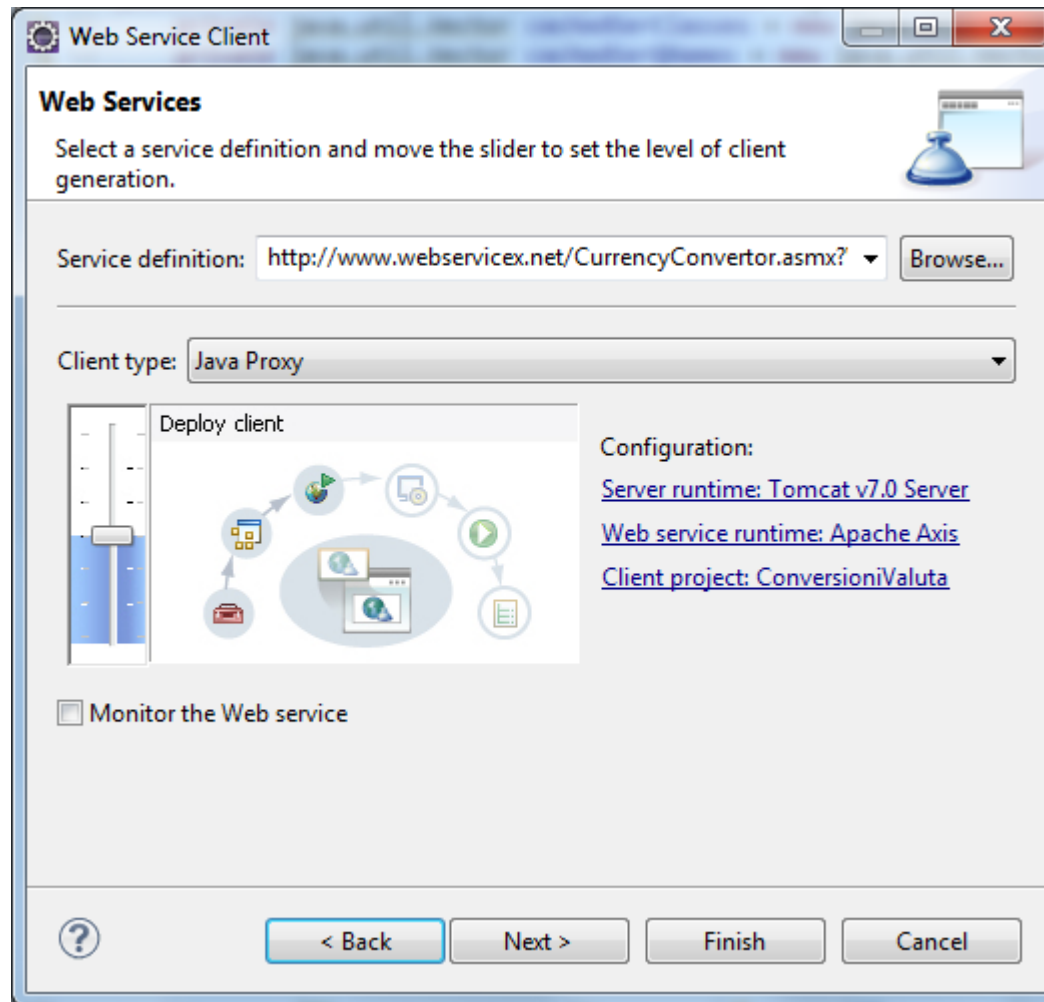
HTTP/1.1 200 OK

Content-Type: application/soap+xml; charset=utf-8

Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ConversionRateResponse xmlns="http://www.webserviceX.NET/">
      <ConversionRateResult>1.7321</ConversionRateResult>
    </ConversionRateResponse>
  </soap12:Body>
</soap12:Envelope>
```

# Creating the WS client



# Creation results

---

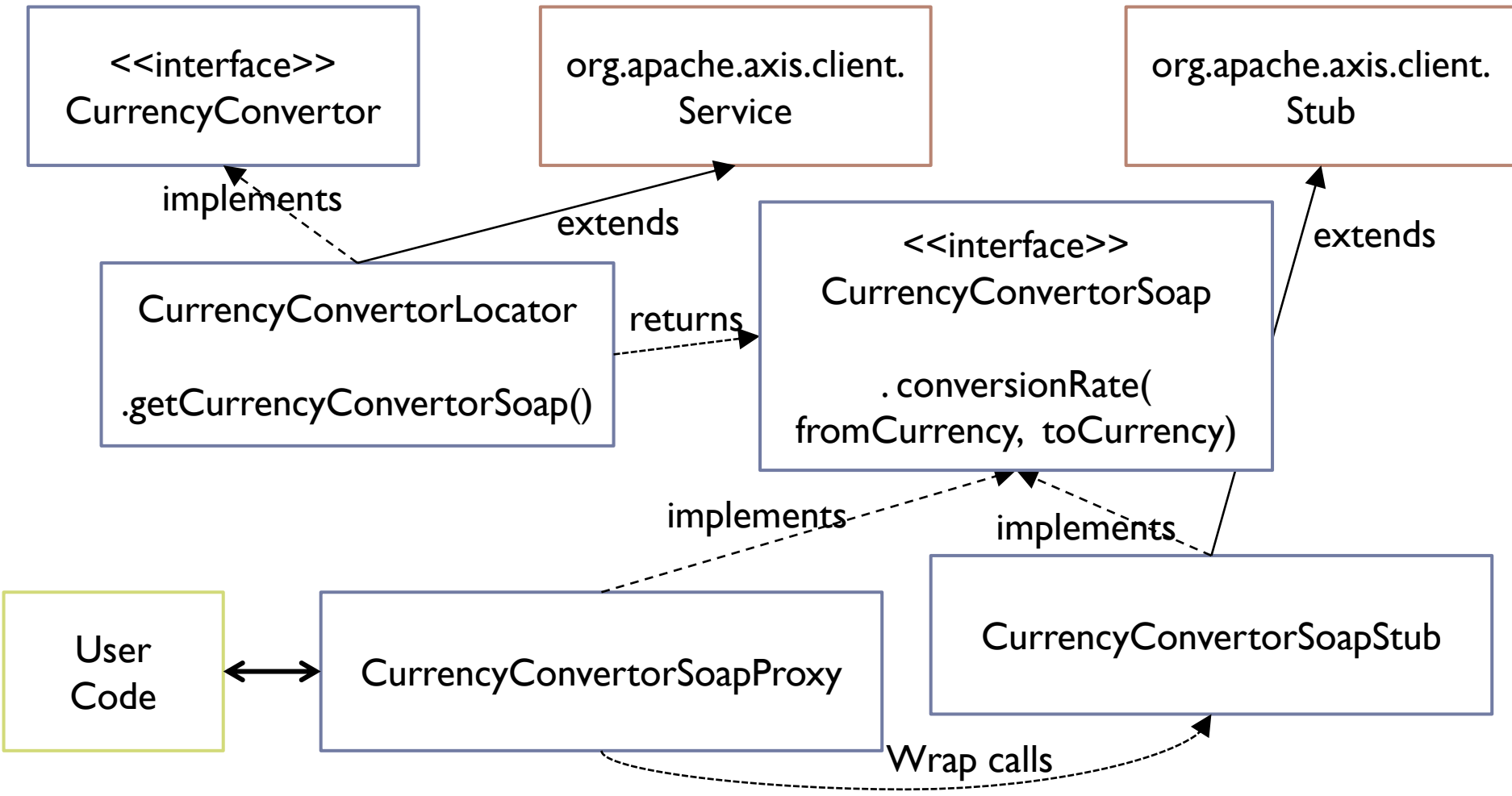
## Axis libraries in project

- WebContent
  - META-INF
  - WEB-INF
    - lib
      - axis.jar 136 21/11/11 17.27 fc
      - commons-discovery-0.2.jar 1
      - commons-logging.jar 136 21
      - jaxrpc.jar 136 21/11/11 17.27
      - saaj.jar 136 21/11/11 17.27 fc
      - wsdl4j.jar 136 21/11/11 17.27
      - web.xml 136 21/11/11 17.27 fcor

## Client proxy and methods

- Java Resources
  - src
    - it.polito.elite.sistinf
    - NET.webserviceX.www
      - Currency.java 136 21/11/11 17.27 f
      - CurrencyConvertor.java 136 21/11/
      - CurrencyConvertorLocator.java 136
      - CurrencyConvertorSoap.java 136 21
      - CurrencyConvertorSoapProxy.java 1
      - CurrencyConvertorSoapStub.java 13

# Generated classes



# User code example

---

WSDL Service

I

```
CurrencyConvertor locator =  
    new CurrencyConvertorLocator() ;
```

C

```
CurrencyConvertorSoap convertor =  
    locator.getCurrencyConvertorSoap() ;
```

WSDL Port

```
rate = convertor.conversionRate(fromC, toC) ;
```

WSDL Operation



# At the core of a web service

---

- ▶ The methods of any(\*) Java class may be wrapped as a Web Service
  - ▶ The remote call is handled by the Axis servlet
  - ▶ Service calls are translated to method calls
  - ▶ Input and output data types are converted
  - ▶ Method is executed
  - ▶ Exceptions are translated to Faults
- ▶ (\*) data type restrictions apply – see the JAX-RPC 1.1 Specification
  - ▶ <http://jcp.org/aboutjava/communityprocess/final/jsr101/index2.html>

# Data type conversions

---

- ▶ Primitive Java data types in the class methods must be converted to the “equivalent” XML data type
- ▶ Complex data types (collections, structures, graphs, ...) must be replicated using the XML data type constructors
  
- ▶ In an (1) automatic and (2) portable way

# Supported data types

---

- ▶ **Primitive data types**
  - ▶ `boolean`, `byte`, `short`, `int`, `long`, `float`, `double`
  - ▶ Also the corresponding “wrapper” classes
- ▶ **Java arrays, even multi-dimensional**
  - ▶ E.g., `int[]`, `String[]`, `String[][]`
- ▶ **Other specific classes:**
  - ▶ `java.lang.String`, `java.util.Date`,  
`java.util.Calendar`, `java.math.BigInteger`,  
`java.math.BigDecimal`, `javax.xml.namespace.Qname`,  
`java.net.URI`
- ▶ **In all other cases, you must define custom serializers and deserializers**

# Automatic translations (I)

---

Java Primitive Type	XML Data Type
boolean	xsd:boolean
byte	xsd:byte
short	xsd:short
int	xsd:int
long	xsd:long
float	xsd:float
double	xsd:double

Java Class	XML Data Type
java.lang.String	xsd:string
java.math.BigInteger	xsd:integer
java.math.BigDecimal	xsd:decimal
java.util.Calendar	xsd:dateTime
java.util.Date	xsd:dateTime
javax.xml.namespace.QName	xsd:QName
java.net.URI	xsd:anyURI

## Automatic translations (II)

---

```
<!-- Schema fragment -->
```

```
<element name="numbers"  
type="soapenc:Array"/>
```

```
// Java  
int[] numbers;
```

```
<!-- Schema instance -->
```

```
<numbers soapenc:arrayType="xsd:int[3]">
```

```
  <member>1</member>
```

```
  <member>2</member>
```

```
  <member>3</member>
```

```
</numbers>
```

# Requirements on Value Types

---

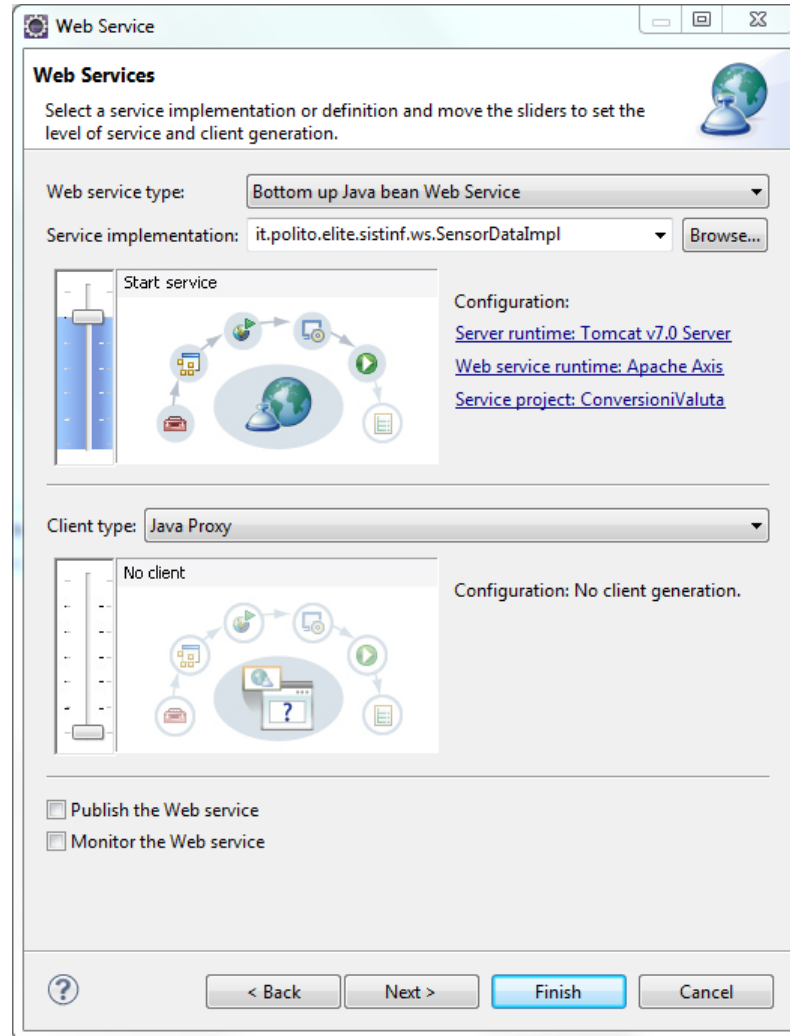
- ▶ Must have a public default constructor.
- ▶ Must not implement (directly or indirectly) the `java.rmi.Remote` interface.
- ▶ May implement any other Java interface or extend another class.
- ▶ May contain public, private, protected, package-level fields.
  - ▶ The Java type of a public field must be a supported JAX-RPC type
- ▶ May contain methods. No specified restrictions.
- ▶ May contain static or transient fields.
- ▶ May be designed as a JavaBeans class. In this case
  - ▶ Bean properties are required to follow the JavaBeans design pattern of setter and getter methods.
  - ▶ The Java type of a bean property must be a supported JAX-RPC type

# WSDL generation

---

- ▶ Java package -> WSDL document
- ▶ Service Endpoint Interface -> wsdl:portType
- ▶ Methods -> wsdl:operation
  - ▶ Name = method name
  - ▶ Method signature -> wsdl:input, wsdl:output, wsdl:fault
  - ▶ Parameters -> wsdl:message

# Creating the WS server



# Example

---

- ▶ Create a web service that publishes data from a set of configured sensors
- ▶ Two main methods
  - ▶ Get the list of all available sensors (a list of string IDs)
  - ▶ Given the ID of one sensor, get the last available data sample for that sensor
- ▶ Data sample: sensor ID + timestamp + value + unit of measure

# DataSample POJO

---

```
public class DataSample {
```

```
String sensorId ;
```

```
double value ;
```

```
Date timestamp ;
```

```
String unitOfMeasure ;
```

```
// constructors
```

```
// getters and setters
```

```
}
```

# Proposed Web Service interface

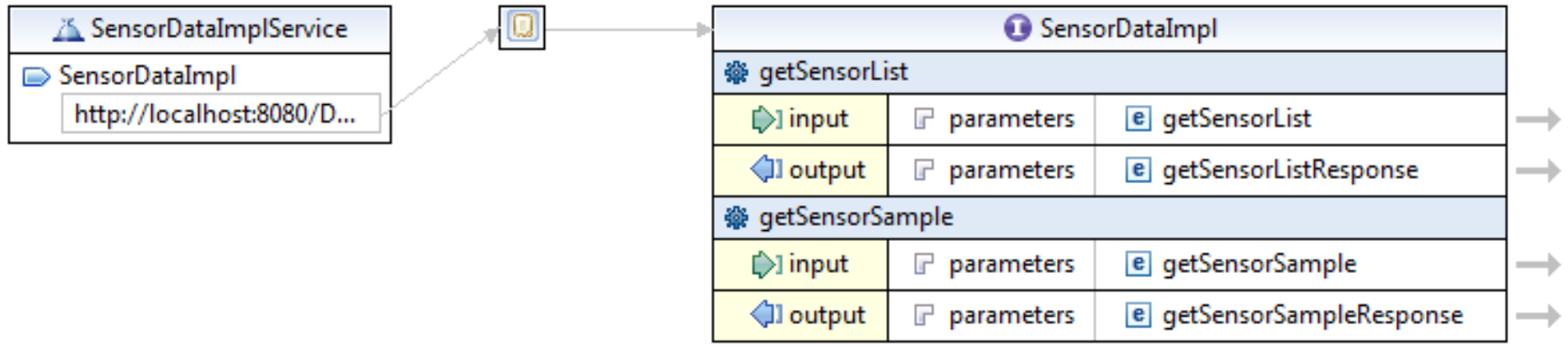
---

```
public interface SensorData extends java.rmi.Remote {
    /**
     * Return the list of available sensors to query
     *
     * @return A comma-separated list of IDs of all available sensors
     */
    public String getSensorList();

    /**
     * Return the last available sample for the specified sensor.
     *
     * @param sensorId
     *         the sensor to read
     * @return a {@link DataSample} POJO holding the last reading for the
     *         specified sensor
     */
    public DataSample getSensorSample(String sensorId);
}
```

# Generated WSDL

---



# Testing with Web Service Explorer

The screenshot displays the Eclipse IDE interface with the Web Services Explorer tool. The main window is titled "SensorData.java" and "Web Services Explorer". The left pane shows a tree view of the WSDL structure, including "WSDL Main", "file:/C:/Users/Fulvio/workspace/DatiSensori/WebContent/wsc...", "SensorDataImplService", "SensorDataImplSoapBinding", "getSensorSample", and "getSensorList".

The right pane is divided into two sections:

- Actions:** Contains the "Invoke a WSDL Operation" section. It prompts the user to "Enter the parameters for the WSDL operation 'getSensorSample' and click Go to invoke." The "Endpoints" field is set to "http://localhost:8080/DatiSensori/services/SensorDataImpl". The "Body" section shows the "getSensorSample" operation with a parameter "sensorId" of type "string" set to "a". "Go" and "Reset" buttons are visible.
- Status:** Shows the response body for the "getSensorSampleResponse" operation. The response is a "getSensorSampleReturn" object with the following values:
  - sensorId (string): a
  - timestamp (dateTime): 2011-11-29T21:55:20.817Z
  - unitOfMeasure (string): kW

# References

---

- ▶ **Axis Architecture Guide,**  
<http://axis.apache.org/axis/java/architecture-guide.html>
- ▶ **Currency Convertor Web Service,**  
<http://www.webservices.net/ws/wsdetails.aspx?wsid=10>
- ▶ **JAX-RPC specification v 1.1,**  
<http://jcp.org/en/jsr/summary?id=101>

# Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo 2.5 Italia (CC BY-NC-SA 2.5)”

- ▶ Sei libero:

- ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
- ▶ di modificare quest'opera



- ▶ Alle seguenti condizioni:

- ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
- ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali.
- ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.



- ▶ <http://creativecommons.org/licenses/by-nc-sa/2.5/it/>