

Web Services Tutorial

by Marlon Candido Guérios

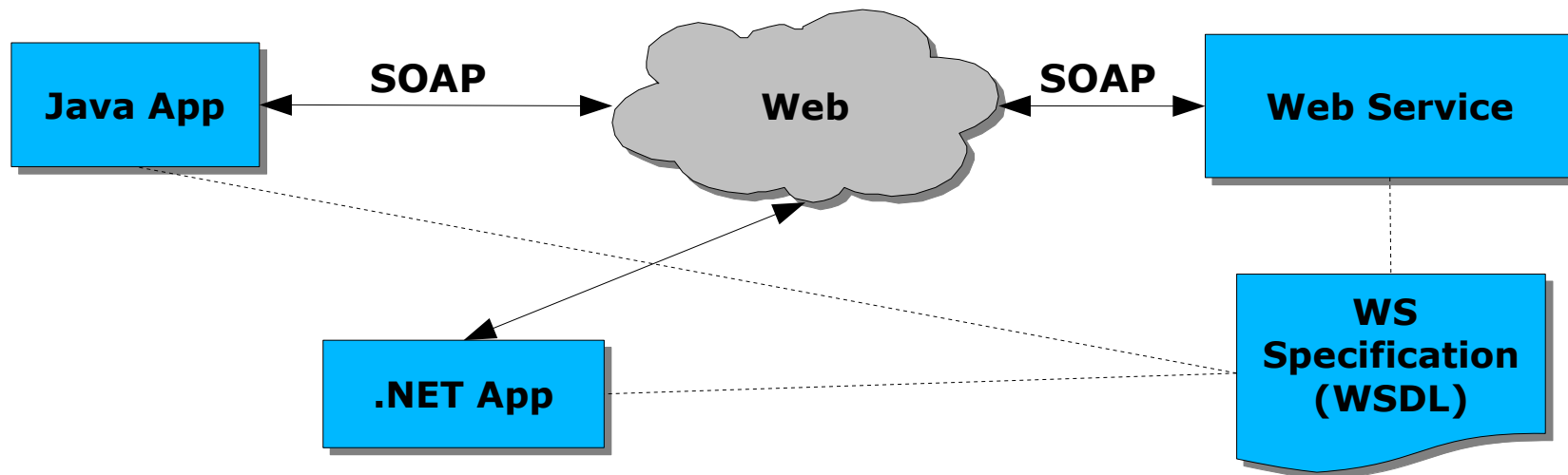
© Copyright 2006 - www.marlonguerios.com

Web Services. What's that?

- Available services to be remotely called by any kind of applications
- It doesn't matter how they are built
- Three things matter:
 - Where it is
 - How to call it
 - How is its response

Web Services. What's that?

- Web services have an address (URL)
- They're based on XML standards
- Web services have a specification
 - Detailing how to call it
 - Detailing how is its answer



SOAP - Simple Access Object Protocol

- SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment
- It is an XML based protocol that consists of three parts
 - an envelope that defines a framework for describing what is in a message and how to process it
 - a set of encoding rules for expressing instances of application-defined datatypes
 - a convention for representing remote procedure calls and responses

WSDL - Web Services Description Language

- WSDL serves to describe Web Services in a structured way
- A WSDL description of a service tells us, in a machine-understandable way, the interface to the service, the data types it uses, and where the service is located
- It's possible to build the binding Java classes just parsing the WSDL document for a specific Web Service

Building a Web Service - Step 1

- Define the service
 - A service is a well-defined task
 - It is independent from another service context or state (no coupling)
 - It exists by itself

Building a Web Service - Step 2

- Implement the service
 - It must be encapsulated in a stateless API
 - It is common to find services implemented in legacy systems that must be made available to other (and maybe newer) services. It just needs a new adapter layer
 - It doesn't matter the technology (Java, .Net, Cobol, etc.)
 - It exists by itself

Building a Web Service - Step 3

- Build the Web Service
 - API implementation
 - WSDL generation
 - Web Service deployment (HTTP/HTTPS)

Building a Web Service - Step 4

- Define the security level
 - There is no standards *de facto*
 - It's delegated to the transport layer
 - The access to the WS can be restricted to HTTPS protocol
 - Declarative security (J2EE standard)
 - URL-Pattern
 - Roles

Building a Web Service - Step 5

- Publishing the Web Service
 - This process is like any other web application
 - WSDL is published by the service itself
 - It is recommended to be stable, that is, once published, every update must be planned. The changes must not be sudden

Building a Web Service - Step 6

- Consume the Web Service
 - Access the public definition (WSDL)
 - Generate the binding classes from WSDL
 - Access the web services through just created binding classes

Apache AXIS

- As SOAP is just a protocol specification, an implementation is needed.
- Axis is essentially a SOAP implementation -- a framework for constructing SOAP processors such as clients, servers, gateways, etc.

Axis Instalation

- Download the Axis package
 - <http://ws.apache.org/axis>
- Unzip the /webapps/axis folder under Tomcat webapps folder
- Start tomcat
- Access <http://localhost:8080/axis>

Building a Web Service - Example

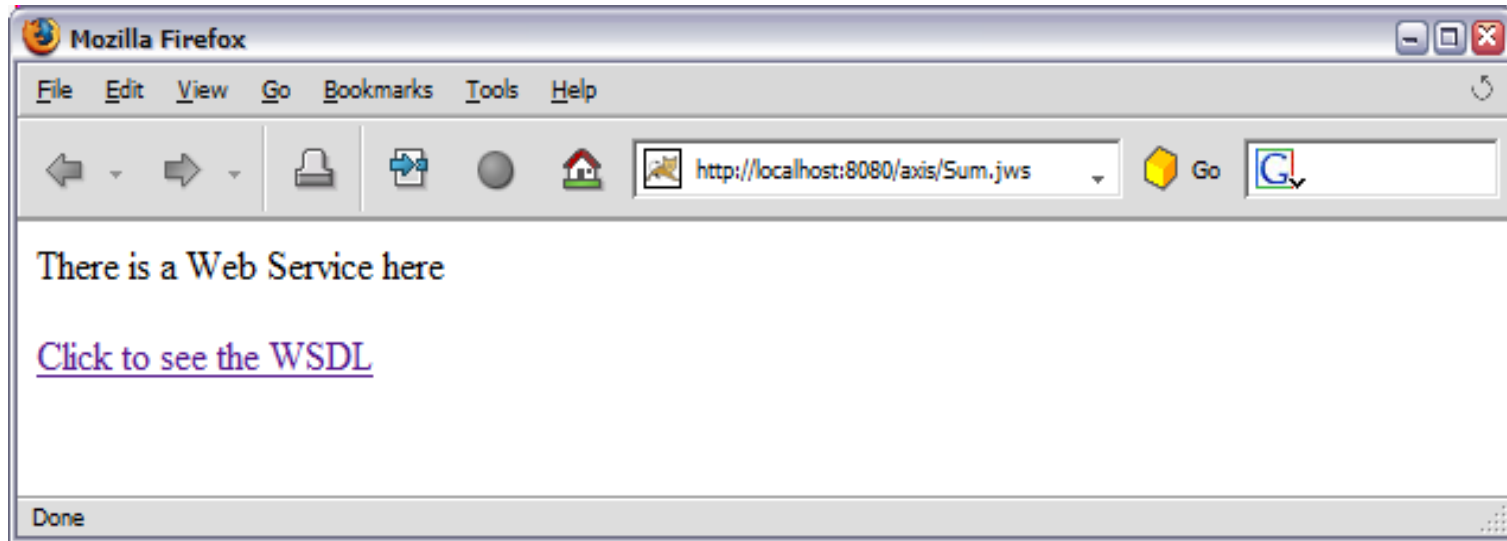
- Create the service

```
public class Sum {  
    public int calculate(int op1, int op2) {  
        return op1 + op2;  
    }  
}
```

- Copy Sum.java file to
<TOMCAT_DIR>/webapps/axis/Sum.jws

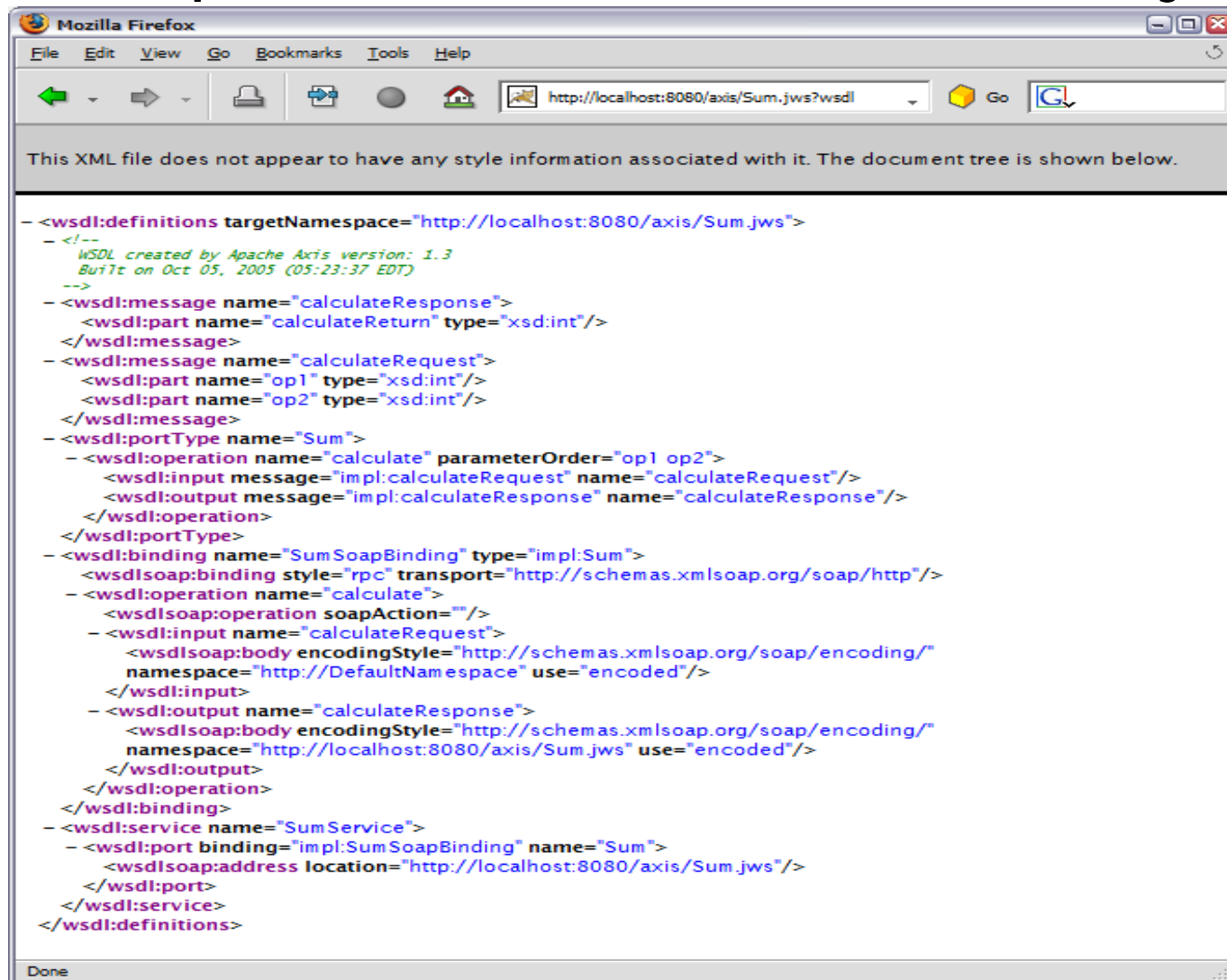
Building a Web Service - Example

- Start tomcat and access:
`http://localhost:8080/axis/Sum.jws`



Building a Web Service - Example

- Access: <http://localhost:8080/axis/Sum.jws?wsdl>



The screenshot shows a Mozilla Firefox browser window displaying the WSDL file for a Sum web service. The address bar shows the URL <http://localhost:8080/axis/Sum.jws?wsdl>. The main content area displays the XML WSDL document, which defines a service named "SumService" with a port named "Sum". The service has a single operation named "calculate" that takes two integer parameters and returns an integer. The WSDL is as follows:

```
- <wsdl:definitions targetNamespace="http://localhost:8080/axis/Sum.jws">
  - <!--
    WSDL created by Apache Axis version: 1.3
    Built on Oct 05, 2005 (05:23:37 EDT)
  -->
  - <wsdl:message name="calculateResponse">
    <wsdl:part name="calculateReturn" type="xsd:int"/>
  </wsdl:message>
  - <wsdl:message name="calculateRequest">
    <wsdl:part name="op1" type="xsd:int"/>
    <wsdl:part name="op2" type="xsd:int"/>
  </wsdl:message>
  - <wsdl:portType name="Sum">
    - <wsdl:operation name="calculate" parameterOrder="op1 op2">
      <wsdl:input message="impl:calculateRequest" name="calculateRequest"/>
      <wsdl:output message="impl:calculateResponse" name="calculateResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  - <wsdl:binding name="SumSoapBinding" type="impl:Sum">
    <wsdl:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    - <wsdl:operation name="calculate">
      <wsdl:soap:operation soapAction=""/>
      - <wsdl:input name="calculateRequest">
        <wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://DefaultNamespace" use="encoded"/>
      </wsdl:input>
      - <wsdl:output name="calculateResponse">
        <wsdl:soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://localhost:8080/axis/Sum.jws" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  - <wsdl:service name="SumService">
    - <wsdl:port binding="impl:SumSoapBinding" name="Sum">
      <wsdl:soap:address location="http://localhost:8080/axis/Sum.jws"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Building a Web Service - Example

- It is possible to generate WSDL from source code using Java2WSDL tool

```
java org.apache.axis.wsdl.Java2WSDL -o Sum.wsdl  
-l"http://localhost:8080/axis/Sum.jws"  
-n "urn:Sum" -p"com.marlongueros.webservices.example1"  
"urn:Sum" samples.userguide.example6.WidgetPrice
```

Building a Web Service - Example

- WSDL

Return type

```
- <wsdl:message name="calculateResponse">  
  <wsdl:part name="calculateReturn" type="xsd:int"/>  
</wsdl:message>
```

Parameters

```
- <wsdl:message name="calculateRequest">  
  <wsdl:part name="op1" type="xsd:int"/>  
  <wsdl:part name="op2" type="xsd:int"/>  
</wsdl:message>
```

Method

```
- <wsdl:portType name="Sum">  
  - <wsdl:operation name="calculate" parameterOrder="op1 op2">  
    <wsdl:input message="impl:calculateRequest" name="calculateRequest"/>  
    <wsdl:output message="impl:calculateResponse" name="calculateResponse"/>  
  </wsdl:operation>  
</wsdl:portType>
```

Building a Web Service - Example

- WSDL

Binding details

```
- <wsdl:binding name="SumSoapBinding" type="impl:Sum">  
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>  
  - <wsdl:operation name="calculate">  
    <wsdlsoap:operation soapAction=""/>  
    - <wsdl:input name="calculateRequest">  
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
        namespace="http://DefaultNamespace" use="encoded"/>  
    </wsdl:input>  
    - <wsdl:output name="calculateResponse">  
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
        namespace="http://localhost:8080/axis/Sum.jws" use="encoded"/>  
    </wsdl:output>  
  </wsdl:operation>  
</wsdl:binding>
```

Address

```
- <wsdl:service name="SumService">  
  - <wsdl:port binding="impl:SumSoapBinding" name="Sum">  
    <wsdlsoap:address location="http://localhost:8080/axis/Sum.jws"/>  
  </wsdl:port>
```

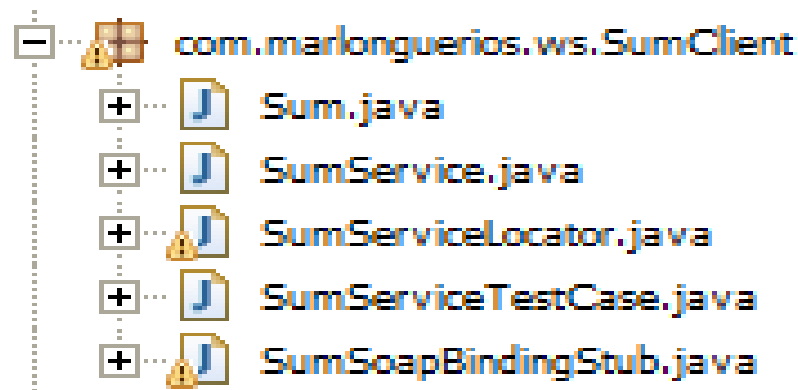
Building a Web Service - Example

- Generate client classes

Main class:

Program arguments:

- Generated classes



Building a Web Service - Example

Binded Interface

⊕ * Sum_PortType.java

```
package com.marlonguerios.ws.SumClient;

public interface Sum extends java.rmi.Remote {
    public int calculate(int op1, int op2) throws java.rmi.RemoteException;
}
```

Service Interface

⊕ * SumService.java

```
package com.marlonguerios.ws.SumClient;

public interface SumService extends javax.xml.rpc.Service {
    public java.lang.String getSumAddress();

    public com.marlonguerios.ws.SumClient.Sum getSum() throws javax.xml.rpc.ServiceException;

    public com.marlonguerios.ws.SumClient.Sum getSum(java.net.URL portAddress) throws javax.xml.rpc.ServiceException;
}
```

Building a Web Service - Example

Service Locator

```
* SumServiceLocator.java
package com.marlonguerios.ws.SumClient;

public class SumServiceLocator extends org.apache.axis.client.Service implements com.marlonguerios.ws.SumClient.SumService {

    public SumServiceLocator() {
    }

    public SumServiceLocator(org.apache.axis.EngineConfiguration config) {
        super(config);
    }

    public SumServiceLocator(java.lang.String wsdlLoc, javax.xml.namespace.QName sName) throws javax.xml.rpc.ServiceException {
        super(wsdlLoc, sName);
    }

    // Use to get a proxy class for Sum
    private java.lang.String Sum_address = "http://localhost:8080/axis/Sum.jws";

    public java.lang.String getSumAddress() {
        return Sum_address;
    }

    // The WSDO service name defaults to the port name.
    private java.lang.String SumWSDOServiceName = "Sum";

    public java.lang.String getSumWSDOServiceName() {
        return SumWSDOServiceName;
    }
}
```

Building a Web Service - Example

Binded Implementation (invoke the method)

```
+ * SumSoapBindingStub.java
package com.marlonguerios.ws.SumClient;

public class SumSoapBindingStub extends org.apache.axis.client.Stub implements com.marlonguerios.ws.SumClient.Sum {
    private java.util.Vector cachedSerClasses = new java.util.Vector();
    private java.util.Vector cachedSerQNames = new java.util.Vector();
    private java.util.Vector cachedSerFactories = new java.util.Vector();
    private java.util.Vector cachedDeserFactories = new java.util.Vector();

    static org.apache.axis.description.OperationDesc [] _operations;

    static {
        _operations = new org.apache.axis.description.OperationDesc[1];
        _initOperationDesc1();
    }

    private static void _initOperationDesc1(){
    }

    public SumSoapBindingStub() throws org.apache.axis.AxisFault {}

    public SumSoapBindingStub(java.net.URL endpointURL, javax.xml.rpc.Service service) throws org.apache.axis.AxisFault {}

    public SumSoapBindingStub(javax.xml.rpc.Service service) throws org.apache.axis.AxisFault {}

    protected org.apache.axis.client.Call createCall() throws java.rmi.RemoteException {}

    public int calculate(int op1, int op2) throws java.rmi.RemoteException {}
}
```

Building a Web Service - Example

Test Case 1

```
public void test1SumCalculate() throws Exception {
    com.marlonguerios.ws.SumClient.SumSoapBindingStub binding;
    try {
        binding = (com.marlonguerios.ws.SumClient.SumSoapBindingStub)
            new com.marlonguerios.ws.SumClient.SumServiceLocator().getSum();
    }
    catch (javax.xml.rpc.ServiceException jre) {
        if(jre.getLinkedCause()!=null)
            jre.getLinkedCause().printStackTrace();
        throw new junit.framework.AssertionFailedError("JAX-RPC ServiceException caught: " + jre);
    }
    assertNotNull("binding is null", binding);

    // Time out after a minute
    binding.setTimeout(60000);

    // Test operation
    int value = -3;
    value = binding.calculate(2, 7);
    // TBD - validate results

    System.out.println("Sum result: " + value);
}
```

Building a Web Service - Example

Test Case 2

```
public void test2SumCalculate() throws Exception {  
    SumService service = new SumServiceLocator();  
    Sum sum = service.getSum(new URL("http://localhost:8080/axis/services/Sum"));  
    System.out.println("Remote sum: " + sum.calculate(5, 7));  
}
```

Deploying a Web Service

Create a deploy.wsdd file

```
1 <deployment xmlns="http://xml.apache.org/axis/wsdd/"
2     xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
3
4     <service name="Sum" provider="java:RPC">
5         <parameter name="className" value="Sum"/>
6         <parameter name="allowedMethods" value="*" />
7     </service>
8 </deployment>
```

- Copy the service related classes to WEB-INF/classes folder under axis web application
- Run the Axis admin tool to deploy the service:
 - `org.apache.axis.client.AdminClient deploy.wsdd`
Processing file deploy.wsdd
<Admin>Done processing</Admin>
- The service address now is:
<http://localhost:8080/axis/services/Sum>

Reference

<http://ws.apache.org/axis>