

Overview

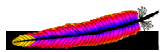
1. Introduction to Web Services
2. Overview of Web Services architecture
3. [Web services platforms \(Java, .Net\) "Hallo World" tutorials for Java and .Net](#)
4. Key Web Services technologies: XML schemata and namespaces, SOAP, WSDL, and UDDI
5. Interoperability
6. Web Services Orchestration
7. Web services security

1

My Java "Hallo World" Web Service

- Simple hands-on introduction to
 - Java Web Services: available libraries
 - Apache – Tomcat – Axis: server concept
- Examples
 - Write some simple test Web Services and Clients
 - Write a Web Service that is a client to the Google Web Service

2



Apache HTTP Server

<http://httpd.apache.org>

- Apache HTTP Server: open-source HTTP server
- Secure, efficient, extensible, standard, open source HTTP server
- Working for UNIX and Windows.
- [Basic transport protocol](#)

3



Tomcat Application Server

<http://jakarta.apache.org>

- Tomcat application server
- Reference implementation for
 - Java Servlet
 - Java Server Pages
- Allows to access Java applications provided via an (apache) HTTP server

4



Axis Libraries

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

- Axis is a SOAP engine: a framework for constructing SOAP documents on clients and servers
- Support for WSDL
 - Generate WSDL out of Java interfaces
 - Generate Subs and Skeletons out of WSDL
- [Used by application servers](#), e.g., Tomcat

5



Java Libraries

<http://java.sun.com/webservices>

- Document processing
 - Java API for XML Processing (JAXP) processes XML documents using various parsers
 - Java Architecture for XML Binding (JAXB) maps Java objects to XML and vice versa
- Remote computation
 - Java API for XML-based RPC (JAX-RPC) using SOAP with Attachments API (SAAJ) to sends SOAP method calls to remote parties and receives the results
 - Java API for XML Registries (JAXR) provides a standard way to access business registries and share information

6

API for XML processing JAXP

We need to parse and transform XML:

- XML parsers with different interface levels: SAX-event and DOM-tree levels
- Document Object Model (DOM) implementation itself
- XSLT processing of XML translation

7

API for XML Binding (JAXB)

- In Web Services, there is no standard binding between data types in a language (e.g. Java) and XML Schema
- Best practices and guarantee for interoperability due to WS-Interoperability organization
 - JAXB as used by default follows this recommendation
- Special closed world applications could benefit from tailored mappings
 - JAXB additionally provides the framework to define special mappings

8

API for XML-based RPC (JAX-RPC)

- Default serialization / deserialization of Java data types
- Remote Procedure Call (RPC) to endpoint addresses
- Support for WS-I
 - Basic Profile 1.1
 - Attachment Profile 1.0
 - Simple SOAP Binding Profile 1.0

9

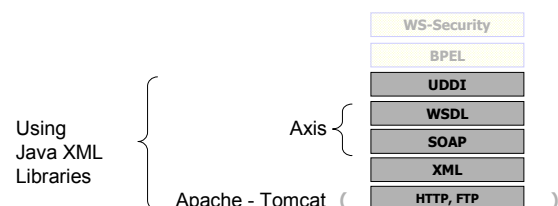
SOAP with Attachments API (SAAJ)

- Used by JAX-RPC and using itself JAXP
- Writing SOAP messages directly
- SOAP is a messaging and processing framework that goes beyond its use in Web Services.

10

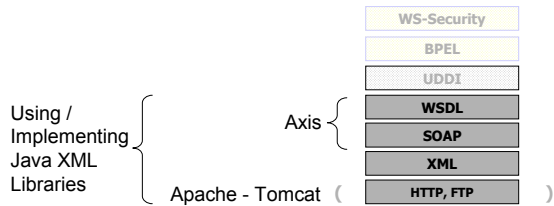
API for XML Registries (JAXR)

- Access to UDDI registries
- Access to ebXML registries not supported. (More specifically: JAXR supports Version 2 UDDI registries but not to Version 1 UDDI registries.)



11

12



13

Simple Web Service

```
public class TestWS {
    public String testMethod(String in){
        return in + "?";
    }
}
```

14

Simple Web Client

```
import ...
public class TestClient {
    static void main(String [] args) throws Exception {
        String endpoint =
            "http://localhost:80/axis/TestWS.jws";
        Service service = new Service();
        Call call = (Call) service.createCall();
        call.setTargetEndpointAddress(new URL(endpoint));
        call.setOperationName("testMethod");
        call.addParameter("in",
            XMLType.XSD_STRING, ParameterMode.IN);
        call.setReturnType( XMLType.XSD_STRING );
        String ret=(String) call.invoke(new Object[]{"test"});
        System.out.println("Got result: " + ret);
    }
}
```

JAX-RPC Objects

Actual Call

15

Simple Web Service deployment

```
% ftp <localhost>/TestWS.java
<webapp>/TestWS_<login>.jws
```

16

Compile and Start the Client

```
% javac -classpath "..." TestClient.java
% java -classpath "..." TestClient

% Got result : A test client ?
```

17



- Create a handle for the server and a request object
 - Set target WS
 - Set parameter types
 - Set actual parameter
- Call
 - Serializes request to SOAP
 - Deserializes the SOAP reply to a Java object
- Gets SOAP request
- Possibly compiles/loads the requested WS class and creates object of that class
- Executes the method on the actual parameters and computes the result (if any)
- Returns the SOAP reply

18

Java Web Services (JWS)

Pros:

- Simple to create
- Simple to deploy
- Simple to start with

Cons:

- No user defined binding of service name and implementation class
- No WSDL
 - Hard to publish
 - Hard to administrate
 - No remote transparency

19

SOAP Monitor

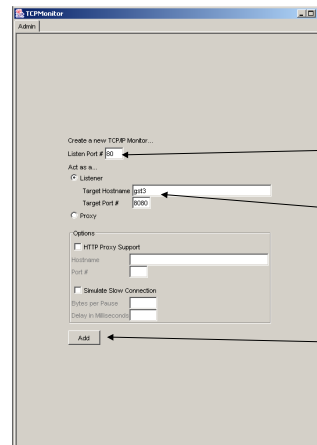
- Allows you to check for SOAP messages exchanged between client and server
- Works like a proxy:
 - Client calls localhost's monitor with the SOAP request
 - Monitor displays the SOAP request and forwards it to the actual server
 - SOAP reply gets displayed and forwarded to client

20

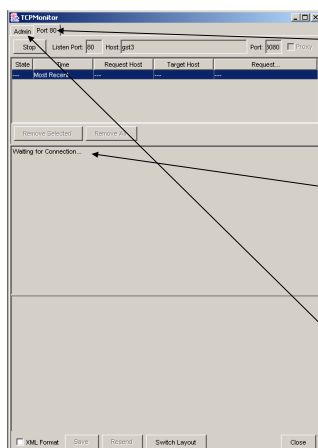
Start the SOAP Monitor

```
% java -classpath "." org.apache.axis.utils.tcpmon
% TCPmonitor.bat
```

21



- New call target
Unique new Port:
e.g.
http://localhost:1234/axis/TestWS.jws
- Original call target
Target Host and Port:
e.g.
http://localhost:8180/axis/TestWS.jws
- Add monitor



- New monitor ...
- ... waits for messages
- One could actually run several monitors at the same time

Change Simple Web Client

```
import ...
public class TestClient {
    public static void main(String [] args) throws Exception {
        String endpoint =
            "http://localhost:1234/axis/TestWS.jws";
        Service service = new Service();
        Call call = (Call) service.createCall();
        call.setTargetEndpointAddress(new URL(endpoint));
        call.setOperationName("testMethod");
        call.addParameter("in",
            XMLType.XSD_STRING, ParameterMode.IN);
        call.setReturnType( XMLType.XSD_STRING );
        String ret=(String) call.invoke(new Object[]{"test"});
        System.out.println("Got result: " + ret);
    }
}
```

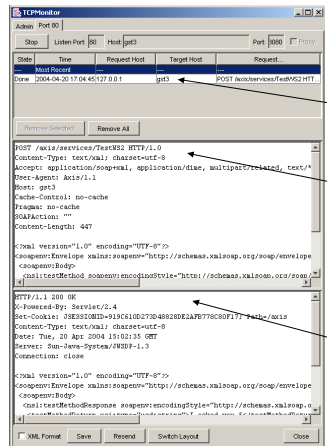
24

Re-compile and Re-start the Client

```
% javac -classpath "..." TestClient.java
% java -classpath "..." TestClient

% Got result : A test client ?
```

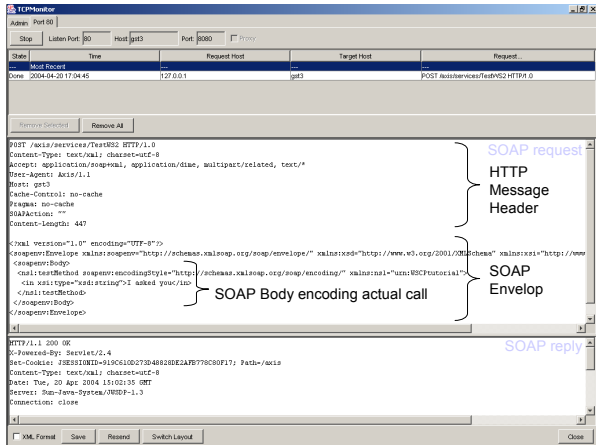
25



Recent calls

SOAP request

SOAP reply



Custom Deployment – WSDD

- Web Services Deployment Descriptor
 - XML format
 - Axis specific, **Not** W3C/Web Services standard !
- Specifies
 - Services to (un-)deploy, i.e. (de-)activate
 - Mapping of logic Web Service name to
 - Implementation classes
 - Runtime objects thereof in a session
 - Further administration information
 - ...

28

A Deployment Descriptor

```
<deployment
  xmlns="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="TestWS2" provider="java:RPC">
    <parameter name="className" value="test.TestWS2"/>
    <parameter name="allowedMethods" value="*" />
    <parameter name="scope" value="Request" />
  </service>
</deployment>
```

WS to implementation mapping

WS to object mapping
Session | Request | Application

29

An Undeployment Descriptor

```
<undeployment
  xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="TestWS2"/>
</undeployment>
```

30

Deploying / Undeploying

```
% java -classpath "..." org.apache.axis.client.AdminClient
-h<host>
-p<port>
-u<user>
-w<password>
<deployment file>.wsdd

% java -classpath "..." org.apache.axis.client.AdminClient
-h
...
<undeployment file>.wsdd
```

31

Another Web Service

```
package test;

public class TestWS2 {
    private int count = 0;
    public String testMethod(String in){
        return in + "(calls: " + count++ + ")";
    }
}
```

32

Another Web Client

```
import ...
public class TestClient2 {
    public static void main(String[] arg throws Exception {
        String endpoint =
            "http://localhost:8180/axis/services/TestWS2";
        String serviceClass = "TestWS2";
        String method = "testMethod";
        /* basically before */
        ...
        call = SOAP monitor (new QName(serviceClass, testMethod));
        ...
        String ret = (String) call.invoke( newObject[]{"test"} );
        System.out.println("Got result: " + ret);
    }
}
```

Default WS location

Still use SOAP monitor

33

Compile and deploy

```
% javac test/*.java
% java -classpath "..." org.apache.axis.client.AdminClient
-hlocalhost
-p8180
-uuser
-wPassword test/deploy.wsdd
% ftp <local>/test/TestWS2_<login>.class
<webapp>/WEB-INF/classes/test/TestWS2.class

% ./deploy.bat
% ftp <local>/test/TestWS2.class
<webapp>/WEB-INF/classes/test/TestWS2.class
```

34

Start the Client

```
% java -classpath "..." test.TestClient2
% Got result 2: tested (calls: 0)
```

35



- Gets SOAP request
- According to WSDD
 - Maps logic WS name to WS class
 - Creates object
- Executes the method on the actual parameters and computes the result (if any)
- Returns the SOAP reply
- Basically as before
- Create a handle for the server and a request object
 - Set target WS
 - Set parameter types
 - Set actual parameter
- Call
 - Serializes request to SOAP
 - Deserializes the SOAP reply to a Java object

36

Web Services Classes – deployment using WSDD

Pros:

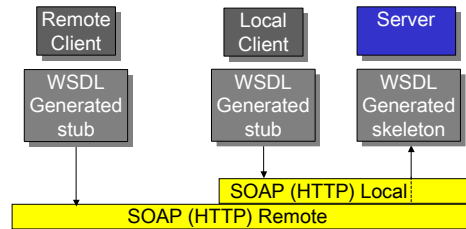
- Still pretty simple to create, deploy
- User defined binding of service name and implementation class

Cons:

- No WSDL subs & skeletons: **No remote transparency**

37

Remote transparency: WSDL



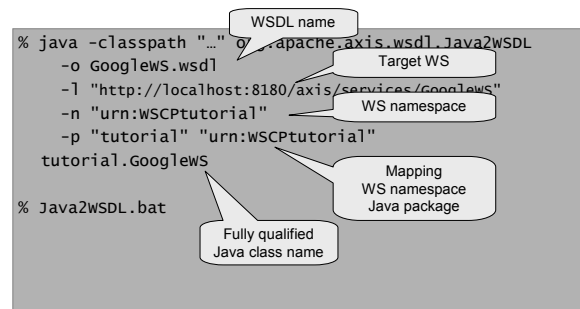
38

Generate WSDL: Java2WSDL

- Web Services Description Language (WSDL) corresponds to an interface of a Web Service
- Information can be extracted
- WSDL encoding can be generated

39

Java2WSDL Example



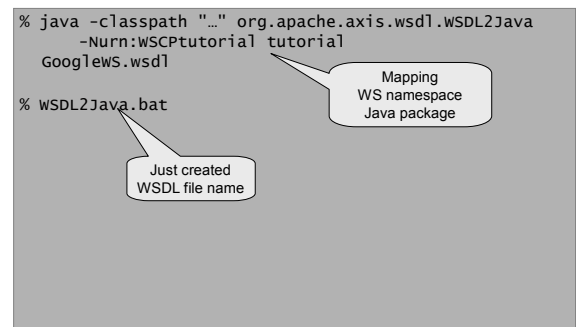
40

Generate Stubs: WSDL2Java

- Stub classes from a Java interface/class *:
 - *.java: New interface file with appropriate remote usages.
 - *SoapBindingImpl.java: default server implementation WS, **modify manually to actual implementation.**
 - *Service.java: client side service interface.
 - *ServiceLocator.java: client side service implementation factory.
 - *SoapBindingSkeleton.java: Server side skeleton.
 - *SoapBindingStub.java: Client side stub.
 - (data types): Java files produced for non-standard types.
- deploy.wsdd / undeploy.wsdd: (Un-)deployment descriptors

41

WSDL2Java Example



42

Web Service deployment

```
% javac tutorial/ *.java
% java -classpath "..."
org.apache.axis.client.AdminClient
-hlocalhost -p8180 -uUser -wPassword
tutorial/deploy.wsdd
% ftp <local>/tutorial/*.class
<webapp>/WEB-INF/classes/tutorial/*.class

% deployGoogle.bat
% ftp <local>/tutorial/*.class
<webapp>/WEB-INF/classes/tutorial/*.class
```

43

Compile and Start the Client

```
% javac -classpath "..." GoogleClient.java
% java -classpath "..." GoogleClient search WSCC

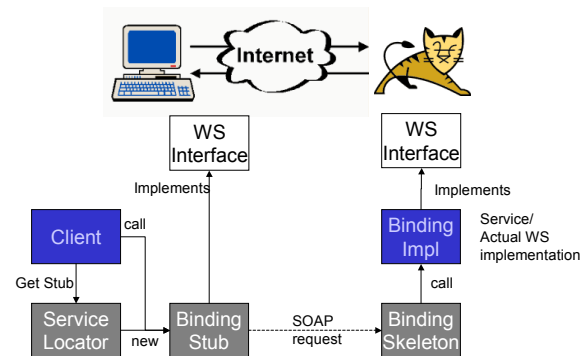
% Answer: ...
[
  URL = "http://wscc.info/"
  Title = "This site was created using SiteDirect from wipmind <b>...</b>"
  Snippet = "This site was created using SiteDirect from wipmind
webpublicering webdesign web-br> content management publiceringsverktog wcm
e-handel e-business e-shop webshop <b>...</b> "
  Directory Category = {SE="", FVW=""}
  Directory Title = ""
  Summary = ""
  Cached Size = "2k"
  Related information present = true
  Host Name = ""
], ...
```

44



- Create a locator object
- Get a stub
- Call stub as if it was a local service
- Stub manages the SOAP request to / reply from Skeleton
- Skeleton gets SOAP request
- Calls WS impl as if it as a local client
- WS returns to skeleton
- Skeleton manages the SOAP reply to Stub

45



46

Web Services – generated using WSDL

Pros:

- Easy to create, deploy
- User defined binding
- Can be published
- Remote/Language transparency
- Static invocation is faster

Cons:

- Static invocation is less flexible
- More complex

47

Resources

- Java Web Services Tutorial
<http://java.sun.com/webservices/docs/1.1/tutorial/doc>
 - Great for the XML APIs
 - Refers to a complete installation
 - Leaves out Axis
- Axis User's Guide:
<http://ws.apache.org/axis/java/user-guide.html>
 - Great for getting hands on Axis
 - Assumes some knowledge on XML APIs
- Standards: <http://www.w3c.org>

48