

Java web services using JAX-WS

11, 12 Dec. 2009

Pune, India

IndicThreads.com Conference On Java Technology

Speaker Name Lalit Mohan Chandra Bhatt

Company Name Crayom Engineering Services

Scope

- Understanding how JAX-WS can be used to implement SOAP based web services both at server and client side.



Are webservices hard !

Dave Podnar's Five Stages of Dealing with Web Services

- Denial - It's Simple Object Access Protocol, right?
- Over Involvement - OK, I'll read the SOAP, WSDL, WS-I BP, JAX-RPC, SAAJ, JAX-P... specs. next, I'll check the Wiki and finally follow an example showing service and client sides.
- Anger - I can't believe those #\$\$%&*@s made it so difficult!
- Guilt - Everyone is using Web Services, it must be me, I must be missing something.
- Acceptance - It is what it is, Web Services aren't simple or easy

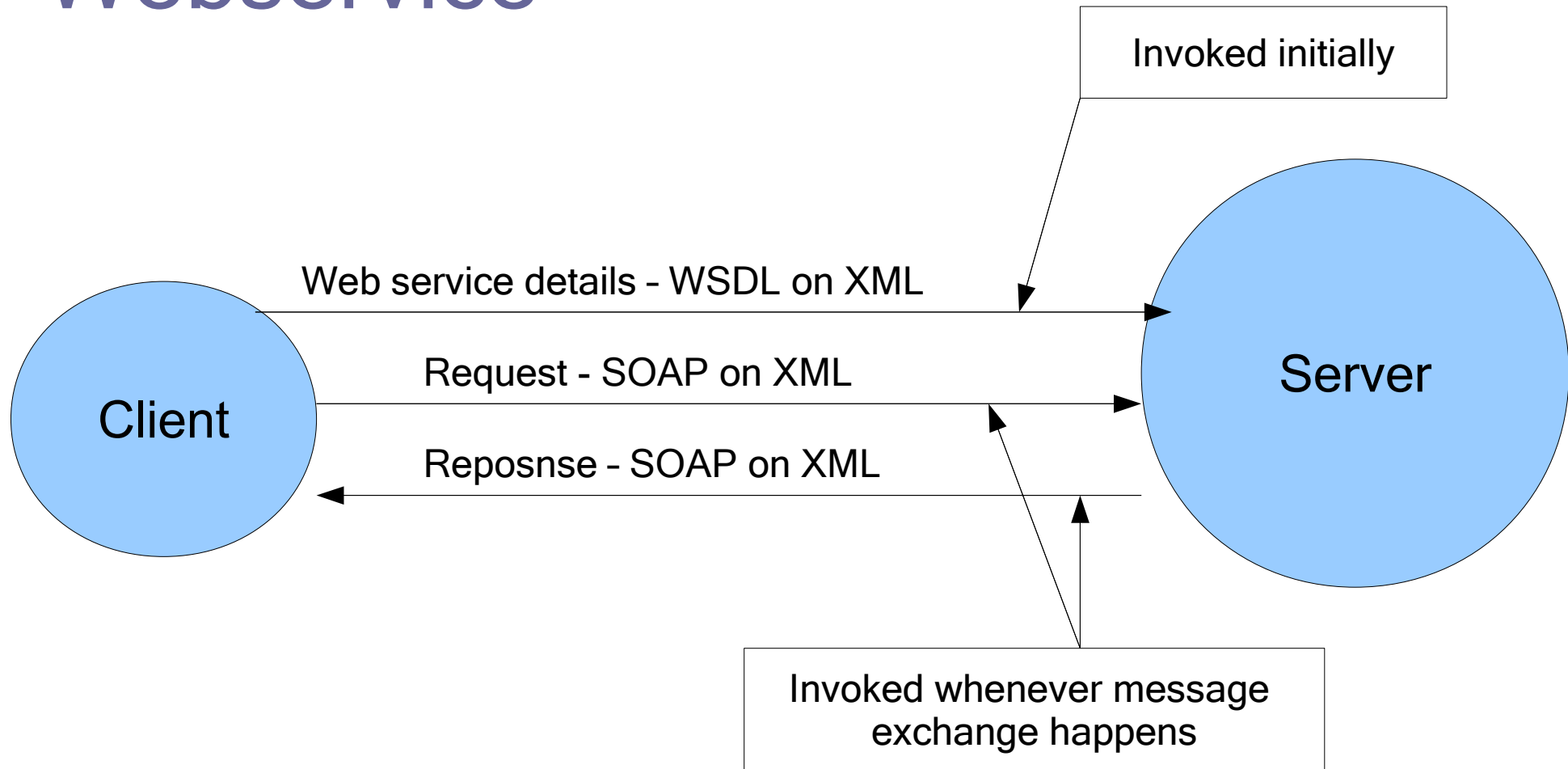


Jargons Jargons

XML XSD
XSLT Xpath JAXP
SAX DOM JAXB StaX
SOAP WSDL UDDI
JAX-RPC JAX-WS JAX-RS
SAAJ WS* BP
Axis Metro
ESB
SOA



Webservice



Webservice - JAX-WS way

- Plain old Java Object (POJO) can be easily exposed as web service.
- Annotation driven
- Data binding through JAXB
- Server Independent



JAX-WS

Demo



JAX-WS - Servlet Way

- Write the class
- Annotate
- Register in web.xml
- Deploy - The server runtime will
 - Generate and publish WSDL.
 - Map SOAP request to a Java method invocation.
 - Translate method return into a SOAP response.



JAX-WS - Servlet Way

@WebService

```
public class TemperatureConverter {
```

@WebMethod

```
public double celsiusToFahrenheit(double temp){
```

```
    ...
```

```
}
```

@WebMethod

```
public double fahrenheitToCelsius(double temp){
```

```
    ...
```

```
}
```

```
}
```



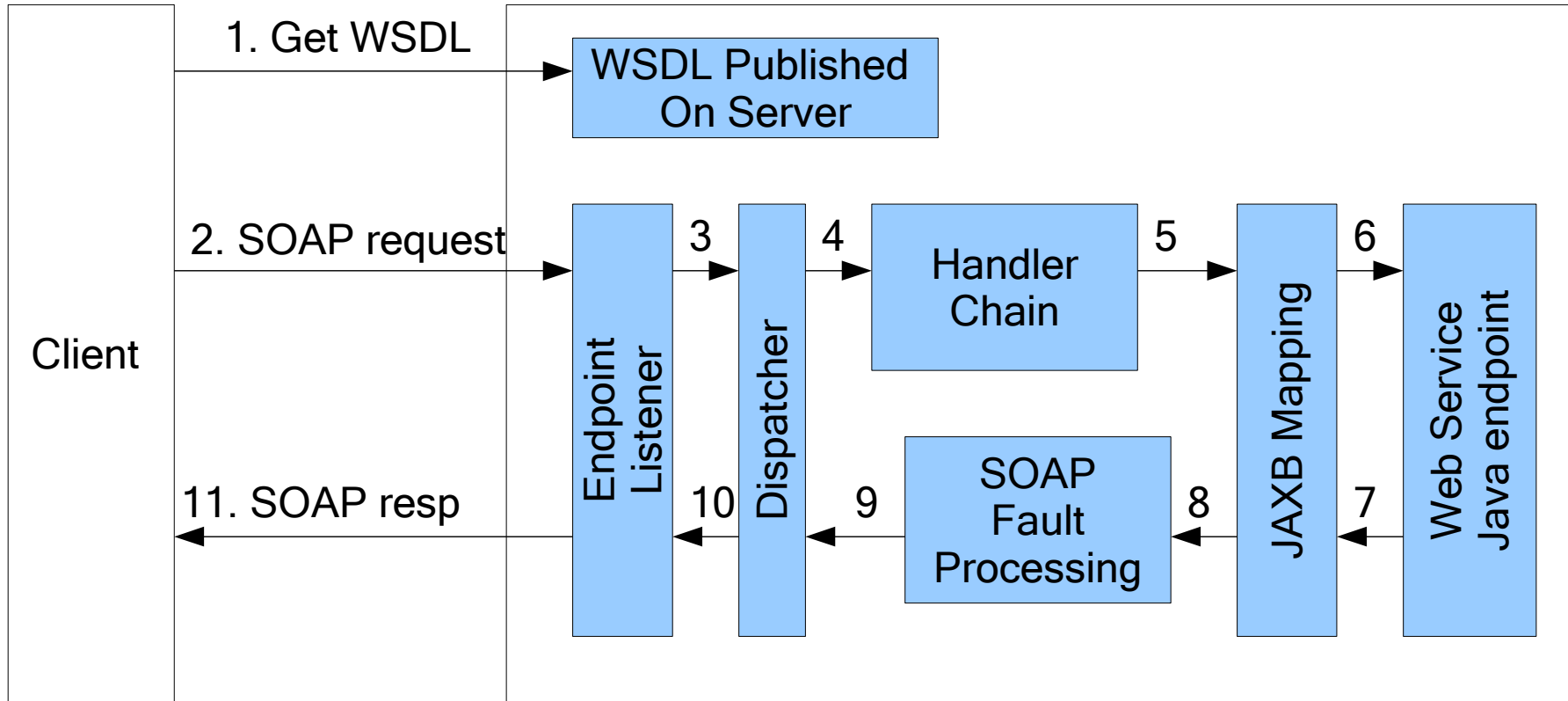
JAX-WS - Servlet Way

```
<servlet>  
  <servlet-name>tempConv</servlet-name>  
  <servlet-class>  
    com.lalit.TemperatureConverter  
  </servlet-class>  
</servlet>
```

```
<servlet-mapping>  
  <servlet-name>tempConv</servlet-name>  
  <url-pattern>/tempConv</url-pattern>  
</servlet-mapping>
```



JAX-WS Servlet Way



JAX-WS

EJB 3.0 way

- Annotate
- Deploy ejb jar



JAX-WS - EJB 3.0 way

`@Stateless`

`@WebService`

```
public class TemperatureConverter {
```

```
//Rest code remains same.
```



JAX-WS - JavaSE Endpoint

- Starting Java 6
- Generate the artifact using wsgen tool.
- wsgen tool generates JAXB mapped classes
- Use embedded HttpServer to deploy the webservice



JAX-WS - JavaSE Endpoint

```
public static void main(String[] args) {  
    TemperatureConverter tc= new  
        TemperatureConverter();  
  
    //Java comes with an embedded Http server  
    //which is used to host the service  
    Endpoint endpoint = Endpoint.publish  
("http://localhost:8080/tempConv", tc);  
  
    //Keeping commented, keeps the server running  
    /*endpoint.stop()*/  
}  
}
```



JAX-WS - Client side

- Generate artifact using wsimport pointing to WSDL
- wsimport generates JAXB binding classes and service endpoint
- Call the web service using service end point



JAX-WS - Client side

```
//Make the instance of service class
TemperatureConverterService service =
    new TemperatureConverterService();

//Get port to invoke webservice
TemperatureConverter port =
    service.getPort
        (TemperatureConverter.class);

//Call the web service.
double fahr = port.celsiusToFahrenheit(100);
```



JAX-WS - start from WSDL

- JAX-WS supports start from WSDL approach

```
@WebService(name = "TemperatureConvertor",  
  endpointInterface="com.crayom.ws.TemperatureConvertor",  
  targetNamespace = "http://ws.crayom.com/",  
  wsdlLocation = "WEB-INF/TemperatureConvertorDocStyle.wsdl")  
public class TemperatureConvertorService implements  
  TemperatureConvertor {
```



JAX-WS - Provider

- Web Service endpoints may choose to work at the XML message level by implementing the Provider interface.
- The endpoint accesses the message or message payload using this low-level, generic API
- Implement one of the following
 - `Provider<Source>`
 - `Provider<SOAPMessage>`
 - `Provider<DataSource>`.



JAX-WS - Provider

```
@WebServiceProvider(serviceName = "TempConvProvider",
    portName="TempConvPort",
    targetNamespace = "http://www.crayom.com/om",
    wsdlLocation="WEB-INF/wsdl/TempConvProvider.wsdl")
@ServiceMode(value=Service.Mode.PAYLOAD)
public class TempConvProvider implements Provider<Source>{

    public Source invoke(Source request) {
        ...
        return resp;
    }
}
```

PAYLOAD gives the body of message.
MESSAGE will give whole SOAP Message

Provide WSDL
yourself



JAX-WS - Dispatch

- Web service client applications may choose to work at the XML message level by using the Dispatch<T> APIs.
- The javax.xml.ws.Dispatch<T> interface provides support for the dynamic invocation of service endpoint operations.
- Similar to Provider on server side



JAX-WS - Dispatch

```
Service service = Service.create(url, serviceName);
```

```
Dispatch<Source> sourceDispatch =  
    service.createDispatch(portQName,  
                            Source.class,  
                            Service.Mode.PAYLOAD);
```

//request is a XML which is put into SOAP payload

```
StreamSource streamSource = new StreamSource  
    (new StringReader(request));
```

```
Source result = sourceDispatch.invoke(streamSource);
```



JAX-WS - Apart from this

- JAX-WS provides support for SOAP fault handling in terms of exceptions.
- JAX-WS supports handlers both on client and server side , which can process SOAP headers. SOAP headers are used to build Quality of services (QOS).
- JAX-WS supports asynchronous communication



Thank you

