WebServices

Prof. Dr. Alexandre L'Erario

Definição de WebService

- Ampara arquitetura orientada a serviços:
  - Distribuída (na web)
  - Integração de negócios
  - Governança distribuída
  - Serviços menos complexos
  - Microserviços

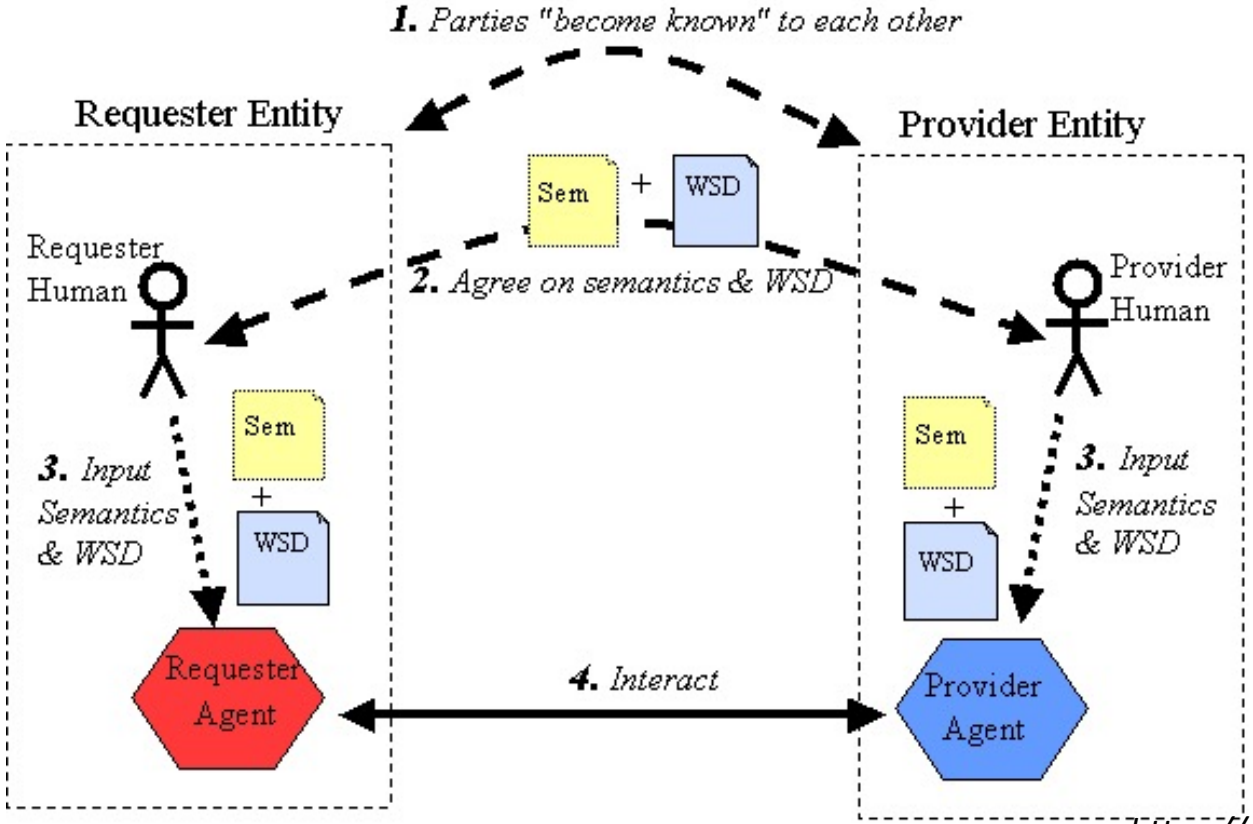
Referencia dos Próximos slides: https://www.w3.org/TR/ws-arch

Definição de WebService – W3C

“A Web service is a software system designed to support **interoperable machine-to-machine interaction over a network**. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”

https://www.w3.org/TR/ws-arch/#whatis

Visão geral do desenvolvimento

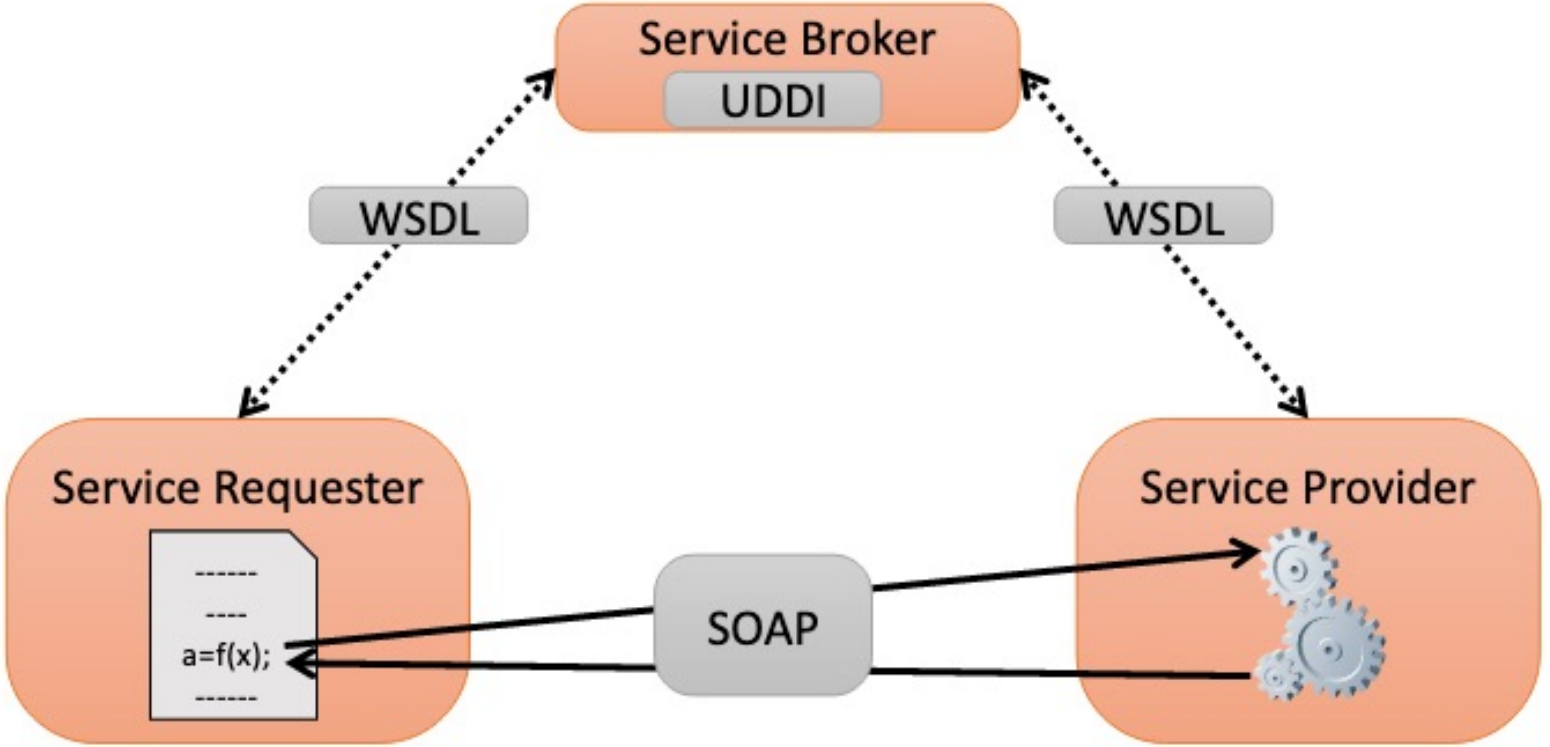


https://www.w3.org/TR/ws-arch/

Características

- Opera sobre o protocolo http/https
  - Demais protocolos também podem ser utilizados
  - Interoperabilidade com Firewalls
  - Diferente do RCP (C) ou RMI (Java)
  - Utiliza XML como conteúdo

Funcionamento



UDDI

- Universal Description, Discovery and Integration
  - Sistema de registro
  - Públicos e
  - Privativos
  - “Serviço de Diretórios” de Web Services SOAP

WSDL - Web Services Description Language

- Descreve o WebService como um contrato
  - Descrição abstrata dos serviços (XSD): envio e recebimento
  - Documentos WSDL

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

SOAP - Simple Object Access Protocol

- Construção da mensagem (envelope, header, body)
  - Message exchange patterns (MEP)
  - Define o modelo de processamento da mensagem: origem, destino e intermediários
  - Tratamento de falhas
  - Binding a protocolos WEB

Criação de Webservice SOAP em Java

- Pode ser implementado em um EJB Stateless
- Pacote javax foi substituído pelo jakarta
- Retomado e ativo na API EE 9.0
  - API 8.0 não contém algumas especificações para SOAP

Pom.xml

```
<dependency>
        <groupId>jakarta.platform</groupId>
        <artifactId>jakarta.jakartaee-api</artifactId>
        <version>9.0.0</version>
        <type>jar</type>
</dependency>
<dependency>
        <groupId>org.glassfish.metro</groupId>
        <artifactId>webservices-rt</artifactId>
        <version>2.3</version>
        <scope>provided</scope>
</dependency>
```

Servidor

Cliente

```
<dependency>
        <groupId>jakarta.platform</groupId>
        <artifactId>jakarta.jakartaee-api</artifactId>
        <version>9.0.0</version>
        <type>jar</type>
 </dependency>
 <dependency>
<groupId>com.sun.xml.messaging.saaj</groupId>
        <artifactId>saaj-impl</artifactId>
        <version>2.0.1</version>
 </dependency>
```

Implementação em Java - Servidor

```
import jakarta.jws.WebMethod;
import jakarta.jws.WebParam;
import jakarta.jws.WebService;
```

Atenção para os imports

```
@WebService(serviceName = "SoapService")
public class SoapService {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

Annotations

WSDL - Gerado automaticamente

```
<definitions targetNamespace="http://ws.br/" name="SoapService">
<types>
<xsd:schema>
<xsd:import namespace="http://ws.br/" schemaLocation="http://82868e56e240:8080/WsSoap/SoapService?xsd=1"/>
</xsd:schema>
</types>
<message name="hello">
<part name="parameters" element="tns:hello"/>
</message>
<message name="helloResponse">
<part name="parameters" element="tns:helloResponse"/>
</message>
<portType name="SoapService">
<operation name="hello">
<input wsam:Action="http://ws.br/SoapService/helloRequest" message="tns:hello"/>
<output wsam:Action="http://ws.br/SoapService/helloResponse" message="tns:helloResponse"/>
</operation>
</portType>
<binding name="SoapServicePortBinding" type="tns:SoapService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="hello">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding>
<service name="SoapService">
<port name="SoapServicePort" binding="tns:SoapServicePortBinding">
<soap:address location="http://82868e56e240:8080/WsSoap/SoapService"/>
</port>
</service>
</definitions>
```

Aplicação

http://<endereço>/WsSoap/SoapService?wsdl

@WebService(serviceName = "SoapService")

Implementação do cliente

```
String soapEndpointUrl = "http://localhost:8080/WsSoapServer/ServiceSoap?wsdl";
String soapAction = "http://localhost:8080/WsSoapServer/ServiceSoap";
SoapClient sc = new SoapClient();
sc.callSoapWebService(soapEndpointUrl, soapAction);
```

```
public class SoapClient {
   public void callSoapWebService(String soapEndpointUrl, String soapAction) {
      try {
         // Criar conexao SOAP
         SOAPConnectionFactory soapConnectionFactory = SOAPConnectionFactory.newInstance();
         SOAPConnection soapConnection = soapConnectionFactory.createConnection();

         // Enviar SOAP Message para o server
         SOAPMessage soapResponse = soapConnection.call(createSOAPRequest(soapAction), soapEndpointUrl);

         // Imprimir resposta
         System.out.println("Response SOAP Message:");
         soapResponse.writeTo(System.out);

         soapConnection.close();
      } catch (Exception e) { ...
```

import jakarta...

```
private static SOAPMessage createSOAPRequest(String soapAction) throws Exception {
      //criar mensagem SOAP
      MessageFactory messageFactory = MessageFactory.newInstance();
      SOAPMessage soapMessage = messageFactory.createMessage();
      //criar envelope SOAP
      createSoapEnvelope(soapMessage);
      MimeHeaders headers = soapMessage.getMimeHeaders();
      headers.addHeader("SOAPAction", soapAction);
      soapMessage.saveChanges();
      //Exibir mensagem
      System.out.println("Request SOAP Message:");
      soapMessage.writeTo(System.out);
     return soapMessage;
   }
```

```
private static void createSoapEnvelope(SOAPMessage soapMessage) throws SOAPException {
      SOAPPart soapPart = soapMessage.getSOAPPart();
      //verificar no wsdl o namespace utilizado
      String myNamespace = "ns2";
      String myNamespaceURI = "http://ws.br/";
      // Preencher SOAP Envelope
      SOAPEnvelope envelope = soapPart.getEnvelope();
      envelope.addNamespaceDeclaration(myNamespace, myNamespaceURI);
      // Preencher SOAP Body
      SOAPBody soapBody = envelope.getBody();
      SOAPElement soapBodyElem = soapBody.addChildElement("hello", myNamespace);
     //o child name foi criado sem namespace
      SOAPElement soapBodyElem1 = soapBodyElem.addChildElement("name");
      soapBodyElem1.addTextNode("Alexandre");
   }
```

Considerações finais

- WebService SOAP implica em um protocolo XML dentro do HTTP
  - O overhead SOAP é implícito da tecnologia
  - O payload SOAP é pequeno
- Sucessor: RestFul