

Construindo Web Services com JAX-WS

Anne Caroline
David Jansen
Eike Stálei

Estrutura da apresentação

1. Introdução
2. JAX-WS: Visão Geral
3. Criação Web Service e clientes com JAX-WS
4. Tipos Suportados pelo JAX-WS
5. Referências

Web Services

“Os Web Services fornecem um meio padrão de interoperar entre diferentes aplicativos de software, rodando em uma variedade de plataformas e / ou estruturas” segundo a W3C;

“Os Web Services permitem que aplicações cliente e servidor se comuniquem por meio do protocolo HTTP da World Wide Web (WWW)”

Microserviços;

JAX-WS e JAX-RS;



Web Services

“Os Web Services fornecem um meio padrão de interoperar entre diferentes aplicativos de software, rodando em uma variedade de plataformas e / ou estruturas” segundo a W3C;

“Os Web Services permitem que aplicações cliente e servidor se comuniquem por meio do protocolo HTTP da World Wide Web (WWW)”

Microserviços;

JAX-WS e JAX-RS;



JAX-WS: Visão Geral

A chamada de operação do Web Service é representada por um protocolo baseado em XML, como **SOAP**;

SOAP

- especificação para troca de informações entre sistemas;
- utiliza de documentos XML para transferência de objetos;
- utiliza o WSDL;



Criando um Web Service com JAX-WS

Etapas básicas para a criação dos códigos Web Service e do Cliente

1. Codifique a classe de implementação.
2. Compile a classe de implementação.
3. Empacote os arquivos em um arquivo WAR (Web Application Archive).
4. Implemente o arquivo WAR. Os artefatos de serviço da web, que são usados para se comunicar com clientes, são gerados pelo GlassFish Server durante a implantação.
5. Codifique a classe do cliente.
6. Use o ***wsimport*** do Maven para gerar e compilar os artefatos do Web Service necessários para se conectar ao serviço.
7. Compile a classe do cliente.
8. Execute o cliente.

Requisitos de um Endpoint JAX-WS

```
package jakarta.tutorial.helloservice;

import javax.xml.ws.WebService;
import javax.xml.ws.WebMethod;

1 @WebService
public class Hello {
    private final String message = "Hello, ";

    3 public Hello() {
    }

    2 @WebMethod
    public String sayHello(String name) {
        return message + name + ".";
    }
}
```

Criando um Web Service com JAX-WS

- 1 - Essa anotação deve ser colocada na classe de implementação;
- 2 - Essa anotação define os métodos de negócio expostos aos clientes do Web Service;
- 3 - A classe de implementação deve ter um construtor público padrão;

Outros requisitos

- A classe não deve ser declarada como ***abstract*** ou ***final***;
- ela pode usar as anotações ***@PostConstruct*** ou ***@PreDestroy*** para retornar chamadas de evento de ciclo de vida

Codificando a classe de implementação de endpoint

```
package jakarta.tutorial.helloservice;

import javax.ws.WebService;
import javax.ws.WebMethod;

@WebService
public class Hello {
    private final String message = "Hello, ";

    public Hello() {
    }

    @WebMethod
    public String sayHello(String name) {
        return message + name + ".";
    }
}
```


Criação, empacotamento e implantação do serviço

NetBeans IDE:

1. No menu Arquivo, escolha Abrir projeto.
2. Na caixa de diálogo Abrir projeto, navegue até: `tut-install/examples/jaxws`
3. Selecione a pasta helloservice-war.
4. Clique em Abrir projeto.
5. Na guia Projetos, clique com o botão direito do mouse no projeto helloservice-war e selecione Executar.
6. Este comando constrói e empacota o aplicativo em um arquivo WAR, helloservice-war.war, localizado em `tut-install / examples / jaxws / helloservice-war / target /`, e implanta esse arquivo WAR na instância do GlassFish Server.

Maven:

1. Em uma janela de terminal, vá para: `tut-install/examples/jaxws/hello-service-war/`
2. Digite o seguinte comando: `mvn install`
3. Este comando constrói e empacota o aplicativo em um arquivo WAR, `hello-service-war.war`, localizado no diretório de destino e, em seguida, implanta o WAR no GlassFish Server.

5. Você pode visualizar o arquivo WSDL do serviço implementado solicitando a URL <http://localhost:8080/helloservice-war/HelloService?wsdl> em um navegador da web. Agora você está pronto para criar um cliente que acesse este serviço.

Testando os métodos de um endpoint no Web Service

Requisitos:

- NetBeans rodando
- JDK 8
- GlassFish 4.1

Executar projeto e verificar WSDL:

- <http://localhost:8080/helloservice-war/HelloService?wsdl>

Pós - Execução:

- Acessar Endpoint: <http://localhost:8080/helloservice-war/HelloService?Tester>

Criando Aplicativo Cliente

O HelloAppCliente acessa o método sayHello do nosso HelloService.

```
import jakarta.tutorial.helloservice.endpoint.HelloService;  
import javax.xml.ws.WebServiceRef;  
  
public class HelloAppClient {  
    @WebServiceRef(wsdlLocation =  
        "http://localhost:8080/helloservice-war/HelloService?WSDL")  
    private static HelloService service;
```

Recupera porta para o serviço, invocando getHelloPortno:

```
jakarta.tutorial.helloservice.endpoint.Hello port = service.getHelloPort();
```

Ele invoca o sayHello método da porta:

```
return port.sayHello(arg0);
```

Criando Web Cliente (Servlet)

- Semelhante ao HelloAppCliente.
- Funciona do lado do servidor.
- Muda a apresentação da resposta.

```
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        out.println("<html lang=\"en\">");
        out.println("<head>");
        out.println("<title>Servlet HelloServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet HelloServlet at " +
            request.getContextPath () + "</h1>");
        out.println("<p>" + sayHello("world") + "</p>");
        out.println("</body>");
        out.println("</html>");

    }
}
```

```
private String sayHello(java.lang.String arg0) {
    jakarta.tutorial.helloservice.endpoint.Hello port =
        service.getHelloPort();
    return port.sayHello(arg0);
}
```

Tipos suportados por JAX-WS

- Mapeamento Schema-to-Java

XML Schema Type

xsd:string

xsd:integer

xsd:int

xsd:long

xsd:short

xsd:decimal

xsd:float

xsd:double

xsd:boolean

xsd:byte

Java Data Type

java.lang.String

java.math.BigInteger

int

long

short

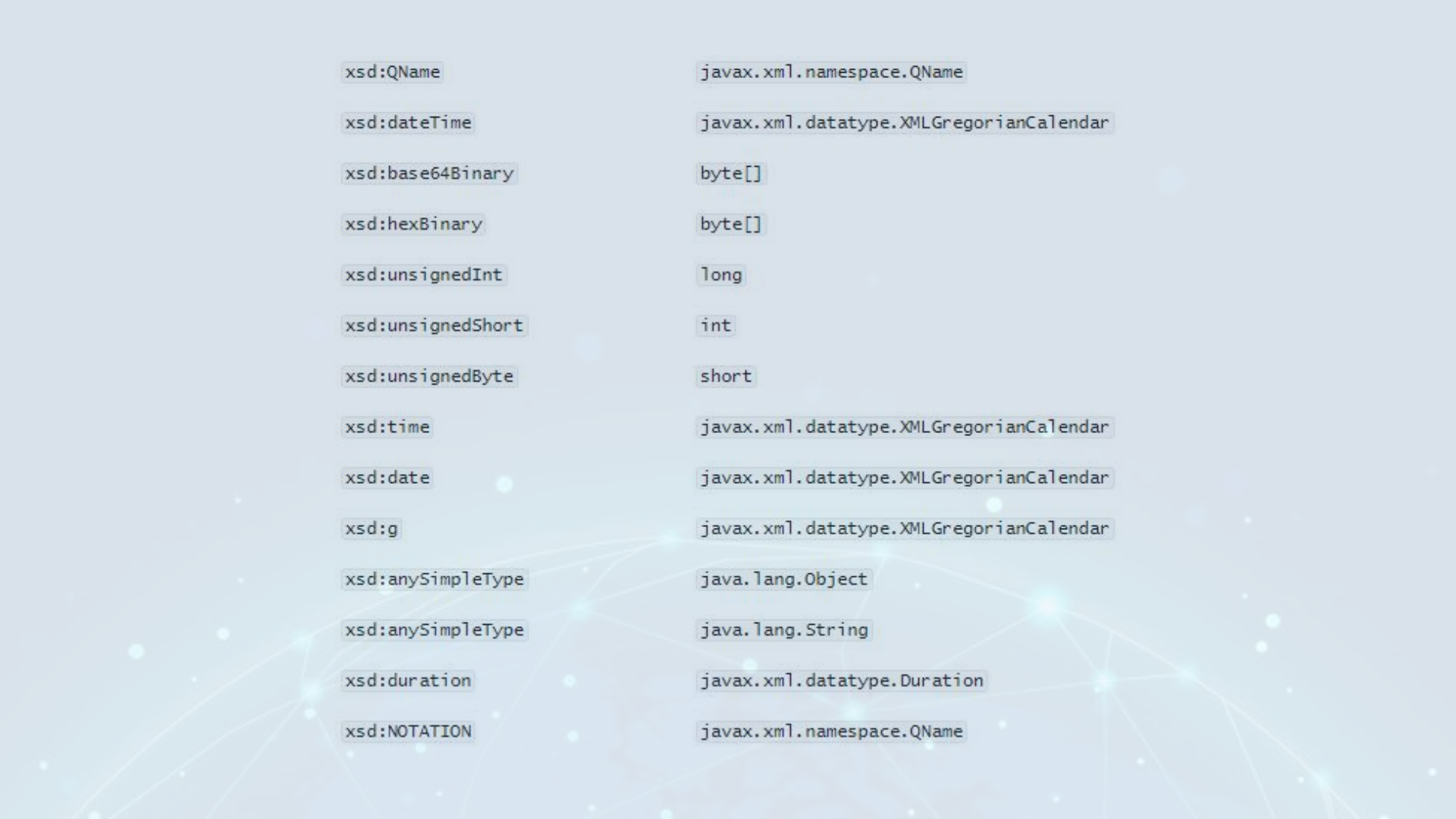
java.math.BigDecimal

float

double

boolean

byte



xsd:QName	javax.xml.namespace.QName
xsd:dateTime	javax.xml.datatype.XMLGregorianCalendar
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]
xsd:unsignedInt	long
xsd:unsignedShort	int
xsd:unsignedByte	short
xsd:time	javax.xml.datatype.XMLGregorianCalendar
xsd:date	javax.xml.datatype.XMLGregorianCalendar
xsd:g	javax.xml.datatype.XMLGregorianCalendar
xsd:anySimpleType	java.lang.Object
xsd:anySimpleType	java.lang.String
xsd:duration	javax.xml.datatype.Duration
xsd:NOTATION	javax.xml.namespace.QName

- Java-to-Schema Mapping

Java Class	XML Data Type
<code>java.lang.String</code>	<code>xs:string</code>
<code>java.math.BigInteger</code>	<code>xs:integer</code>
<code>java.math.BigDecimal</code>	<code>xs:decimal</code>
<code>java.util.Calendar</code>	<code>xs:dateTime</code>
<code>java.util.Date</code>	<code>xs:dateTime</code>
<code>javax.xml.namespace.QName</code>	<code>xs:QName</code>
<code>java.net.URI</code>	<code>xs:string</code>
<code>javax.xml.datatype.XMLGregorianCalendar</code>	<code>xs:anySimpleType</code>
<code>javax.xml.datatype.Duration</code>	<code>xs:duration</code>
<code>java.lang.Object</code>	<code>xs:anyType</code>
<code>java.awt.Image</code>	<code>xs:base64Binary</code>
<code>javax.activation.DataHandler</code>	<code>xs:base64Binary</code>
<code>javax.xml.transform.Source</code>	<code>xs:base64Binary</code>
<code>java.util.UUID</code>	<code>xs:string</code>

Web Services Interoperability e JAX-WS

JAX-WS suporta o Web Services Interoperability (WS-I) Basic Profile versão 1.1.

Para suportar WS-I Basic Profile Versão 1.1, o tempo de execução JAX-WS suporta codificações doc/literal e rpc/literal para serviços, portas estáticas, proxies dinâmicos e Dynamic Invocation Interface (DII).



Referências

[W3C - Definição Web Services \(slide 3\);](#)

[\[Semana X\] Aula 38 - Jakarta Webservices \(Parte 1\);](#)

[\[Semana X\] Aula 39 - Jakarta Webservices \(Parte 2\);](#)

JakartaEE Tutorial (Disponibilizado pelo professor Luis Guisso);

<https://github.com/DWalterJansen/seminario-webdev>