✅ Conceitos Importantes

🔗 Bindings SOAP

- O elemento `<binding>` possui:
    - **name:** nome do binding.
    - **type:** referencia o `portType` (ex.: `OperacoesWS`).
- O elemento `<soap:binding>` define:
    - **style:** pode ser `document` ou `rpc`.
        - `document`: mais usado atualmente, mensagens baseadas em documentos XML.
        - `rpc`: estilo remoto, mais antigo.
    - **transport:** geralmente é HTTP (`http://schemas.xmlsoap.org/soap/http`).

🔧 JAX-WS (Java API for XML Web Services)

- API Java padrão para criar Web Services SOAP.
- Substitui JAX-RPC.
- Suporte a SOAP 1.1, 1.2 e outros protocolos XML.
- Utiliza **anotações** para simplificar o desenvolvimento:
    - `@WebService`: define a classe como Web Service.
    - `@WebMethod`: expõe o método no serviço.
    - `@WebParam`: nomeia os parâmetros na interface WSDL.

🏗️ JAXB (Java Architecture for XML Binding)

- Permite transformar objetos Java em XML e vice-versa (Marshalling e Unmarshalling).
- Alternativa mais simples que usar SAX ou DOM diretamente.
- Processo básico:
    1. **Vincular** o esquema XML (geração de classes).
    2. **Construir** uma árvore de objetos Java.
    3. **Gerar** o XML a partir dos objetos (ou ler um XML para objetos).

💻 Exemplo Completo de Web Service

✅ Servidor (OperacoesWS)

```
package com.webservice;

import java.util.ArrayList;
import java.util.List;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(serviceName = "OperacoesWS")
public class OperacoesWS {

    @WebMethod(operationName = "reverterTexto")
    public String reverterTexto(@WebParam(name = "texto") String texto) {
        return new StringBuilder(texto).reverse().toString();
    }

    @WebMethod(operationName = "ListarCidades")
    public List<String> ListarCidades(@WebParam(name = "estado") String estado) {
        List<String> lista = new ArrayList<>();
        if(estado.equalsIgnoreCase("SP")) {
            lista.add("São Paulo");
            lista.add("Campinas");
            lista.add("Ribeirão Preto");
        } else if(estado.equalsIgnoreCase("AC")) {
            lista.add("Rio Branco");
```

```
            lista.add("Sena Madureira");
            lista.add("Cruzeiro do Sul");
        }
        return lista;
    }
}
```

🌐 WSDL Gerado

- Disponível em:

http://localhost:8084/Operacoes/OperacoesWS?wsdl

- Contém as definições de:
    - **types:** tipos XML.
    - **message:** mensagens (entrada/saída).
    - **portType:** operações.
    - **binding:** protocolos e formatos.
    - **service:** endereço do serviço.

✅ Cliente Java (Servlet)

```
package com.webservice.client;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "ClienteWSServlet", urlPatterns = {"/clientews"})
public class ClienteWSServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String texto = request.getParameter("texto");
        String estado = request.getParameter("estado");

        out.println("<h2>Texto fornecido: " + texto + "</h2>");
        out.println("<h3>Texto invertido: " + reverterTexto(texto) + "</h3>");

        out.println("<h2>Estado fornecido: " + estado + "</h2>");
        out.println("<h3>Cidades:</h3><ul>");
        for(String cidade : listarCidades(estado)) {
            out.println("<li>" + cidade + "</li>");
        }
        out.println("</ul>");
    }

    private static List<String> listarCidades(String estado) {
        com.webservice.OperacoesWS_Service service = new
com.webservice.OperacoesWS_Service();
        com.webservice.OperacoesWS port = service.getOperacoesWSPort();
        return port.listarCidades(estado);
    }

    private static String reverterTexto(String texto) {
        com.webservice.OperacoesWS_Service service = new
com.webservice.OperacoesWS_Service();
        com.webservice.OperacoesWS port = service.getOperacoesWSPort();
        return port.reverterTexto(texto);
    }
}
```

📑 Funcionamento

- O Web Service é exposto via SOAP.
- O WSDL descreve como os clientes podem consumir o serviço.
- O cliente (um Servlet) acessa os métodos do Web Service da mesma forma que chamaria métodos locais, graças às classes geradas automaticamente a partir do WSDL.

🚩 Resumo das Anotações Importantes

Anotação	Função
`@WebService`	Define a classe como Web Service SOAP.
`@WebMethod`	Expõe o método na interface do serviço.
`@WebParam`	Nomeia o parâmetro que aparece no WSDL.

🚀 Vantagens do JAX-WS

- Simplicidade com uso de anotações.
- Integração fácil com servidores como Tomcat, GlassFish, JBoss.
- Compatível com ferramentas para geração automática de clientes (`wsimport`) e WSDL (`wsgen`).