Unidade 14: Web Services

Prof. Daniel Caetano

INTRODUÇÃO

Na maior parte deste curso foram estudados serviços desenvolvidos com o uso de tecnologias Servlet. Entretato, os Servlets nem sempre serão suficientes, isto é, nem sempre serão capazes de atender às nossas necessidades. Sendo assim, uma nova tecnologia torna-se necessária.

Nesta aula veremos uma tecnologia que supera algumas das limitações dos Servlets.

1. O QUE É UM WEB SERVICE?

Antes de entendermos o que é um WebService, é preciso entender... por que um WebService? Já não aprendemos a fazer serviços Java com Servlets?

Sim, já aprendemos a fazer serviços com Servlets. E os servlets possuem uma infinidade de vantagens sobre os WebServices, em especial quando executados em um container como o GlassFish, que permite acesso a todos os recursos do Java. Qual é o problema, então?

O primeiro problema com os Servlets é que, de forma “direta” eles só conseguem transferir requisições entre servlets que estejam em um mesmo equipamento. É possível contornar esse problema, mas não é tão simples.

Um outro problema é que a tecnologia Servlet é exclusiva da plataforma Java... e nem sempre os serviços de uma empresa serão todos em Java. Como assim? Alguém faz algo que não seja em Java? Sim, e muito! E a tendência não é que isso mude! Na verdade, a tendência é que daqui algum tempo (oxalá muito tempo) o Java é que deixe de ser usado, substituído por outras coisas. E aí? Como fica a interoperabilidade entre os Servlets e outros sistemas?

É possível garantir algum tipo de interoperabilidade, mas aquelas que envolvem a transferência de objetos dentro das requisições, por exemplo, certamente trarão enormes problemas. Em geral, toda a troca de dados binários pode levar a problemas potenciais por diferentes razões:

a) Computadores diferentes podem representar dados diferentemente na memória.
b) Linguagens diferentes vão certamente representar os dados de maneira diferente na memória.
c) Os tipos de dados de uma linguagem podem não estar disponíveis em outra linguagem.

Alguns podem se perguntar: "Mas os servlets não trocam requisições e respostas HTTP padrão? Como essa babel pode ocorrer?"

É aí que mora o desastre: o HTTP é um protocolo para **texto**. Tudo que for transmitido como texto tem boa chance de se manter legível entre diferentes plataformas, hoje ou no futuro. É por isso que foi feita uma referência explícita a **troca de dados binários**, como imagens e objetos de programação. O protocolo HTTP não define uma forma clara e detalhada para especificar esse tipo e informação.

A solução foi criar uma padronização para especificar dados binários e estruturas de dados mais complexas, como um objeto. Essa padronização, criada pela Microsoft e IBM, por ser usada justamente para o desenvolvimento de **serviços** em arquiteturas de software orientadas a serviços, recebeu o nome de **SOAP: Service Oriented Architeture Protocol**.

Como a IBM e a Microsoft já se cansaram de reinventar a roda, o protocolo SOAP foi definido com base na especificação XML, permitindo a definição de tipos complexos. O protocolo SOAP define o formato de mensagens de requisições e de mensagens de resposta, que tem aparência como as indicadas a seguir:

SOAP Request

```
----------------------------
    <?xml version="1.0" encoding="UTF-8"?>
  <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
      <ns2:print xmlns:ns2="http://hello.me.org/">
        <nome>Fulano</nome>
      </ns2:print>
    </S:Body>
  </S:Envelope>
----------------------------
```

SOAP Response

```
----------------------------
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:printResponse xmlns:ns2="http://hello.me.org/">
      <return>OlÃ¡, Fulano</return>
    </ns2:printResponse>
  </S:Body>
</S:Envelope>
----------------------------
```

Foge ao escopo deste curso detalhar o protocolo SOAP, mas é importante saber que a sintaxe dele obedece ao padrão XML e que ele é o caminho encontrado para **padronizar a comunicação** entre os Web Services (Serviços Web).

Ok, então um Web Service é como se fosse um servlet, só que a comunicação entre eles não ocorre por texto/binário puros pelo "HTTP", mas sim pelo envio de arquivos XML formatado segundo as regras do protocolo SOAP, através do HTTP?

ISSO...! Ou quase!

2. WEB SERVICES SÃO DINÂMICOS

O que significa que os Web Services são dinâmicos? E... que os Servlets são estáticos?

Vamos imaginar um exemplo. Imaginemos que estamos construindo um aplicativo que faz reserva em hoteis. Você especifica as características, ele procura o hotel mais barato que as atenda e faz a reserva. Ora, esse programa precisa se conectar com os sistemas dos diversos hoteis para poder fazer as verificações e reservas, correto?

É possível fazer isso com servlets até um determinado ponto: a lista de servidores de hoteis precisa ser definida manualmente, e você precisará entrar em contato com todos os desenvolvedores dos hoteis para obter o protocolo de cada sistema, isto é, como é que você deve solicitar as informações para cada sistema.

Isso muda nos WebServices? **SIM!**

Muda porque, primeiramente, existe um protocolo chamado **UDDI - Universal, Description, Discovery and Integration**. Esse protocolo é usado por um sistema que é uma espécie de "google" dos aplicativos Web, e ele existe para que um programa possa buscar por serviços, de diferentes origens, que estejam online naquele instante e possam responder a requisições. Se um hotel fecha, por exemplo, o UDDI deixará de informar aquele serviço como disponível. Se um novo hotel surge, suas informações são publicadas no UDDI.

Ok, já achei um novo serviço, mas como usá-lo? Também para isso existe um padrão de divulgação, chamado **WSDL: Web Services Description Language**. O WSDL descreve quais operações estão disponíveis, quais parâmetros enviar e o quê será recebido de volta.

Tanto o UDDI quanto o WSDL foram definidos também com base na sintaxe do XML. Um WSDL tem uma cara mais ou menos assim:

```
----------------------------------
<!--
 Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.1-hudson-28-.
-->
—
<!--
 Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.1-hudson-28-.
-->
—
<definitions targetNamespace="http://imcws/" name="IMCWSService">
—
<types>
—
<xsd:schema>
<xsd:import namespace="http://imcws/" schemaLocation="http://localhost:8080/IMCWSApp/IMCWSService?xsd=1"/>
</xsd:schema>
</types>
—
<message name="calcula">
<part name="parameters" element="tns:calcula"/>
</message>
—
```

```
<message name="calculaResponse">
<part name="parameters" element="tns:calculaResponse"/>
</message>
—
<portType name="IMCWS">
—
<operation name="calcula">
<input wsam:Action="http://imcws/IMCWS/calculaRequest" message="tns:calcula"/>
<output wsam:Action="http://imcws/IMCWS/calculaResponse" message="tns:calculaResponse"/>
</operation>
</portType>
—
<binding name="IMCWSPortBinding" type="tns:IMCWS">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
—
<operation name="calcula">
<soap:operation soapAction=""/>
—
<input>
<soap:body use="literal"/>
</input>
—
<output>
<soap:body use="literal"/>
</output>
</operation>
</binding>
—
<service name="IMCWSService">
—
<port name="IMCWSPort" binding="tns:IMCWSPortBinding">
<soap:address location="http://localhost:8080/IMCWSApp/IMCWSService"/>
</port>
</service>
</definitions>
-----------------------------------
```

Feio, não? Muito feio mas... funciona!

3. QUAL O PROCEDIMENTO PARA USAR ISSO?

Tudo ocorre entre 3 elementos: o "**Solicitante do Serviço**", o "**Distribuidor de Serviços**" e o "**Provedor de Serviço**".

A. Primeiramente, uma aplicação que usará os serviços (o Solicitante do Serviço) consulta o chamado "Distribuidor de Serviços", que usa o protocolo UDDI para receber informações sobre o serviço desejado e devolver uma lista de serviços.

B. Com a lista de serviços, o "Solicitante de Serviço" solicita informações sobre como usar um serviço específico, o que ele recebe do "Distribuidor de Serviços" no formado "WSDL".

C. De posse do WSDL, o "Solicitante do Serviço" consegue contatar o "Provedor de Serviço" e, através do protocolo SOAP, envia uma requisição e, posteriormente, recebe uma resposta.

Tudo isso parece muito complicado!

E É complicado. Por sorte, o pessoal da Oracle/Sun e do NetBeans já pensou em tudo para nós, tornando a tarefa de criar um WebService bastante simples...

4. GERANDO UM WEBSERVICE NO NETBEANS

PASSO 1. Crie um novo projeto para uma Aplicação Web: "**Arquivo > Novo Projeto...**" e selecione "**Java Web > Aplicação Web**". Dê o nome de "**IMCWSApp**" a esta aplicação e finalize.

PASSO 2. Nos pacotes de código fonte, crie um pacote chamado "**imcws**".

PASSO 3. No pacote "**imcws**", clique com o botão direito e selecione "**Novo > Serviço Web**". Dê o nome de "**IMCWS**" para o serviço.

PASSO 4. O NetBeans irá criar uma nova pasta chamada "**Serviços Web**". Abra-a.

PASSO 5. Na pasta "**Serviços Web**", clique com o botão direito no **IMCWS** e selecione "**Adicionar Operação...**".

PASSO 6. No nome da operação indique "**calcula**". Como tipo de retorno, selecione "**double**".

PASSO 7. Acrescente dois parâmetros: "**peso**" e "**altura**", ambos do tipo "**double**". Finalmente, clique em "Ok".

PASSO 8. Edite o código do **IMCWS.java** para que o código do método "calcula" fique com o seguinte conteúdo:

```
double imc = peso / (altura * altura);
return imc;
```

PASSO 9. Grave e, depois, clique com o botão direito no ícone do projeto, de nome **IMCWSApp** e selecione a opção "**Implantar**".

PASSO 10. Agora, vamos testar se está tudo ok com nosso serviço Web: na pasta "**Serviços Web**", clique com o botão direito em **IMCWS** e selecione a opção "**Testar Serviço Web**".

PASSO 11. O link "**WSDL File**" mostra o arquivo WSDL gerado pelo NetBeans. Se você indicar os valores para peso e altura nos campos e clicar no botão calcula, verá como resultado o valor calculado e também os arquivos SOAP enviados do cliente para o servidor (SOAP Request) e do servidor para o cliente (SOAP Response).

5. CONSUMINDO UM WEBSERVICE NO NETBEANS

Um WebService pode ser consumido a partir de qualquer aplicação Java. Neste tutorial, vamos construir um servlet que usa o WebService criado anteriormente para calcular o IMC. **NÃO** feche o projeto IMCWSApp!

PASSO 12. Crie um novo projeto para uma Aplicação Web: "**Arquivo > Novo Projeto...**" e selecione "**Java Web > Aplicação Web**". Dê o nome de "**IMCWSClienteApp**" a esta aplicação e finalize.

PASSO 13. Clique com o botão direito no ícone do novo projeto, de nome "**IMCWSClienteApp**" e selecione: "**Novo > Cliente para serviço web**".

NOTA: Se a opção não estiver disponível, vá em "**Novo > Outro...**" e depois selecione "**Serviços Web > Cliente para serviço web**".

PASSO 14. Na janela que vai aparecer, quando for solicitado que seja especificado o "WSDL" do serviço web, você pode fazer duas coisas: ou indicar como "**Projeto**" e apontar o nome de nosso outro projeto... ou, caso você não tenha acesso ao código do projeto, você pode indicar na terceira linha direto do endereço do WSDL do serviço, como, neste exemplo http://localhost:8080/IMCWSApp/IMCWSService?WSDL

PASSO 15. Agora, nosso sistema está pronto para usar nosso WebService, mas precisamos criar o sisteminha, que será composto de um formulário "index.jsp", que apresentará o formulário perguntando peso e altura, e um servlet chamado "**imc.java**". Vamos começar pelo **index.jsp**. Modifique o "index.jsp" para que tenha essa aparência:

index.jsp

```
<%--
  Document   : index
  Created on : 28/05/2011, 16:01:16
  Author     : djcaetano
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
  <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cálculo de Índice de Massa Corporal com Web Services</title>
  </head>
  <body>
    <h1>Digite suas informações!</h1>
    <form action ="imc">
      <p>Peso: <input type="text" name="peso"></p>
      <p>Altura: <input type="text" name="altura"></p>
      <input type="submit" value="Calcular!">
    </form>
  </body>
</html>
```

PASSO 16. Agora é hora de criar o servlet. Criemos o pacote de nossa aplicação: clique com o botão direito em "**Pacotes de Código Fonte**" e selecione "**Novo > Pacote Java**". Dê o nome de "**imcapp**" para o pacote.

PASSO 17. Neste pacote, clique com o botão direito e crie o servlet "**imc**": "**Novo > Servlet**" e dê o nome de "**imc**", não se esquecendo de **adicionar a informação ao descritor**.

PASSO 18. Agora, **atenção**! Na área de projeto do NetBeans, abra a pasta "**Referências de Serviços Web**". Dentro dela haverá várias subpastas. Abra o caminho todo: "**Referências de Serviços Web > IMCWSService > IMCWSService > IMCWSPort > calcula**". Este último terá uma bolinha vermelha, indicando tratar-se de uma funcionalidade.

PASSO 19. **Mais atenção** ainda! Clique e segure o botão esquerdo sobre esta bolinha e **arraste** o mouse até a área de código do servlet "imc.java", na região **após** o "processRequest". Isso fará aparecer o seguinte "método", que é um acesso ao WebService.

```
-------------------------------------------------
    private float calcula(float peso, float altura) {
    imcws.IMCWS port = service.getIMCWSPort();
    return port.calcula(peso, altura);
  }
-------------------------------------------------
```

PASSO 20. Vamos usar o método de acesso criado. Altere o método processRequest para receber os dados do formulário e chamar o WebService, da seguinte forma:

imc.java

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

        try {
            // Recebe valores do formulário
            double peso = Double.valueOf(request.getParameter("peso"));
            double altura = Double.valueOf(request.getParameter("altura"));
            if (altura == 0.0) altura = 1.0;

            // Calcula IMC pelo Web Service
            double imc = calcula(peso, altura);

            // Imprime resultado
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Resultado do WebService IMCWS</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>IMC: " + imc + "</h1>");
            out.println("</body>");
            out.println("</html>");
          } finally {
            out.close();
          }
    }
```

Pronto! Você criou seu primeiro Web Service e um programa que o usa!