

Desenvolvimento de Aplicações Java
Plataforma Corporativa

Tutorial
Web Services
Clientes

Outubro 2016

Sumário

1.Introdução.....	3
2. Consumo de webservices SOAP	4
2.1. Cliente Java para um webservice SOAP	4
2.2.1. Preparação – publicação do webservice SOAP	4
2.2.2. Geração de artefatos necessários para consumo do webservice SOAP	4
2.2.3. Criar a aplicação Java que irá consumir o WS SOAP	5
3. Consumo de webservices REST	7
3.1. Suporte ao JSON.....	9
3.1.1. Parse de uma mensagem JSON.....	11

1.Introdução

Implementação de clientes Java que consomem web services.

2. Consumo de webservices SOAP

Como a publicação de webservices não exige o uso de uma plataforma específica, a criação de clientes que consomem um webservice não devem depender de nada além de uma descrição de seus métodos (end-points) e argumentos (elementos).

Toda a descrição necessária para que aplicações possam consumir um webservice SOAP deve constar no arquivo **wsdl**.

2.1. Cliente Java para um webservice SOAP

Para consumir um webservice SOAP precisamos criar classes Java para acessar estes webservices. Estas classes auxiliares são geradas através da utilização de um utilitário chamado **wsimport**, que gerará todos os artefatos necessários para implementarmos uma aplicação cliente Java para um webservice SOAP.

2.2.1. Preparação – publicação do webservice SOAP

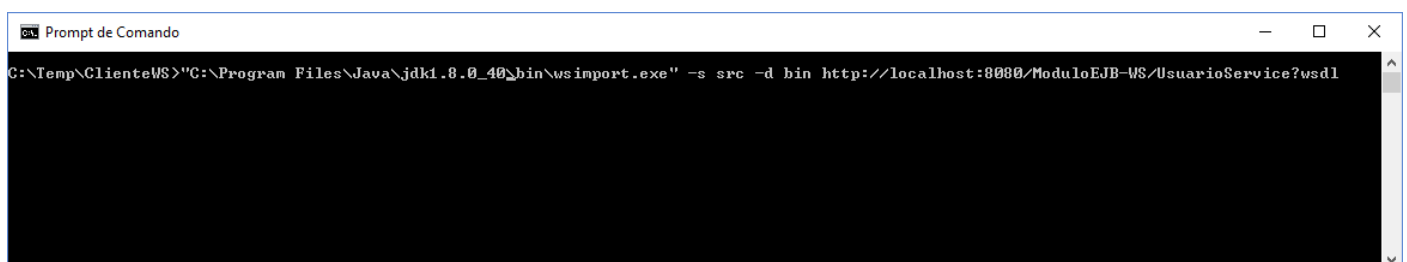
Rode o Wildfly e implante a aplicação corporativa desenvolvida na atividade de laboratório com o webservice SOAP **UsuarioService**.

Verifique se o webservice foi publicado corretamente, consultando o respectivo **wsdl** utilizando o seu navegador de preferência.

2.2.2. Geração de artefatos necessários para consumo do webservice SOAP

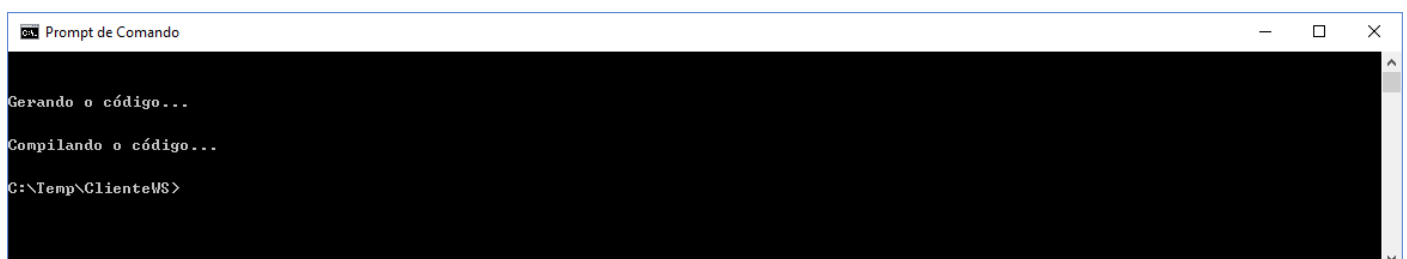
Crie uma pasta **ClienteWS** na pasta **C:\Temp**. Dentro desta nova pasta, crie uma pasta **src**, onde iremos gerar o código fonte dos artefatos gerados para acessar o ws SOAP e uma pasta **bin**, onde iremos gerar os artefatos compilados para acesso ao ws SOAP. Abra uma janela de prompt de comando e navegue até este novo diretório e execute o comando:

```
"C:\Program Files\Java\jdk1.8.0_40\bin\wsimport.exe" -s src -d bin http://localhost:8080/ModuloEJB/UsuarioService?wsdl
```

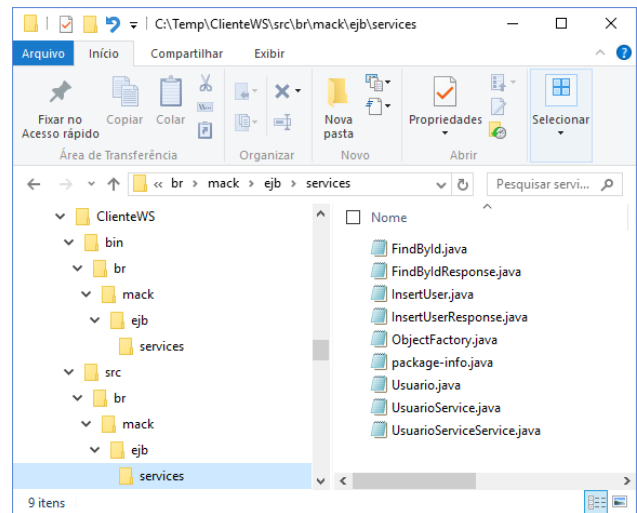
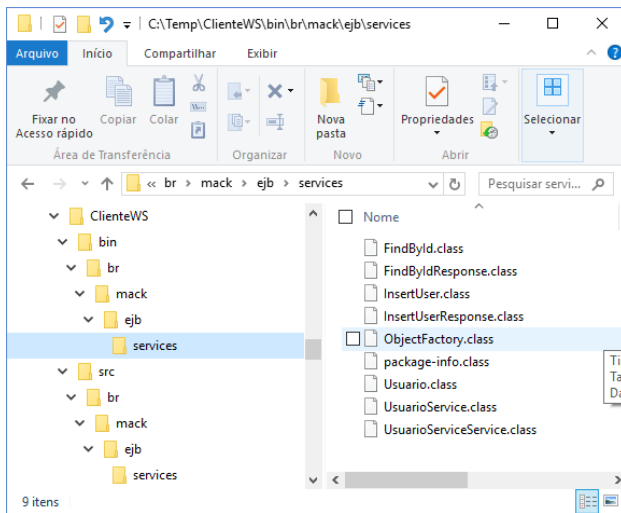


```
Prompt de Comando
C:\Temp\ClienteWS>"C:\Program Files\Java\jdk1.8.0_40\bin\wsimport.exe" -s src -d bin http://localhost:8080/ModuloEJB-WS/UsuarioService?wsdl
```

Como resultado da execução, devemos observar no prompt de comando:

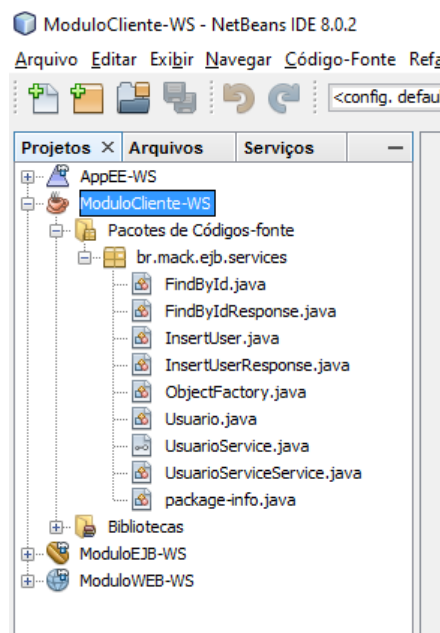


```
Prompt de Comando
Gerando o código...
Compilando o código...
C:\Temp\ClienteWS>
```



2.2.3. Criar a aplicação Java que irá consumir o WS SOAP

Criar uma aplicação Java (simples) chamada: **ModuloCliente-WS**. Copiar para esta aplicação as classes geradas no passo anterior (conteúdo da pasta **src** para **Pacotes de Códigos-fonte**).



Crie a classe `mack.app.MinhaAppSOAP`.

```
package mack.app;
```

```
import.ejb.services.Usuario;
```

```
import mack.ejb.services.UsuarioService;
```

```
import mack.ejb.services.UsuarioServiceService;
```

```
public class MinhaAppSOAP {
```

```
public static void main(String[] args){  
    UsuarioServiceService uss = new UsuarioServiceService();  
    UsuarioService us = uss.getUsuarioServicePort();  
    Usuario u = new Usuario();  
    u.setId(2);  
    u.setName("Nome");  
    u.setFamilyName("Sobrenome");  
    us.insertUser(u);  
    Usuario uRet = us.findById(1);  
    System.out.println(uRet.getId() + " " + uRet.getName() + " " + uRet.getFamilyName());  
}  
}
```

Execute a aplicação.

3. Consumo de webservices REST

O consumo de webservices REST é um pouco mais simples, mas devemos conhecer a URL que iremos consultar, além dos elementos que iremos enviar/receber.

No mesmo projeto de aplicação Java anterior, crie a classe `mack.app.MinhaAppRest`.

Nesta classe, implemente o método estático `consultaPorId` que recebe como parâmetro um `id` do tipo inteiro.

```
public static void consultaPorId(int id) {
    try {

        String baseURL =
"http://localhost:8080/AppFrontController/GerenciamentoUsuarios/usuarios/";

        URL url = new URL(baseURL + "usuario/" + id);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/xml");

        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Falha : Erro HTTP: " + conn.getResponseCode());
        }

        BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

        String output;
        System.out.println("Output from Server .... \n");
        while ((output = br.readLine()) != null) {
            System.out.println(output);
        }

        conn.disconnect();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Implemente o método estático **main**, coloque uma chamada para o método implementado e execute sua aplicação.

```
public static void main(String[] args) {  
    consultaPorId(1);  
}
```

Por fim, implemente o método de classe **insereUsuario** que recebe como parâmetro duas Strings, um nome e um sobrenome.

```
public static void insereUsuario(String nome, String sobrenome) {  
    try {  
  
        String baseUrl = "http://localhost:8080/ModuloWEB-  
WS/GerenciamentoUsuarios/usuarios/";  
  
        URL url = new URL(baseUrl + "novousuario");  
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
        conn.setRequestMethod("PUT");  
        conn.setDoOutput(true);  
        conn.setRequestProperty("Content-Type", "application/xml");  
  
        String input =  
"<usuario><id>2</id><name>"+nome+"</name><familyName>"+sobrenome+"</familyName></usu-  
ario>";  
  
        OutputStream os = conn.getOutputStream();  
        os.write(input.getBytes());  
        os.flush();  
  
        if (conn.getResponseCode() != HttpURLConnection.HTTP_CREATED) {  
            throw new RuntimeException("Falha : Erro HTTP: "+ conn.getResponseCode());  
        }  
  
        BufferedReader br = new BufferedReader(new  
InputStreamReader((conn.getInputStream())));  
  
        String output;  
  
        System.out.println("Resposta do servidor .... \n");
```



```

        while ((output = br.readLine()) != null) {
            System.out.println(output);
        }

        conn.disconnect();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Modifique o método estático **main**, coloque uma chamada para o novo método implementado e execute sua aplicação.

```

public static void main(String[] args) {
    consultaPorId(1);
    insereUsuario("NovoNome", "NovoSobrenome");
    consultaPorId(2);
}

```

Teste sua implementação.

3.1. Suporte ao JSON

Quando utilizamos webservices REST, é bastante comum termos clientes que não gostariam de precisar manipular XML e prefeririam utilizar o JSON. Para termos suporte ao JSON, devemos alterar a configuração (anotação) do webservice REST.

Modifique a anotação do método **buscaUsuarioPorId** da classe **UsuarioRestService** do projeto **ModuloEJB-WS** para:

```

@GET
@Path("/usuario/{id}")
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
public Usuario buscaUsuarioPorId(@PathParam("id") final int id) {
    System.out.println("Buscando usuario: " + id);
    Usuario u = ub.buscaUsuarioPorId(id);
    if (u==null){
        throw new WebApplicationException (Response.Status.NOT_FOUND);
    }
}

```

```
}  
  
return u;  
  
}
```

Note que o webservice pode agora produzir um XML ou um JSON.

Implemente as alterações, recompile o módulo EJB, reconstrua a aplicação corporativa e implante novamente a aplicação corporativa. Lembre-se que o hashmap de usuários será reinicializado sem nenhum usuário cadastrado.

A aplicação cliente deve então definir o tipo de dado que gostaria de receber. Para requisitar uma resposta JSON, devemos implementar um novo método estático na classe **mack.app.MinhaAppRest** chamado **consultaPorIdJson** que recebe como parâmetro um **id** do tipo inteiro.

```
public static void consultaPorIdJson(int id) {  
    try {  
        String baseUrl = "http://localhost:8080/ModuloWEB-  
WS/GerenciamentoUsuarios/usuarios/";  
  
        URL url = new URL(baseUrl + "usuario/" + id);  
  
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
  
        conn.setRequestMethod("GET");  
  
        conn.setRequestProperty("Accept", "application/json");  
  
        if (conn.getResponseCode() != 200) {  
            throw new RuntimeException("Falha : Erro HTTP: " + conn.getResponseCode());  
        }  
  
        BufferedReader br = new BufferedReader(new  
InputStreamReader((conn.getInputStream())));  
  
        String output;  
  
        System.out.println("Output from Server .... \n");  
  
        while ((output = br.readLine()) != null) {  
            System.out.println(output);  
        }  
  
        conn.disconnect();  
    } catch (MalformedURLException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

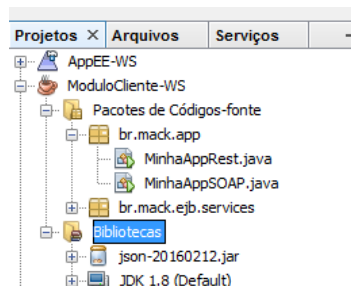
Modifique o método estático **main**, coloque uma chamada para o novo método implementado e execute sua aplicação.

```
public static void main(String[] args) {  
  
    insereUsuario("NovoNome", "NovoSobrenome");  
  
    consultaPorIdJson(1);  
  
}
```

Teste sua implementação.

3.1.1. Parse de uma mensagem JSON na aplicação Java

Para tratar uma mensagem JSON, vamos utilizar uma biblioteca, a **org.json**. Baixe a biblioteca do Moodle e inclua nas bibliotecas de sua aplicação Java.



Crie um método de classe chamado **parseJson** que recebe como parâmetro uma String com o json para processamento.

```
public static void parseJson(String jsonString){  
  
    JSONObject jsonObject = new JSONObject(jsonString);  
  
    System.out.println(jsonObject.getInt("id"));  
  
    System.out.println(jsonObject.getString("name"));  
  
    System.out.println(jsonObject.getString("familyName"));  
  
}
```

Modifique seu método **consultaPorIdJson** e chame o método de processamento de Json.

```
public static void consultaPorIdJson(int id) {  
  
    try {  
  
        String baseURL =  
"http://localhost:8080/AppFrontController/GerenciamentoUsuarios/usuarios/";  
  
        URL url = new URL(baseURL + "usuario/" + id);  
  
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
  
        conn.setRequestMethod("GET");  
  
        conn.setRequestProperty("Accept", "application/json");  
  

```

```

        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Falha : Erro HTTP: " + conn.getResponseCode());
        }

        BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

        String output;

        System.out.println("Output from Server .... \n");

        while ((output = br.readLine()) != null) {
            System.out.println(output);
            parseJson(output);
        }

        conn.disconnect();
    } catch (MalformedURLException e) {

        e.printStackTrace();

    } catch (IOException e) {

        e.printStackTrace();

    }
}

```

3.1.2. Utilização do webservice REST em uma página HTML utilizando javascript

Para realizar acesso ao serviço REST publicado, vamos publicar na aplicação WEB (**AppFrontController**) uma página HTML chamada **rest.html** que referencia um arquivo javascript (**rest.js**) e uma página de estilo (**styles.css**).

[rest.html](#)

```

<!DOCTYPE HTML>

<html>

<head>

<title>Usuario Rest</title>

<link rel="stylesheet" href="css/styles.css" />

</head>

<body>

```

```
<input type="text" id="idBusca"/>

<button id="btnBusca">Busca usuario</button>

<button id="btnAdiciona">Novo usuario</button>


<form id="usuarioForm">

<label>Id:</label>
<input id="usuarioId" name="id" type="text" disabled />

<label>Name:</label>
<input type="text" id="name" name="name" required>

<label>Family Name:</label>
<input type="text" id="familyName" name="familyName"/>

<button id="btnSave">Save</button>

</form>

<script src="js/jquery-1.7.1.min.js"></script>
<script src="js/rest.js"></script>

</body>
</html>
```

rest.js

```
var rootURL = "http://localhost:8080/AppFrontController/GerenciamentoUsuarios/usuarios";

var usuarioAtual;

$('#btnBusca').click(function() {
    busca($('#idBusca').val());
    return false;
});
```

```
$('#idBusca').keypress(function(e){  
    if(e.which == 13) {  
        busca($('#idBusca').val());  
        e.preventDefault();  
        return false;  
    }  
});
```

```
$('#btnAdiciona').click(function() {  
    novoUsuario();  
    return false;  
});
```

```
$('#btnSave').click(function() {  
    if ($('#usuarioId').val() == '')  
        salvaUsuario();  
    return false;  
});
```

```
function busca(idBusca) {  
    if (idBusca == '')  
        findAll();  
    else  
        buscaPorId(idBusca);  
}
```

```
function novoUsuario() {  
    usuarioAtual = {};  
    renderizaDetalhes(usuarioAtual);  
}
```

```
function buscaPorId(idBusca) {
```

```
console.log('buscaPorId: ' + idBusca);
$.ajax({
    type: 'GET',
    url: rootURL + '/usuario/' + idBusca,
    dataType: "json",
    success: function(data){
        //$('#btnDelete').show();
        console.log('Sucesso buscaPorId: ' + data.name);
        usuarioAtual = data;
        renderizaDetalhes(usuarioAtual);
    },
    error: function(data){
        console.log('Error: ');
    }
});
}
```

```
function salvaUsuario() {
    console.log('salvaUsuario');
    $.ajax({
        type: 'PUT',
        contentType: 'application/xml',
        url: rootURL+'/novousuario',
        dataType: "xml",
        data: formToXML(),
        success: function(data){
            $('#usuarioId').val(data.id);
        }
    });
}
```

```

function renderizaDetalhes(usuario) {
    $('#usuarioId').val(usuario.id);
    $('#name').val(usuario.name);
    $('#familyName').val(usuario.familyName);
}

function formToXML() {
    var usuarioId = $('#usuarioId').val();
    if (usuarioId === "") usuarioId='1';
    var name = $('#name').val();
    var familyName=$('#familyName').val();

    return '<usuario><id>' + usuarioId + '</id><name>' + name + '</name><familyName>' +
familyName + '</familyName></usuario>';
}

```

styles.css

```

* {
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
    font-size: 18px;
}

.header {
    padding-top: 5px;
}

.leftArea {
    position: absolute;
    left: 10px;
    top: 70px;
    bottom: 20px;
    width: 260px;
    border:solid 1px #CCCCCC;
    overflow-y: scroll;
}

```



```
.mainArea {  
  position: absolute;  
  top: 70px;  
  bottom: 20px;  
  left: 300px;  
  overflow-y: scroll;  
  width: 300px;  
}
```

```
.rightArea {  
  position: absolute;  
  top: 70px;  
  bottom: 20px;  
  left: 650px;  
  overflow-y: scroll;  
  width: 270px;  
}
```

```
ul {  
  list-style-type: none;  
  padding-left: 0px;  
  margin-top: 0px;  
}
```

```
li a {  
  text-decoration: none;  
  display: block;  
  color: #000000;  
  border-bottom: solid 1px #CCCCCC;  
  padding: 8px;  
}
```

```
li a: hover {  
  background-color: #4B0A1E;
```

```
color: #BA8A92;
}

input, textarea {
    border: 1px solid #ccc;
    min-height: 30px;
    outline: none;
}

.mainArea input {
    margin-bottom: 15px;
    margin-top: 5px;
    width: 280px;
}

textarea {
    margin-bottom: 15px;
    margin-top: 5px;
    height: 200px;
    width: 250px;
}

label {
    display: block;
}

button {
    padding: 6px;
}

#idBusca {
    width: 160px;
}
```