Desenvolvimento de Aplicações Java

Plataforma Corporativa

Tutorial

Webservices

Outubro 2016

Sumário

1.Introdução

Introdução e uso de WebServices.

1.1. Instalação do SoapUI

Para instalar o SopaUI, baixe o arquivo **`SoapUI-5.2.1-win32-standalone-bin.zip`** do moodle ou do site do SoapUI (https://www.soapui.org/downloads/open-source.html) para sua pasta **`C:\Temp`**.

Descompacte o arquivo. Navegue até a pasta **`bin`** da instalação do SoapUI e execute o arquivo em lote **`soapui.bat`**.

Pode ser que a variável de ambiente **`JAVA_HOME`** não esteja configurada no seu sistema operacional. Procure pelo diretório base de instalação do JDK e atualize o seu arquivo **`soapui.bat`**, inserindo na 2ª linha o comando:

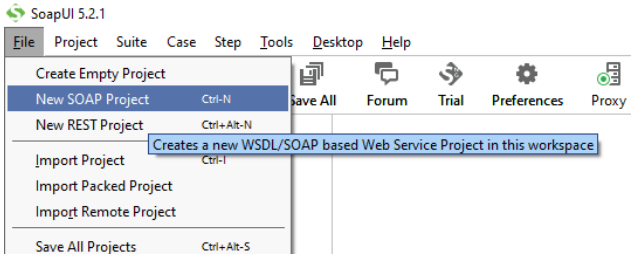
```
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_40
```

Note que você deve atualizar o path para os diretórios da instalação do jdk em sua máquina.

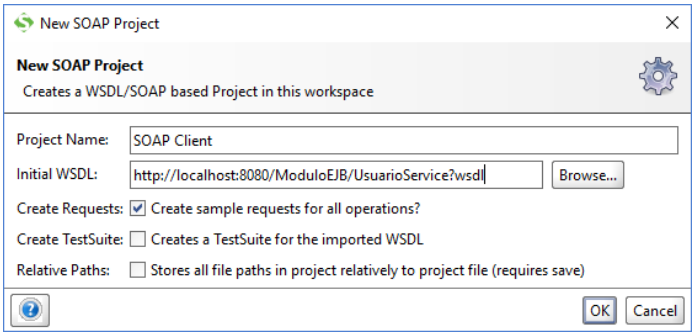
2. Webservices - SOAP

Instalar a aplicação corporativa desenvolvida até o momento.

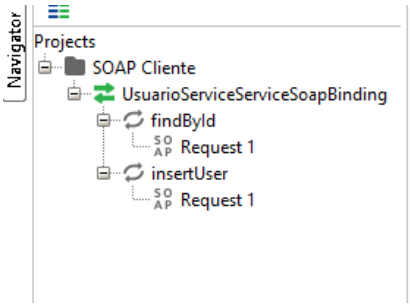
- Anotar a classe **`ejb.entities.Usuario`**, estude e entenda as anotações utilizadas.
- Implementar o bean de sessão sem estado **`ejb.services.UsuarioService`**, estude e entenda as anotações utilizadas.
- Limpar e construir.
- Implantar no Wildfly.
- Abra e entenda o arquivo **`wsdl`**. Abra este arquivo no seu navegador utilizando a URL: http://localhost:8080/ModuloEJB/UsuarioService?wsdl .
- Note que se você criou o seu módulo EJB com outro nome e sua classe de serviço também com outro nome, será necessário redefinir a URL.
- Utilizar o SOAP UI para acessar o serviço publicado. Abrir o SoapUI e criar um projeto SOAP.
    - Clique em Files -> New SOAP Project



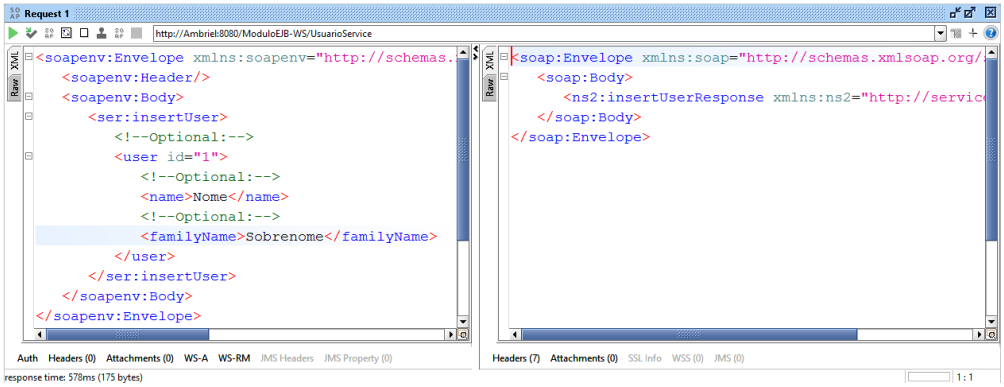
    - De um nome ao seu projeto e indique a URL para o seu wsdl e clique em OK.



    - Seu projeto foi criado



    - Note que há dois **`endpoints`** definidos, o **`findById`** e o **`insertUser`**, como definido em suas anotações no bean de sessão sem estado **`UsuarioService`**. Clicando em Request 1 do endpoint insertUser, você verá que o SoapUI já abre uma janela com um XML SOAP pre-preenchido para que você insira os valores e envie a requisição HTTP com o conteúdo SOAP para o seu webservice. Na janela à direita, você verifica o resultado da execução. Para enviar a requisição HTTP, clique no botão “play” verde.



- Utilizando o **`endpoint findById`** (abrindo request 1) podemos consultar usuários por **`id`**. Note que como implementamos, o bean de sessão singleton **`UsuarioBean`** define o **`id`** do usuário automaticamente, a partir do número de elementos no mapa. Uma busca pelo id 1, tem o retorno apresentado abaixo:

2.1. Usuario.java

```
package ejb.entities;


import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.SequenceGenerator;

import javax.persistence.Table;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlAttribute;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;


@XmlRootElement(name = "usuario")

@XmlAccessorType(XmlAccessType.FIELD)

@Entity

@Table(name="tb_usuario")

public class Usuario {


    @XmlAttribute

    @Id

    @Column(name="usuario_id")

    @SequenceGenerator(name="usuarioGenerator",
sequenceName="usuario_id_sequence",allocationSize=1)

    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="usuarioGenerator")

    private int id;


    @XmlElement

    @Column(name="nome")

    private String nome;


    @XmlElement
```

```
@Column(name="sobrenome")
private String sobrenome;

@XmlElement
@Column(name="login")
private String login;

@XmlElement
@Column(name="hash")
private String hash;

    public Usuario() {
}
public Usuario(int id, String nome, String sobrenome,String login, String hash) {
    this.id = id;
    this.nome = nome;
    this.sobrenome = sobrenome;
    this.login=login;
    this.hash=hash;
}
public String getNome() {
    return nome;
}
public void setNome(String nome) {
    this.nome = nome;
}
public String getSobrenome() {
    return sobrenome;
}
public void setSobrenome(String sobrenome) {
    this.sobrenome = sobrenome;
}
public void setId(int id) {
    this.id = id;
}
```

```
    public int getId() {
        return this.id;
    }
    public String getLogin() {
        return login;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public String getHash() {
        return hash;
    }
    public void setHash(String hash) {
        this.hash = hash;
    }
    public String toString() {
        StringBuffer sbResult = new StringBuffer();
        sbResult.append("id = ");
        sbResult.append(id);
        sbResult.append(", nome = ");
        sbResult.append(nome);
        sbResult.append(", sobrenome = ");
        sbResult.append(sobrenome);
        return sbResult.toString();
    }

}
```

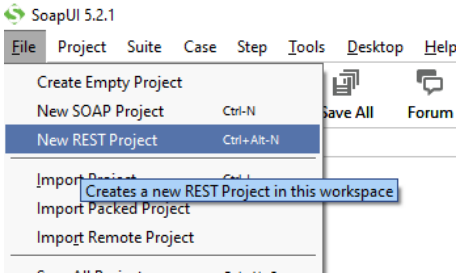
2.2. UsuarioService.java

```
package ejb.services;


import ejb.beans.UsuarioBean;
import ejb.entities.Usuario;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;


@Stateless
@LocalBean
@WebService
public class UsuarioService {


    @EJB
    UsuarioBean ub;


    @WebMethod(operationName = "findById")
    public Usuario buscaUsuarioPorId(@WebParam(name = "id") final int id) {
        Usuario u = ub.buscaUsuarioPorId(id);
        return u;
    }
    @WebMethod(operationName = "insertUser")
    public void insereUsuario(@WebParam(name = "user") final Usuario u) {
        ub.criaUsuario(u);
    }
}
```

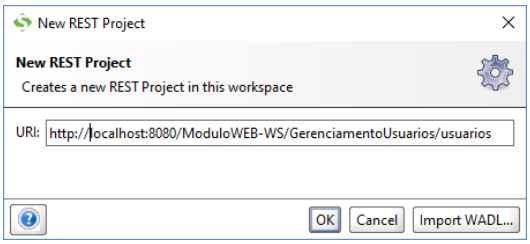
3. Webservices - REST

No projeto corporativo.

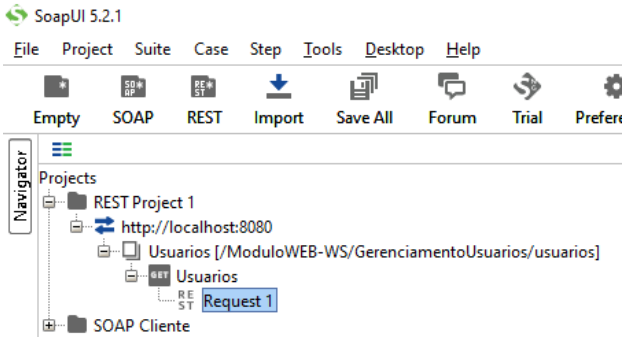
- Implementar o bean de sessão sem estado **`ejb.services.UsuarioRestService`** estude e entenda as anotações utilizadas. Note que é um bean corporativo, portanto deve ser criado no **`ModuloEJB.`**
- Implementar a classe **`mack.rest.AppRest`** no **`AppFrontController`**.
- Limpar e construir.
- Implantar no Wildfly a aplicação corporativa.
- A URL base do webservice REST será:
    - http://localhost:8080/AppFrontController/GerenciamentoUsuarios/usuarios .
- Observe que temos na URL o nome da aplicação WEB, o nome da aplicação REST (anotação utilizada na classe **`AppRest.java`**) e a anotação de path utilizada na classe **`UsuarioRestService.java`**.
- Utilizar o SOAP UI para acessar o serviço publicado. Abrir o SoapUI e criar um projeto REST.
    - Clique em Files -> New REST Project



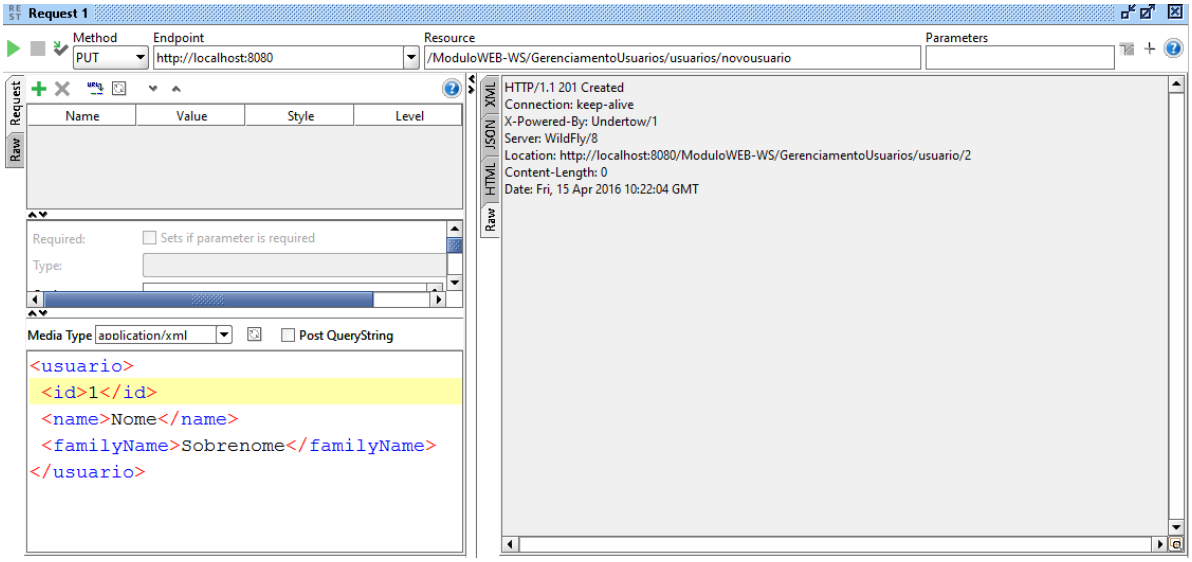
    - Indique a URL base para o webservice REST e clique em OK.



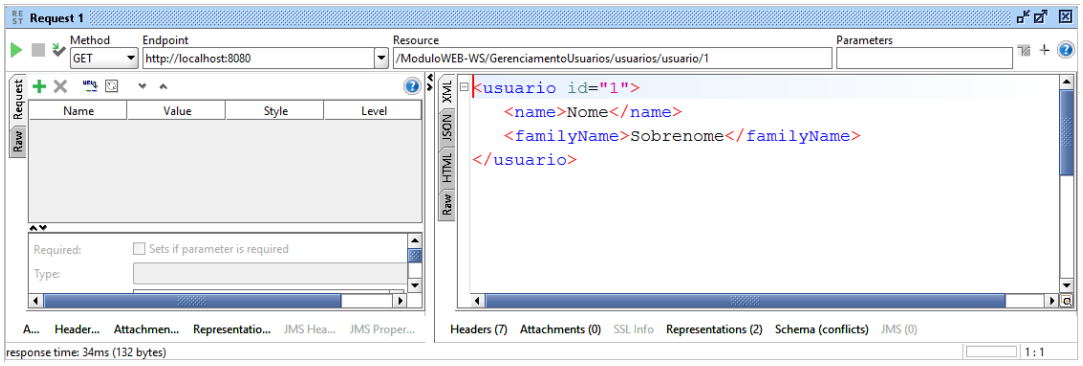
    - Seu projeto foi criado



    - Abra Resquest 1. Escolha o método PUT, complemente o recurso, adicionando no final /novousuario, selecione Media Type como application/xml e preencha com o XML de um novo usuário. Envie a requisição clicando no botão “play” verde. Selecione no painel de resposta o formato raw e verifique se a resposta HTTP é a 201 (Created).



- Selecione agora o método GET, modifique Resource para: /AppFrontController/GerenciamentoUsuarios/usuarios/usuario/1 e clique no botão “play” verde. No painel de resposta, selecione XML. Verifique se o usuário com id=1 foi recuperado.



- Abra o seu navegador e experimente abrir a URL http://AppFrontController/GerenciamentoUsuarios/usuarios/usuario/1 .

3.1. UsuarioRestService.java

```
package ejb.services;


import ejb.beans.UsuarioBean;
import ejb.entities.Usuario;
import java.io.StringReader;
import java.net.URI;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.WebApplicationException;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;



@Stateless
@LocalBean
@Path("/usuarios/")
public class UsuarioRestService {
    @EJB
    UsuarioBean ub;

    @GET
    @Path("/usuario/{id}")
    @Produces(MediaType.APPLICATION_XML)
    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;
    }

    @Path("/novousuario")
    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public Response insereUsuario(String usuarioXml) {
        int id = 0;
        try {
            JAXBContext jc = JAXBContext.newInstance(Usuario.class);
            Unmarshaller u = jc.createUnmarshaller();
            StringReader reader = new StringReader(usuarioXml);
            Usuario usuario = (Usuario) u.unmarshal(reader);
            Usuario user = ub.criaUsuario(usuario);
            id = user.getId();
        } catch (JAXBException ex) {
            throw new WebApplicationException (Response.Status.INTERNAL_SERVER_ERROR);
        }
        return Response.created(URI.create("/usuario/"+id)).build();
    }

}
```

3.2. AppRest.java

```
package mack.rest;


import javax.ws.rs.ApplicationPath;

import javax.ws.rs.core.Application;


@ApplicationPath("GerenciamentoUsuarios")

public class AppRest extends Application{


}
```