**PRUEBA JAVA**

Henry Daniel Casas Pira

**Proyecto CLIENTEWS.**

* Lo primero que se modifica es el archivo POM.XML introduciendo las dependencias correspondientes indicando que se van a utilizar las librerías core y media del proyecto jersey, librerías necesarias para que pueda ejecutarse correctamente nuestro cliente.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>mx.com.gm</groupId>

<artifactId>ClienteSgaRestWS</artifactId>

<version>1.0</version>

<packaging>jar</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.glassfish.jersey.core</groupId>

<artifactId>jersey-client</artifactId>

<version>2.25.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-jaxb</artifactId>

<version>2.25.1</version>

</dependency>

</dependencies>

<name>ClienteWS</name>

</project>

* El siguiente paso es crear una Clase Java llamada Persona.java (Usuario).

NOTA: El proyecto dado lo realicé con XML, ya que con JSON obtuve algunos problemas, problemas de codificación.

package domain;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement Con el objetivo de poder enviar y recibir vía XML

public class Persona {

Se indican los atributos que se van a manejar en el Proyecto con sus respectivos Getters y Settlers.

private int idPersona;

private String nombre;

private String apellidoPaterno;

private String apellidoMaterno;

private String email;

private String telefono;

public int getIdPersona() {

return idPersona;

}

public void setIdPersona(int idPersona) {

this.idPersona = idPersona;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getApellidoPaterno() {

return apellidoPaterno;

}

public void setApellidoPaterno(String apellidoPaterno) {

this.apellidoPaterno = apellidoPaterno;

}

public String getApellidoMaterno() {

return apellidoMaterno;

}

public void setApellidoMaterno(String apellidoMaterno) {

this.apellidoMaterno = apellidoMaterno;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getTelefono() {

return telefono;

}

public void setTelefono(String telefono) {

this.telefono = telefono;

}

@Override

public String toString() {

return "Persona{" + "idPersona=" + idPersona + ", nombre=" + nombre + ", apellidoPaterno=" + apellidoPaterno + ", apellidoMaterno=" + apellidoMaterno + ", email=" + email + ", telefono=" + telefono + '}';

}

}

* Posteriormente agregamos nuestro cliente en una nueva Clase Java.

package test;

import java.util.List;

import domain.Persona;

import javax.ws.rs.client.Client;

import javax.ws.rs.client.ClientBuilder;

import javax.ws.rs.client.Entity;

import javax.ws.rs.client.Invocation;

import javax.ws.rs.client.WebTarget;

import javax.ws.rs.core.GenericType;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

public class TestPersonaServiceRS {

// Definimos todas las variables para poder trabajar con nuestro cliente

Lo primero que defino es la URL Base teniendo en cuenta el puerto de mi servidor, como base para las llamadas a el web service, posteriormente defino una variable Cliente que se importa del paquete javax.ws.rs-client. Siguiente tengo la variable llamada WebTarget para poder llamar los métodos del web service, se define la clase persona, un listado de personas(usuarios) , la variable invocationBuilder para mandar a llamar el ws, finalmente la variable response la cual va ser la respuesta del ws.

private static final String URL\_BASE = "http://localhost:8080/ServidorWs/webservice";

private static Client cliente;

private static WebTarget webTarget;

private static Persona persona;

private static List<Persona> personas;

private static Invocation.Builder invocationBuilder;

private static Response response;

public static void main(String[] args) {

Se crea una instancia del cliente del ws, se crea un Nuevo cliente para poder trabajar para mandar a llamar el ws.

cliente = ClientBuilder.newClient();

Como siguiente paso puedo leer una persona con el método get indicando en el target la url base e indexando el path de /personas.

webTarget = cliente.target(URL\_BASE).path("/personas");

Se proporciona un idPersona valido para poder realizar una prueba, adjuntándolo al método PATH, de igual manera realizando una petición indicando el tipo de dato, en este caso un XML.

persona = webTarget.path("/25").request(MediaType.APPLICATION\_XML).get(Persona.class);

Imprimiendo esta persona entregada por el metodo GET.

System.out.println("Persona recuperada: " + persona);

Basicamente en este apartado se realize la llamada mediante el metodo de encontrar persona por id formando un URI.

Se Leen todas las personas con el metodo get con readEntity de tipo List<>, haciendo la petición mandando a llamar con el metodo GET.

personas = webTarget.request(MediaType.APPLICATION\_XML).get(Response.class).readEntity(new GenericType<List<Persona>>(){});

System.out.println("\nPersonas recuperadas:");

imprimirPersonas(personas);

Posteriormente se Agrega una persona con el metodo post haciendo petición de tipo XML

Persona nuevaPersona = new Persona();

nuevaPersona.setNombre("Daniel");

nuevaPersona.setApellidoPaterno("Casas");

nuevaPersona.setApellidoPaterno("Pira");

nuevaPersona.setEmail("hdaniel0519@mail.com");

invocationBuilder = webTarget.request(MediaType.APPLICATION\_XML);

response = invocationBuilder.post(Entity.entity(nuevaPersona, MediaType.APPLICATION\_XML));

Con la variable invocationbuilder lo que hago es mandar a llamar el método post y proporcionando una entidad nueva indicando el objeto que he creado anteriormente y el tipo de información que voy a estar enviando.

System.out.println("");

Imprimiendo en la consola la respuesta solicitando el status al objeto response.

System.out.println(response.getStatus());

Recupero la persona recien agregada para despues modificarla y al final eliminarla

Persona personaRecuperada = response.readEntity(Persona.class);

System.out.println("Persona agregada: " + personaRecuperada);

Posteriormente se Modifica una persona con el metodo put

Persona personaModificar = personaRecuperada;

personaModificar.setApellidoMaterno("CambioApeMat");

String pathId = "/" + personaModificar.getIdPersona(); Genero el pathid concatenando el id del objeto que se va mejorar.

}

Por medio del objeto webtarget mando a llamar el método path agregando el id respectivo e indicando un XML,

invocationBuilder = webTarget.path( pathId ).request(MediaType.APPLICATION\_XML);

response = invocationBuilder.put(Entity.entity(personaModificar, MediaType.APPLICATION\_XML));

System.out.println("");

System.out.println(response.getStatus());

System.out.println("Persona modificada: " + response.readEntity(Persona.class));

Eliminar una persona de forma similar a lo anterior.

Persona personaEliminar = personaRecuperada;

String pathEliminarId = "/" + personaEliminar.getIdPersona();

invocationBuilder = webTarget.path( pathEliminarId ).request(MediaType.APPLICATION\_XML);

response = invocationBuilder.delete();

System.out.println("");

System.out.println(response.getStatus());

System.out.println("Persona Eliminada: ");

}

Por último se agrega el método imprimir personas que básicamente es la iteración de cada una de las personas del listado respectivo.

private static void imprimirPersonas(List<Persona> personas){

for(Persona persona: personas){

System.out.println("Persona:" + persona);

}

}

}

**Proyecto SERVIDORWS.**

En el proyecto se expone el metodo listarPersonas, resgistrarPersona, modificarPersona, eliminar Persona del EJB con la ayuda del API JAX-RS.

Creo la clase PersonaServieRS para exponer los métodos anteriormente mencionados via RES web services.

package mx.com.gm.sga.servicio.rest;

import java.util.List;

import javax.ejb.EJB;

import javax.ejb.Stateless;

import javax.ws.rs.Consumes;

import javax.ws.rs.DELETE;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.Response.Status;

import mx.com.gm.sga.domain.Persona;

import mx.com.gm.sga.servicio.PersonaService;

En esta clase se agregan las anotación de JaxRs en primera medida, agregando como PATH /personas, indicando que es un EJB con @Satateless.

@Path("/personas")

@Stateless

public class PersonaServiceRS {

@EJB También se puede utilizar @inject

private PersonaService personaService;

Posteriormente se empiezan agregar los métodos correspondientes.

@GET Método Http Indicando que pueden producir datos con JSON o XML.

@Produces(value={MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public List<Persona> listarPersonas() {

return personaService.listarPersonas();

}

Defino el Método encontrar usuario por ID con GET de igual manera agregando un ID con la anotación @PATH

@GET

@Produces(value={MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

@Path("{id}") hace referencia a /personas/{id}

public Persona encontrarPersonaPorId(@PathParam("id") int id) {

return personaService.encontrarPersonaPorId(new Persona(id));

}

En este caso, el método agregarPersona puede producir y consumir datos de tipo JSON o XML

@POST

@Produces(value={MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

@Consumes(value={MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public Response agregarPersona(Persona persona) { Se agrega una respuesta para indicar si se ejecuta de manera correcta la inserción

try { Agregando una respuesta con el objeto response.

personaService.registrarPersona(persona);

return Response.ok().entity(persona).build();

} catch (Exception e) { Dado caso que se genere un error

return Response.status(Status.INTERNAL\_SERVER\_ERROR).build();

}

}

@PUT Método para modificar una usuario.

@Produces(value={MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

@Consumes(value={MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

@Path("{id}")

public Response modificarPersona(@PathParam("id") int id, Persona personaModificada) {

try {

Persona persona = personaService.encontrarPersonaPorId(new Persona(id));

if (persona != null) {

personaService.modificarPersona(personaModificada);

return Response.ok().entity(personaModificada).build();

} else {

return Response.status(Status.NOT\_FOUND).build(); Caso en que no se encuentre un objeto

}

} catch (Exception e) {

return Response.status(Status.INTERNAL\_SERVER\_ERROR).build();

}

}

@DELETE

@Path("{id}")

public Response eliminarPersonaPorId(@PathParam("id") int id) {

try {

personaService.eliminarPersona(new Persona(id));

return Response.ok().build(); //Se construte la respuesta como anteriormente mencionado.

} catch (Exception e) {

return Response.status(404).build();

}

}

}

Clase Persona.java

package mx.com.gm.sga.domain;

import java.io.Serializable;

import java.util.List;

import javax.persistence.Basic;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlTransient;

@Entity

@Table(name = "persona")

@NamedQueries({

@NamedQuery(name = "Persona.findAll", query = "SELECT p FROM Persona p")

, @NamedQuery(name = "Persona.findByIdPersona", query = "SELECT p FROM Persona p WHERE p.idPersona = :idPersona")

, @NamedQuery(name = "Persona.findByNombre", query = "SELECT p FROM Persona p WHERE p.nombre = :nombre")

, @NamedQuery(name = "Persona.findByApellidoPaterno", query = "SELECT p FROM Persona p WHERE p.apellidoPaterno = :apellidoPaterno")

, @NamedQuery(name = "Persona.findByApellidoMaterno", query = "SELECT p FROM Persona p WHERE p.apellidoMaterno = :apellidoMaterno")

, @NamedQuery(name = "Persona.findByEmail", query = "SELECT p FROM Persona p WHERE p.email = :email")

, @NamedQuery(name = "Persona.findByTelefono", query = "SELECT p FROM Persona p WHERE p.telefono = :telefono")})

@XmlRootElement

@XmlAccessorType(XmlAccessType.FIELD)

public class Persona implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "id\_persona")

private Integer idPersona;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 45)

@Column(name = "nombre")

private String nombre;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 45)

@Column(name = "apellido\_paterno")

private String apellidoPaterno;

@Size(max = 45)

@Column(name = "apellido\_materno")

private String apellidoMaterno;

@Basic(optional = false)

@NotNull

@Size(min = 1, max = 45)

@Column(name = "email")

private String email;

@Size(max = 45)

@Column(name = "telefono")

private String telefono;

@XmlTransient

@OneToMany(mappedBy = "persona", cascade = CascadeType.ALL)

private List<Usuario> usuarios;

public Persona() {

}

public Persona(Integer idPersona) {

this.idPersona = idPersona;

}

public Persona(String nombre, String apePaterno, String apeMaterno,

String email, String telefono) {

this.nombre = nombre;

this.apellidoPaterno = apePaterno;

this.apellidoMaterno = apeMaterno;

this.email = email;

this.telefono = telefono;

}

public Persona(Integer idPersona, String nombre, String apellidoPaterno, String email) {

this.idPersona = idPersona;

this.nombre = nombre;

this.apellidoPaterno = apellidoPaterno;

this.email = email;

}

public Persona(Integer idPersona, String nombre, String apellidoPaterno, String apellidoMaterno,

String email, String telefono) {

this.idPersona = idPersona;

this.nombre = nombre;

this.apellidoPaterno = apellidoPaterno;

this.apellidoMaterno = apellidoMaterno;

this.email = email;

this.telefono = telefono;

}

public Integer getIdPersona() {

return idPersona;

}

public void setIdPersona(Integer idPersona) {

this.idPersona = idPersona;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getApellidoPaterno() {

return apellidoPaterno;

}

public void setApellidoPaterno(String apellidoPaterno) {

this.apellidoPaterno = apellidoPaterno;

}

public String getApellidoMaterno() {

return apellidoMaterno;

}

public void setApellidoMaterno(String apellidoMaterno) {

this.apellidoMaterno = apellidoMaterno;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getTelefono() {

return telefono;

}

public void setTelefono(String telefono) {

this.telefono = telefono;

}

public List<Usuario> getUsuarios() {

return usuarios;

}

public void setUsuarios(List<Usuario> usuarios) {

this.usuarios = usuarios;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idPersona != null ? idPersona.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Persona)) {

return false;

}

Persona other = (Persona) object;

if ((this.idPersona == null && other.idPersona != null) || (this.idPersona != null && !this.idPersona.equals(other.idPersona))) {

return false;

}

return true;

}

@Override

public String toString() {

return "Persona{" + "idPersona=" + idPersona + ", nombre=" + nombre + ", apellidoPaterno=" + apellidoPaterno + ", apellidoMaterno=" + apellidoMaterno + ", email=" + email + ", telefono=" + telefono + '}';

}

}

Configuración del archivo web.xml

Se configura el servlet de jersey, indicando el nombre del servlet e indicando la clase de este para manejar los web services de javaxrs, posteriormente se indica el paquete donde tengo los ws, De igual manera se indica que este servlet se debe levantar al inicio de la aplicación y posteriormente indicando cual es el servlet mapping de jersey .

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<context-param>

<param-name>javax.faces.PROJECT\_STAGE</param-name>

<param-value>Development</param-value>

</context-param>

<context-param>

<param-name>javax.faces.FACELETS\_REFRESH\_PERIOD</param-name>

<param-value>0</param-value>

</context-param>

<context-param>

<param-name>primefaces.THEME</param-name>

<param-value>cupertino</param-value>

</context-param>

<error-page>

<exception-type>javax.faces.application.ViewExpiredException</exception-type>

<location>/faces/index.xhtml</location>

</error-page>

<welcome-file-list>

<welcome-file>faces/index.xhtml</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>Faces Servlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Faces Servlet</servlet-name>

<url-pattern>\*.faces</url-pattern>

<url-pattern>/faces/\*</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>JerseyWebApplication</servlet-name>

<servlet-class>

org.glassfish.jersey.servlet.ServletContainer

</servlet-class>

<init-param>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>mx.com.gm.sga.servicio.rest</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>JerseyWebApplication</servlet-name>

<url-pattern>/webservice/\*</url-pattern>

</servlet-mapping>

</web-app>