IMPLEMENTACION DEL SOFTWARE – SISTEMA MEDIADOR

Para la implementación del sistema mediador se utilizó el IDE Jbuilder 8, que da soporte para la creación y despliegue de Servicios Web y facilidades para el desarrollo de aplicaciones Web basadas en servlets y JSPs.

El sistema mediador esta compuesto por 2 módulos:

- El servicio Web de mediación de peticiones.
- El modulo de administración del servicio Web.

El servicio Web fue desarrollado a base de clases Java, utilizando para su despliegue el API Apache Axis. Por medio de Jbuilder se generó el archivo war con los descriptores necesarios para montar el servicio en el servidor Tomcat.

El modulo de administración fue desarrollado de la siguiente forma:

- Las interfaces de usuario fueron implementadas en JSPs.
- La lógica necesaria para la administración fue desarrollada en servlets, beans y clases java.

Algunas de las librerías utilizadas en el proceso de implementación del sistema mediador son:

- JDOM: API utilizada para el parseo y la creación de documentos XML.
- Geotools: API utilizada para la modificación de los archivos de extensión dbf, con el fin de agregarle la información obtenida de las múltiples fuentes de datos.

- Mysql-connector-java: API que permite la conexión con el motor de base de datos MySql.
- Apache Axis: Implementación de la especificación SOAP y WSDL.

A continuación se hace una descripción de las clases que implementan la lógica del sistema mediador.

CLASE: Control_servicio

Descripción: Clase encargada de controlar el procesamiento de las peticiones realizadas por los clientes. Invoca los procesos de validación de usuarios, validación de peticiones, ejecución de peticiones y estructuración de respuestas

Métodos

procesar_peticion_mapa

public Vector **procesar_peticion_mapa**(Vector parametros)

Descripción: Se encarga de recibir la petición de mapas del cliente, extraer los diferentes parámetros e invocar los procesos involucrados con la resolución de dicha petición

.

Parametros:

parametros - vector que contiene los parámetros de la petición realizada por el cliente

Retorna:

un vector con la repuesta estructurada

---

conectar_BD

private void **conectar_BD**()

Descripción: Se encarga de realizar la conexión con la base de datos

---

validar_usuario

private boolean **validar_usuario**(String login, String password)

Descripción: Se encarga realizar la validación de los usuarios del sistema.

Parámetros:

login - Login del usuario que va a ser validado
password - Password del usuario que va a ser validado

Retorna:

booleano que determina si es un usuario valido.

CLASE: Gestion_MetaData

Descripción: Esta clase se encarga de todos los procesos de gestión referentes a los metadatos del sistema. Entre estos procesos encontramos el parseo de los documentos de metadatos de cada fuente, así como el documento WSDL que describe los servicios de cada una de ellas. También genera los documentos de metadatos para los clientes, tanto el documento general de información, así como el documento que clasifica por organización.

Métodos

conectar_BD

private void **conectar_BD**()

Descripción: Se encarga de realizar la conexión con la base de datos

---

parsear_xml

public boolean **parsear_xml**(String nombreF)

Descripción: parsea los documentos xml para la obtención de los meta datos de las fuentes. Primero procesa las etiquetas mapas, obteniendo las zonas y los temas por zona. Si cada tema tiene varios mapas los obtiene y mapea en la BD verificando primero que no existan ya. Posteriormente procesa elementos contenidos en el elemento Informacion

Parámetros:

nombreF - Nombre del documento xml que se va a parsear

---

parsear_tema

private void **parsear_tema**(List pp)

Descripción: Este método se encarga de obtener la información contenida en el elemento tema.

Parámetros:

pp - Lista de los temas de una zona

---

parsear_mapa

private void **parsear_mapa**(List pp)

Descripción : Se encarga de parsear la información contenida dentro del elemento en el documento de metadatos

Parámetros:

pp - Lista con los mapas contenidos en cada una de las etiquetas

---

parsear_wsdl

public boolean **parsear_wsdl**(String Filename)

Descripción: Se encarga de parsear los documentos wsdl que contienen la

descripción de los métodos publicados por las fuentes de datos y los puntos de acceso al servicio

Parámetros:

Filename - Nombre del documento WSDL a parsear

Retorna:

---

generar_metadata

public boolean **generar_metadata**()

Descripción: Este método se encarga de generar el documento xml que será entregado al cliente con la información disponible en las fuentes de datos adscritas al sistema.

generar_metadata_organizacion

public boolean **generar_metadata_organizacion**()

Descripción: Método encargado de generar el documento de metadatos con la información disponible, clasificándola por organización.

consultar

private boolean **consultar**(String sqla, String variable)

Descripción : Este método se encarga de ejecutar las sentencias de consulta de identificadores de las tablas de la base de datos

Parámetros:

sqla - : Sentencia sql a ejecutar

variable - : campo a obtener de la base de datos

Retorna:

booleano que indica el éxito del proceso

---

parsear_Informacion

private boolean **parsear_Informacion**(Element raiz)

Descripción: Este método se encarga de realizar el proceso de parseo de la información contenida en el elemento del documento de metadatos

Parámetros:

raiz - : Elemento raiz del documento a parsear

Retorna:

: booleano que indica el exito de la operacion

CLASE: Gestion_Petición

Descripción: Se encarga de gestionar las peticiones de los clientes. Realiza la validación de las mismas determinando que fuentes contienen información que la puedan resolver.

Métodos

validar_peticion_mapa

public boolean **validar_peticion_mapa**
(Control_Servicio control,
String zona,
String tema,
String N_Mapa,
String ParametroB,
String Valor_Parametro,
Vector consulta)

Descripción: Se encarga realizar la validación de las peticiones

Parámetros:

zona - Identificador de la zona donde se encuentra el mapa solicitado

tema - Nombre del tema al que pertenece el mapa solicitado

N_Mapa - Nombre del mapa solicitado

ParametroB - Parámetro por el cual se va a hacer la búsqueda de

información

Valor_Parametro - Valor del parámetro de búsqueda

consulta - Vector con los parámetros solicitados

control - clase Control_Servicio

Retorna:

booleano que me determina si la petición es valida.

---

ejecutar_consulta

private boolean **ejecutar_consulta**(String sql)

Descripción: Se encarga de ejecutar las consulas a la base de datos

Parámetros:

sql - Consulta a ser ejecutada.

Retorna:

booleano que indica el exito de la ejecucion de la consulta

---

conectar_BD

private void **conectar_BD**()

Descripción: Se encarga de realizar la conexion a la base de datos del sistema.

---

obtener_info_servicio

public Vector **obtener_info_servicio**(int id_fuenteS)

Descrición: Este metodo se encarga de obtener de la base de datos del sistema los parametros necesarios para la invocacion de los servicios de consulta.

Parámetros:

id_fuenteS - Identificador de la fuente

Retorna:

Vector con los parametros solicitados.

CLASE: BeanFuente

Descripción: Este bean permite la gestión la información de las fuentes adscritas al sistema.

Atributos

Conector: Bean encargado de la conexión a la base de datos.

nombre: Nombre de la fuente de datos

organización: Nombre de la organización a la que pertenece la fuente de datos

URL_WSDL: URL donde se encuentra el documento WSDL que describe el servicio Web que permite la consulta de información a la fuente de datos remota

URL_XML: URL donde se encuentra el documento de meta datos de la fuente de datos remota

Atributos obtenidos del documento WSDL

location

targetNameSpace

---

serviceName

---

portBinding

---

portName

Métodos

adicionarFuente

public int **adicionarFuente**()

Descripción: Adiciona una nueva fuente de datos al sistema

Retorna:

Un entero con el estado de la ejecución de la inserción.

---

listarFuentes

public Vector **listarFuentes**()

Descripción: Lista las fuentes adscritas al sistema

Retorna:

vector con las fuentes adscritas al sistema

---

buscarFuente

public boolean **buscarFuente**()

Descripción: Busca una fuente en la base de datos

Retorna:

true si encuentra, false de lo contrario.

---

eliminarFuente

public int **eliminarFuente**()

Descripción: Elimina una fuente de datos del sistema, junto con todos los meta datos que se hayan mapeado en la BD pertenecientes a esta fuente.

Retorna:

Entero con el estado de la ejecución de la eliminación

---

modificarF

public int **modificarF**(String nn)

Actualiza la información de una fuente de datos registrada en el sistema.

Parámetros:

Nombre - de la fuente a actualizar

Retorna:

Entero con el estado de la ejecución de la actualización.

CLASE: BeanConector

Descripción: Este bean permite la conexión con la base de datos del sistema y la ejecución de consultas sobre ella.

Atributos

url

private String **url**

URL de la conexión JDBC

---

login

private String **login**

Login de la Conexión JDBC

---

password

private String **password**

Password de la Conexión JDBC

---

driver

private String **driver**

Driver de la Conexión JDBC

---

cx

private Connection **cx**

Objeto del tipo Connection para manejar la conexión JDBC

---

query

private java.sql.Statement **query**

Objeto Statement para manejar las consultas

---

resultado

public ResultSet **resultado**

Objeto ResultSet para manejar los resultados de la ejecución de consultas tipo Select

---

error

private String **error**

Cadena donde se guardan los errores generados por las excepciones JDBC

Métodos

setUrl

public void **setUrl**(String valor)

Fija la URL utilizada en la conexión

Parámetros:

valor - Cadena que especifica la URL para conectarse

---

getUrl

public String **getUrl**()

Retorna la URL utilizada para la conexión

---

setLogin

public void **setLogin**(String valor)

Fija el Login usado en la Conexión JDBC

Parámetros:

valor - Login usado para la conexión

---

getLogin

public String **getLogin**()

Obtiene el login utilizado para la conexión JDBC

---

setPassword

public void **setPassword**(String valor)

Permite fijar el password para la conexión JDBC

Parámetros:

valor - Password usado en la conexión

---

getPassword

public String **getPassword**()

Obtiene el password utilizado para la conexión

---

setDriver

public void **setDriver**(String valor)

Fija el Driver usado para la conexión JDBC

Parámetros:

valor - Cadena que especifica el Driver

---

getDriver

public String **getDriver**()

Retorna el driver utilizado para la conexión JDBC

---

getError

public String **getError**()

Recupera la cadena de un mensaje de error cuando ha ocurrido una expeción en alguna operación con JDBC

Retorna:

Cadena de error (mensaje de error)

---

conectar

public int **conectar**()

Metodo para realizar la conexión a la BD

Retorna:

1 si la conexión fue exitosa

-1 si hubo error

Si hubo error, con getError puede recuperar la cadena de la excepción

---

actualizar

public int **actualizar**(String sql)

Manda a Ejecutar una sentencia SQL del tipo insert,update, delete o cualquiera que no retorne un resultset

Parámetros:

sql - Cadena que tiene la consulta a ejecutar

Retorna:

rs Número de registros afectados

-1 Si ha ocurrido un error

---

consultar

public boolean **consultar**(String sql)

Ejecuta una consulta del tipo select

Parámetros:

sql - Cadena de la sentencia sql tipo select

Retorna:

true Si la consulta se ejecuto correctamente

false Si hay error en la consulta

Ejecutando el método getError sobre el Bean puede recuperar el error generado

---

getEntero

public int **getEntero**(String nomb)

Permite recuperar valores enteros de un registro mantenido por el bean cuando se conoce el nombre del campo

Parámetros:

nomb - Nombre del Campo que se desea recuperar

Retorna:

n Valor del Campo

-1 Si ha ocurrido un error

---

getEntero

public int **getEntero**(int indice)

Permite recuperar valores enteros de un registro mantenido por el bean a partir de la posición dentro de la sentencia SQL (Posiciion inicial =1)

Parámetros:

indice - Posición en la que se encuentra el campo dentro de la sentencia SQL mandada a Ejecutar

Retorna:

n Valor del Campo

-1 Si ha ocurrido Error

---

getCadena

public String **getCadena**(String nomb)

Permite recuperar un valor tipo String de un registro cuando se conoce el nombre del campo

Parámetros:

nomb - Nombre del Campo

Retorna:

S Cadena Recuperada

"Error al recuperar campo" si ha ocurrido un error

---

getCadena

public String **getCadena**(int indice)

Permite recuperar un String de un registro conociendo su posición en la sentencia SQL (Posición Inicial=1)

Parámetros:

indice - Posición donde esta ubicado el campo dentro de la sentencia SQL

Retorna:

S, Cadena con el valor

"Error al recuperar el campo" si ha ocurrido un Error

---

getObjeto

public Object **getObjeto**(String nomb)

Permite recuperar un elemento tipo Objeto a partir del nombre del campo

Parámetros:

nomb - Nombre del Campo

Retorna:

O, Objeto recuperado null, si ha ocurrido un error

El desarrollador debe convertir el objeto al tipo deseado

---

getObjeto

public Object **getObjeto**(int indice)

Permite Recuperar un objeto a partir de la posición del campo en una sentencia SQL

Parámetros:

indice - Posición del Campo dentro de la Sentencia SQL

Retorna:

O, Objeto recuperado null, si ha ocurrido un error. El desarrollador debe convertir el objeto al tipo deseado

---

getFecha

public Date **getFecha**(String nomb)

Permite Recuperar un Objeto tipo Date cuando se conoce el campo

Parámetros:

nomb - Nombre del Campo

Retorna:

D, Dato tipo Date recuperado null, si ha ocurrido un error

---

getFecha

public Date **getFecha**(int indice)

Permite Recuperar un Objeto tipo Date cuando se conoce su posición

Parámetros:

indice - Posición del campo dentro de la sentencia SQL

Retorna:

D, Dato tipo Date recuperado null, si ha ocurrido un error

siguiente

public int **siguiente**()

Permite avanzar al siguiente registro dentro de un resultado

Retorna:

1, si hay más registros

0, si no hay mas registros

-1, si ha ocurrido una excepción

cerrarCx

public boolean **cerrarCx**()

Permite cerrar la conexión con la BD

Retorna:

true, si se cerro adecuadamente

false, si ha ocurrido un error

CLASE: BeanUsuario

Descripción: Este bean permite la gestión de la información relacionada con los usuarios del sistema.

Atributos

conector

private BeanConector **conector**

Objeto encargado de la conexion con la base de datos.

password

private String **password**

Password del usuario que accede al sistema.

---

nombre

private String **nombre**

Nombre del usuario del sistema.

---

login

private String **login**

login del usuario del sistema.

---

tipo

private String **tipo**

tipo de usuario del sistema: Administrador o cliente

---

privilegios

private String **privilegios**

Privilegios que tiene el usuario: Privilegios totales o restringidos

---

descripcion

private String **descripcion**

Breve descripcion del usuario.

Métodos

setTipo

public void **setTipo**(String t)

Fijar el valor del tipo de usuario

Parámetros:

t - El tipo de usuario a fijar

---

setPassword

public void **setPassword**(String pass)

Fijar el valor del password

Parámetros:

pass - El password a fijar

---

setLogin

public void **setLogin**(String log)

Fijar el valor del Login

Parámetros:

log - El login a fijar

---

setNombre

public void **setNombre**(String nomb)

Fijar el valor del nombre

Parámetros:

nomb - El nombre a fijar

---

setPrivilegios

public void **setPrivilegios**(String priv)

Fijar el valor de los privilegios del usuario

Parámetros:

priv - Los privilegios a fijar

---

setDescripcion

public void **setDescripcion**(String desc)

Fijar la descripcion del usuario del sistema

Parámetros:

desc - Descripcion a fijar

---

getTipo

public String **getTipo**()

Obtener el tipo de usuario

Retorna:

tipo

---

getPassword

public String **getPassword**()

Obtener el valor del password

Retorna:

password

---

getPrivilegios

public String **getPrivilegios**()

Obtener los privilegios del usuario

Retorna:

privilegios

---

getDescripcion

public String **getDescripcion**()

Obtener la descripción del usuario.

Retorna:

Descripción

---

getNombre

public String **getNombre**()

Obtener el valor del nombre

Retorna:

nombre

---

getLogin

public String **getLogin**()

Obtener el valor del login del usuario.

Retorna:

Login

---

adicionarU

public int **adicionarU**()

Adicionar un usuario a la BD

Retorna:

el número de campos modificados, debe ser 1 si la insercción es correcta

---

buscarU

public boolean **buscarU**()

Buscar un usuario

Retorna:

true si el usuario fue encontrado, false en caso contrario

---

eliminarU

public int **eliminarU**()

Eliminar un usuario

Retorna:

1 si lo eliminó correctamente, otro valor en caso contrario

---

validarUsuario

public boolean **validarUsuario**()

Valida el usuario que ingresa al sistema.

Retorna:

true si el usuario es valido, false en caso contrario.

---

modificarP

public int **modificarP**()

Modificar un usuario

Retorna:

1 si actualizó correctamente

CLASE: Consultor

Descripción: Permite la consulta de las diferentes fuentes adscritas al sistema y formatea las respuestas de acuerdo a la solicitud del cliente.

Métodos

RevisarMapaData

private boolean **revisarMapaData**(HashMap mapa)

Método que revisa que todos los campos estén dentro del hasmap para poder realizar la búsqueda acertadamente

Parámetros:

mapa - , HashMap con los datos para realizar la búsqueda

Retorna:

verdadero si tiene todos los campos

---

revisarData

private boolean **revisarData**(HashMap data)

Método que revisa que todos los parámetros para la búsqueda de los datos estén bien configurados dentro del hashMap

Parámetros:

data - HashMap para revisar

Retorna:

verdadero si tiene todos los campos necesarios

---

consultar

public HashMap **consultar**(HashMap mapaData,
Vector dataData,
String indice)

Método que realiza la consulta a las diferentes fuentes, recibe los parámetros de las fuentes y además de cual es la fuente que tiene los datos del mapa y cuales la de los datos.

Parámetros:

mapaData - datos para obtener la consulta de un mapa a una fuente

dataData - datos para obtener la consulta de datos a las fuentes

indice - cadena que determina por medio de que parametro se realizara la busqueda

hashmap con el mapa consolidado con las consultas y una lista de errores

MapaData debe llevar:

url del servicio

zona

tema

usuario

password

namespace de la fuente

mapa

dataData debe llevar un vector de hashmap donde cada uno tiene

url del servicio

zona

tema

usuario

password

paramIn= parametros de entrada (HashMap)

paramOut=parametros de salida (Vector)

namespace de la fuente

índice

Cadena que especifica cual es el parámetro que representa la búsqueda

Retorna

HashMap conteniendo:

En el key "mapa" : Byte[] con los datos del mapa creado recientemente

En el key "errores" : Vector con una lista de los errores presentados

CLASE: ClienteWS

Descripción: clase que realiza las peticiones a los servicios web de las fuentes de datos

Atributos

error

String **error** representa la cadena de error si ha ocurrido

Métodos

GetError

public String **getError**()

Devuelve la cadena error

Retorna:

HashMap o nulo si hubo error

getMapa

public HashMap **getMapa**(HashMap dataMap)

Método encargado de obtener los mapas desde una fuente remota

Parámetros:

dataMap - datos de la fuente que contiene el mapa

Retorna:

HashMap o nulo si hubo error

---

getData

public HashMap **getData**(HashMap dataMap)

Metodo encargado de obtener los datos desde una fuente remota

Parámetros:

dataMap - datos de la fuente que contiene el mapa

Retorna:

HashMap o nulo si hubo error

---

getMetadata

public HashMap **getMetadata**(HashMap dataMap)

Método encargado de obtener el archivo de metadatos desde una fuente remota

Parámetros:

dataMap - datos de la fuente que contiene el mapa

Retorna:

hasmap o nulo si hubo error

CLASE: CreadorDBF

Descripción: Clase encargada de la creación de un dbf a partir de varias fuentes formateadas en HashMaps

Métodos

getError

public String **getError**()

Devuelve la cadena que señala el último error

Retorna:

cadena de error

---

crear_dbf

public boolean **crear_dbf**(Vector rxs,
int[] fuentes,
String indice,
String url_map,
String nombre_mapa)

El método recibe los datos y la dirección del dbf y crea el archivo creado.dbf en el directorio que se especifique.

Parámetros:

rxs - Vector que contiene los hashmap resultados de las consultas
fuentes - Arreglo con las longitudes de los resultados de c/u de las fuentes
indice - String representando el indice de la creación
url_map - String que indica donde esta el mapa y donde están los archivos y el dbf por ejemplo c:\mapas
nombre_mapa - String indica el nombre del archivo dbf que del que se obtendrán los datos

Retorna:

true en caso de creación correcta del dbf

Throws:

IOException -
DbfFileException -

---

crear_dbf

public boolean **crear_dbf**(HashMap rx,
int conterRegArray,
String indice)

Método que crea un dbf a partir de los datos obtenidos de un hashMap en el formato obtenido del wrapper y otros datos:

Parámetros:

rx - Datos para la creación del dbf

conterRegArray -

indice – Cadena que representa el indice del dbf

Retorna:

verdadero si lo pudo crear

Throws:

IOException -

DbfFileException -

CLASE: Archiver

Descripción: Clase encargada del manejo de archivos a alto nivel

Constructor

Archiver

public **Archiver**()

Constructor por defecto

Métodos

copiar

public boolean **copiar**(String origen, String destino)

Método para copiar un archivo en otro el archivo de destino no debe existir

ejemplo de uso:

Archiver a= new Archiver();

a.copiar("C:\\lib_java\\hola.txt","C:\\lib_java\\hola1.txt");

Parámetros:

origen - path del archivo desde el que se va a copiar

destino - path del archivo a donde se va a copiar

Retorna:

true si copio false si no pudo copiar

---

borrarFile

public boolean **borrarFile**(String path)

Método para borrar archivos

Parámetros:

path - Archivo a borrar

Retorna:

true si lo pudo borrar false si no lo puede hacer

---

getAbsolutePath

public String **getAbsolutePath**(String Unknown)

Metodo para obtener el Path Absoluto del directorio actual

Parámetros:

Unknown - archivo o directorio en el directorio actual para obtener el path absoluto

Retorna:

Path Absoluto

---

copiarTemporal

public boolean **copiarTemporal**(String origen,
String destino)

Metodo para copiar un archivo en otro y el otro es temporal asi que al terminar la ejecucion desaparcece el archivo de destino no debe existir
ejemplo de uso:
Archiver a= new Archiver();
a.copiar("C:\\lib_java\\hola.txt","C:\\lib_java\\hola1.txt");

Parámetros:

origen - path del archivo desde el que se va a copiar

destino - path del archivo a donde se va a copiar

Retorna:

true si copio false si no pudo copiar

---

getBytesFromFile

public byte[] **getBytesFromFile**(String origen)

Metodo que devuelve un arreglo de bytes desde un archivo o null si hubo un error

Parámetros:

origen - path del archivo de donde se obtienen los bytes

Retorna:

bytes desde el archivo null si hay un error

---

crearDir

public boolean **crearDir**(String path)

Método para crear directorios

Parámetros:

path - nombre del path donde se creara el directorio

Retorna:

true o false dependiendo del éxito de la operación

---

getError

public String **getError**()

Retorna:

cadena con la descripción del error

---

existe

public boolean **existe**(java.util.Vector urls)

Método que verifica la existencia de una serie de archivos

Parámetros:

urls - vector con los paths de los archivos a verificar

Retorna:

true si existen todos false si al menos uno no existe

CLASE: Stub

public class **Stub**
extends org.apache.axis.client.Stub
implements org.Natural.NaturalPortType

Descripción: Clase Stub de los servicios web de las fuentes de datos sirve para hacer peticiones a cualquier fuente de datos

Constructor

Stub

public **Stub**(java.net.URL endpointURL,
javax.xml.rpc.Service service)

Metodo constructor

Parámetros:

endpointURL - del servicio

service - Servicio de la fuente de datos

Throws:

org.apache.axis.AxisFault -

Stub

public **Stub**(javax.xml.rpc.Service service)

Metodo contstructor

Parámetros:

service - Servicio de la fuente de datos

Throws:

org.apache.axis.AxisFault -

Métodos

getMetadata

public java.util.HashMap **getMetadata**(java.lang.String usuario,
java.lang.String password,
String namespace)

Parámetros:

usuario - de la fuente de datos

password - de la fuente de datos

namespace - de la fuente de datos

Retorna:

HashMap con los resultados de la consulta

Throws:

java.rmi.RemoteException -

---

getMetadata

public java.util.HashMap **getMetadata**(java.lang.String usuario,
java.lang.String password)

Parámetros:

usuario - de la fuente de datos

password - de la fuente de datos

Retorna:

HashMap con los archivos metadatos

Throws:

java.rmi.RemoteException -

---

getData

public java.util.HashMap **getData**(java.lang.String zona,
java.lang.String tema,
java.lang.String usuario,
java.lang.String password,
java.util.HashMap paramIn,
java.util.Vector paramOut)

Método que realiza la invocación de la consulta de datos al servicio web

Parámetros:

zona - de la consulta a la fuente de datos
tema - de la consulta a la fuente de datos
usuario - de la fuente de datos
password - de la fuente de datos
paramIn - HashMap con los parámetros de búsqueda
paramOut - Vector con los parámetros que se quiere obtener

Retorna:

Throws:

java.rmi.RemoteException -

---

getData

public java.util.HashMap **getData**(java.lang.String zona,
java.lang.String tema,
java.lang.String usuario,
java.lang.String password,
java.util.HashMap paramIn,
java.util.Vector paramOut,
String namespace)

Método para solicitar una consulta a un servicio Web de una fuente de datos

Parámetros:

zona - de la fuente de datos
tema - de la fuente de datos

usuario - de la fuente de datos

password - de la fuente de datos

paramIn - HashMap con los parametros a consultar

paramOut - Vector con los campos que se desea obtener

namespace - espacio de nombres de la fuente de datos

Retorna:

HashMap con los datos de la respuesta

Throws:

java.rmi.RemoteException -

---

getMapa

public java.util.HashMap **getMapa**(java.lang.String zona,
java.lang.String tema,
java.lang.String mapa,
java.lang.String usuario,
java.lang.String password,
String namespace)

Método para consultar un mapa de una fuente de datos

Parámetros:

zona - de la fuente de datos

tema - de la fuente de datos

mapa - de la fuente de datos

usuario - de la fuente de datos

password - de la fuente de datos

namespace - de la fuente de datos

Retorna:

HashMap con los resultados de la consulta

Throws:

java.rmi.RemoteException -

CLASE: Compresor

Descripción :Clase que permite la descompresión de un archivo en formato Zip

uso:

```
Compresor u = new Compresor();
u.descomprimir("NombreArchivo.zip");
u.descomprimirFile("NOmbreArchivo.zip");
u.comprimirDir("c:\\prueba","c:\\prueba\\NOmbreArchivo.zip");
```

Métodos

descomprimirFile

public boolean **descomprimirFile**(String archivoName)

Método para la descompresión de un archivo en el mismo lugar donde se encuentra

Parámetros:

archivoName - ruta completa del archivo a descomprimir o comprimir

Retorna:

boolean true en caso de efectuarse la operación correctamente false si algo falla

---

descomprimir

public boolean **descomprimir**(String archivoName)

Método para la descompresión de un archivo en el mismo lugar donde se encuentra, se diferencia porque usa File en lugar de Inputstream

Parámetros:

archivoName - ruta completa del archivo a descomprimir o comprimir

Retorna:

boolean : true en caso de efectuarse la operación correctamente

false si algo falla

---

descomprimir

public String **descomprimir**(String archivoName,
String directorio)

Método para la descompresión de un archivo en el mismo lugar donde se encuentra, se diferencia porque usa File en lugar de Inputstream

Parámetros:

archivoName - ruta completa del archivo a descomprimir o
directorio - directorio donde se deben colocar los archivos extraídos comprimir

Retorna:

String : nombre del ultimo archivo descomprimido en caso de efectuarse la operacion correctamente

o vacío si algo falla

---

getError

public String **getError**()

Método para devolver el error si se ocasiona en alguno de los otros métodos

Retorna:

devuelve el ultimo error

---

comprimirDir

public boolean **comprimirDir**(String directorio,
String archivoDest)

Metodo para compresión de todos los archivos de un directorio

Parámetros:

directorio: - nombre del directorio donde están los archivos

archivoDest: - Nombre del archivo.zip a crear

Retorna:

boolean true caso de que todo salga bien, false si algo falla debe consultarse get error () en caso de error

---