# Estándar Conexión

Autor: Camilo Verdugo G.

En Flex, se crearan unas clases equivalentes, a las clases entidad de la capa 2, para enviarlos desde la capa de presentación los objetos, serializando con esto las entradas captadas de la interfaz.

En Java, se reciben los objetos serializados y mediante las clases de control, se comunica con una clase de la capa 3 que gestiona la conexión con la base de datos.

## Archivos necesarios y organización

La organización de los archivos es la siguiente:

Flex:

* Paquete *transferObjects*: Se crea todas las que son equivalentes a las clases entidad. Dichas clases solo deben ser declarados los atributos. Su instanciación es decir los objetos, serán enviados a Java.

Java:

* Paquete *dao*: Posee una clase que establece la conexión de Jboss con la BD. Sera utilizada por todas las clases de la capa de base de datos.
* Paquete *dao.postgres*: tiene las clases “mantener” las cuales disponen de los métodos para inserción / búsqueda / edición de datos sobre la BD. Cada método contiene sentencias SQL.
* Paquete *services:* Posee las clases de control, que instancian las clases entidad y que instancian las clases de los paquetes *dao.postgres y dao.*
* Paquete *transferObject*: Clases entidad, correspondiente a la capa 2.
* Directorio *WebContent/WEB-INF/flex*: Existe un archivo llamado *remoting-config.xml*  el cual deberá tener registrado todas las clases del paquete *services.*

Servidor:

* Directorio *jboss-4.0.5.GA/server/default/deploy:* Existe un archivo llamado *DBConnectionServer-ds.xml* se deberán configurar los siguientes tags
  + <connection-url>
  + <user-name>
  + <password>

## Normas y procedimientos

Flex: transferObject:

**Creación**: Nuevo -> ActionScript Class

**Convención de nombres**: Primera letra mayúscula

Ejemplo: Cliente.as

**Contenido**: No posee constructor ni métodos, solo atributos.

**Ejemplo:**

**package** transferObjects

{

[**Bindable**]

[RemoteClass(alias=**"transferObjects.Cliente"**)]

**public** **class** Cliente

{

**public** **var** rut:String;

}

}

Java: dao:

No se modifica. Contiene el archivo: DBConnectionManager.java

Java: dao.postgres:

**Creación**: Nuevo -> Class

**Convención de nombres**: Primera letra mayúscula, finalizado con “DB”,

Ejemplo: ClienteDB.java

**Contenido**: Constructor con parámetro <conection> y métodos que realizan acción mantener.

**Ejemplo:**

package dao.postgres;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import transferObjects.Cliente;

public class ClienteDB

{

PreparedStatement insert;

PreparedStatement selectAll;

public PersonDB(Connection connection)

{

try

{

insert = connection.prepareStatement(query);

}

catch (SQLException e)

{

e.printStackTrace();

}

}

public int addCliente(Cliente c){

}

}

Java: dao.service

**Creación**: Nuevo -> Class

**Convención de nombres**: Primera letra mayúscula, finalizado con “Service”,

Ejemplo: ClienteService.java

**Contenido**: Sin constructor, con métodos que realizan acción mantener.

**Ejemplo:**

package services;

import java.sql.Connection;

import java.sql.SQLException;

import dao.DBConnectionManager;

import dao.postgres.ClienteDB;

import transferObjects.Cliente;

public class PersonService

{

public int insertCliente(Cliente cliente)

{

int result=0;

try

{

Connection connection=DBConnectionManager.getConnection();

ClienteDB cdb= new ClienteDB (connection);

result= cdb.insertCliente(cliente);

connection.close();

} catch (SQLException e)

{

e.printStackTrace();

}

return result;

}

Java:transferObjects:

**Creación**: Nuevo -> Class

**Convención de nombres**: Primera letra mayúscula.

Ejemplo: Cliente.java

**Contenido**: Constructor con parámetro para la instanciación de una entidad. Métodos get y set de los atributos.

**Ejemplo:**

**package** transferObjects;

**public** **class** Cliente

{

**private** String nombre;

**public** String getNombre () {

**return** nombre;

}

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

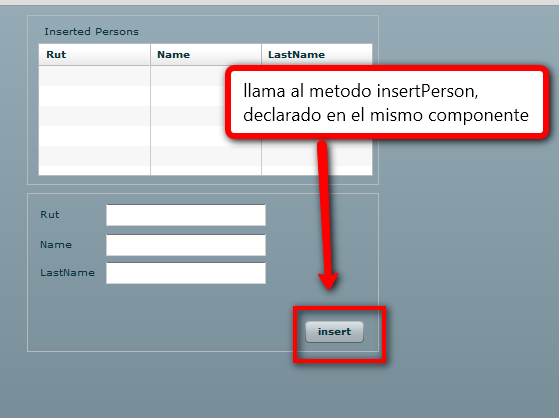
}

}

## Ejemplo

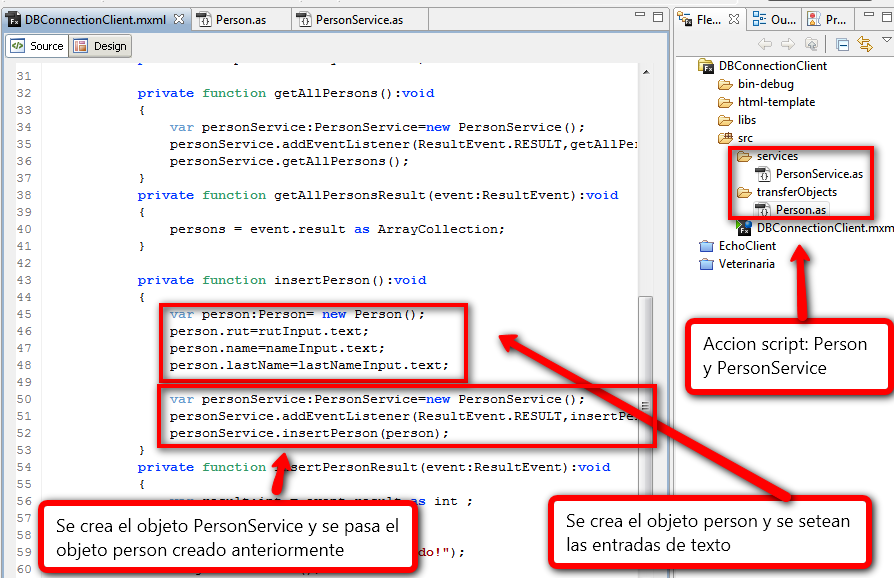
Considerando el ejemplo realizado en la ayudantía.

Clases que interactúan en Flex



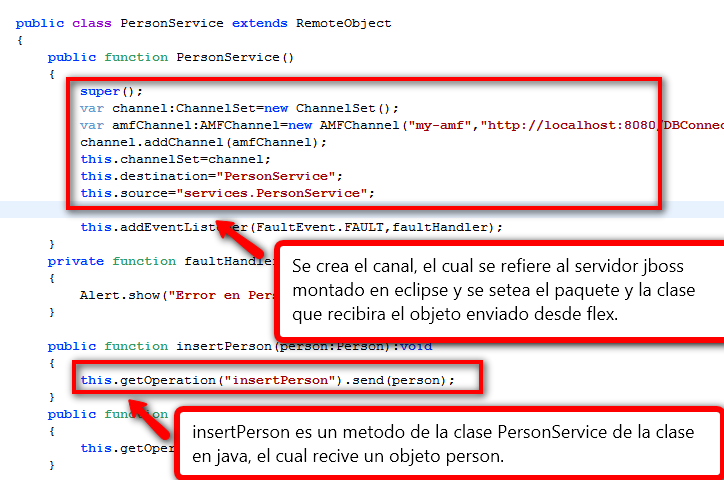
EL control insert llama a un método.

Observacion: En las propiedades de botton solo se hace el llamado al método inserPerson, en este caso no se pasa directamente los inputText de rut, name y lastName, debido a que pueden ser recogidos desde el mismo componente.



El botón insert, llama a la función insertPerson, en la cual se instancia una persona, y se asignan a los atributos, las entradas de los inputText.

Este objeto persona, es pasado al actionScript PersonService.

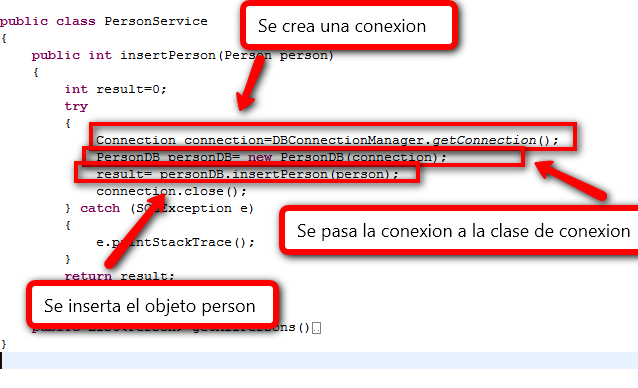


Este action script posee la configuración del servidor y la clase a la cual será enviado el objeto person.

En el método insertPerson, se pasa un objeto person.

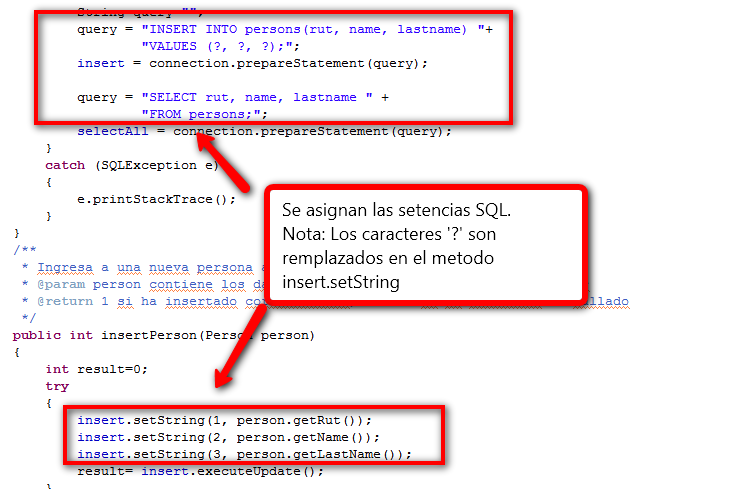
Clases que interactúan en Java

Clase que recibe el objeto que fue enviado desde el action script de flex.



Se declara una conexión, y un objeto que utilizara la conexión e insertara los datos.

Clase PersonDB, se recibe la conexión y el objeto person, y se crea la sentencia SQL



El objeto insert mediante el método setString, pasa en orden los atributos, para esto se debe indicar numéricamente la posición y el valor.

En el caso de insert.setString(1, person.getRut()) corresponde al primer símbolo ‘?’ del atributo “query”, en el cual se inserta el rut.