Sistemas Distribuidos

LLAMADAS A PROCEDIMIENTOS REMOTOS

Implementación con RPC-XML en Java

Repaso…

1. Explicar el concepto de Llamadas a procedimientos remotos (RPC).
2. Mencionar las características de los RPC
3. Menciona los componentes principales de un sistema basado en RPC.
4. Mencionar los 10 pasos de un RPC
5. Mencionas las fallas posibles en el servidor y cliente

Objetivos

- Identificar las implementaciones de RPC: PRC-JSON, RPC-XML.
- Implementar una aplicación basada en XML-RPC
  - Crear e iniciar un servidor.
  - Crear la clase con los métodos a publicar en el servidor
  - Asocia al servidor un controlador
  - Crear el cliente con acceso a procedimientos remotos

RPC-JSON

Es importante definir qué es JSON y cuáles son sus ventajas:

- JSON es un formato de texto para intercambio de datos, tal como XML, sólo que mucho más liviano.
- Se utiliza para representar estructuras de datos simples llamados objetos.
- Existe código para parsear y generar datos JSON para una gran variedad de lenguajes de programación entre ellos Java.

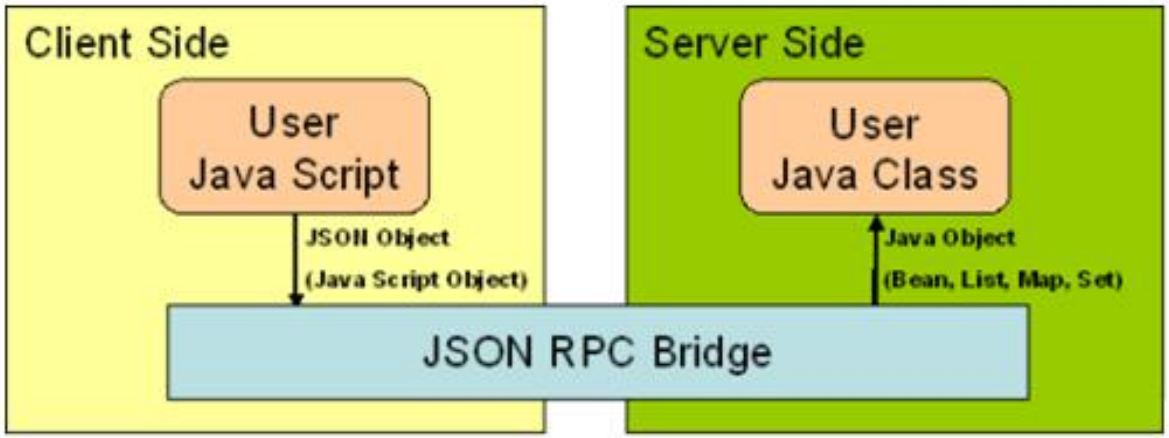
RPC-JSON

JSON está construido en dos estructuras:

- Una colección de pares nombre/valor: Objeto.
- Una lista de Valores ordenada: Colecciones (arreglos).

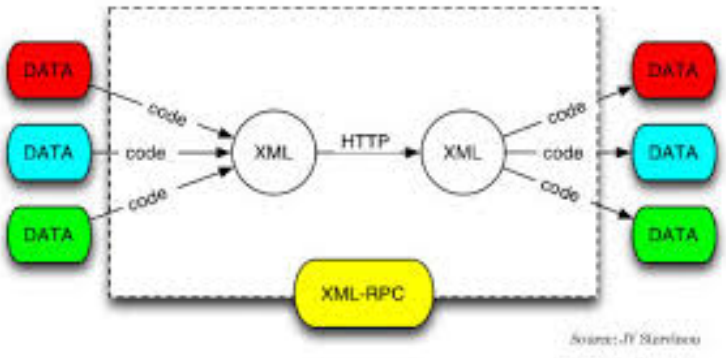
RPC-JSON

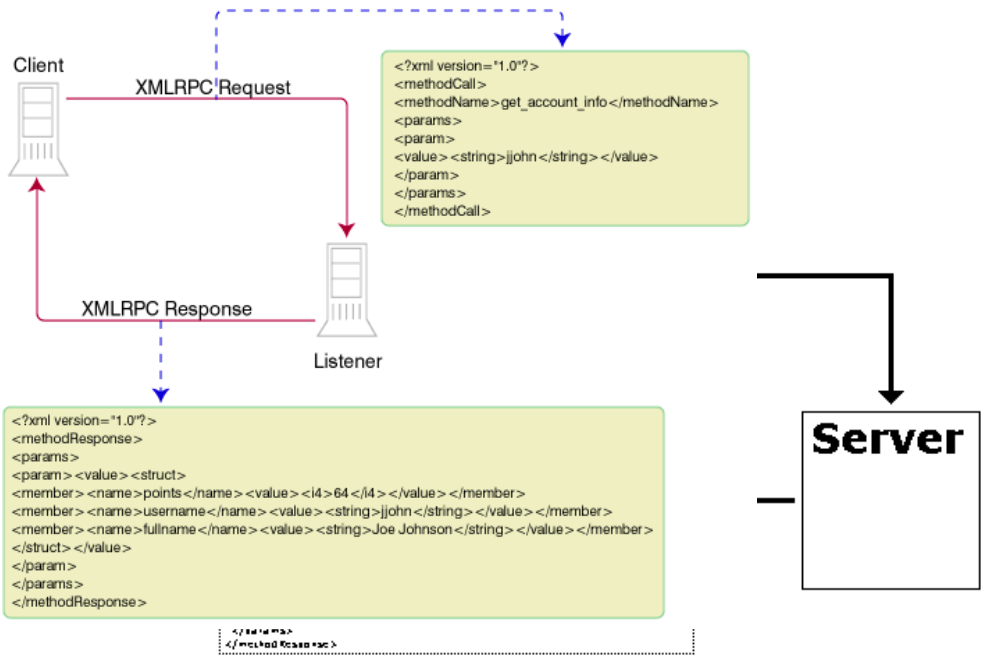

```
{
        “nombre”: “Juan”,
        “apellido”: “Pérez”,
         “address”: { “calle”: “21 2nd Street”, “ciudad”:
                    “Cuernavaca”, “estado”: “Morelos”,
                    “codigoPostal”: 10021 },
        “numerosTelefonicos”: [ "212 555-1234", "646 555-4567" ]
}
```



XML-RPC

Implementación XML-RPC





XML-RPC

Empaquetado de los parámetros

Un cliente envía el siguiente mensaje al servidor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
   <methodName>miServidor.suma</methodName>
   <params>
      <param>
         <value><int>17</int></value>
      </param>
      <param>
         <value><int>13</int></value>
      </param>
   </params>
</methodCall>
```

El servidor envía la siguiente respuesta de regreso al cliente.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
   <params>
      <param>
         <value><int>30</int></value>
      </param>
   </params>
</methodResponse>
```

XML-RPC

- El paquete *org.apache.xmlrpc* viene en la API xlmrpc-1.2.jar
  - Contiene la clase *WebServer* para la implementación de servidor XML-RPC.
  - Se crea una instancia de un servidor mediante la clase WebServer.
  - El servidor es inicializado en un número de puerto.

```
System.out.println("Iniciando el servidor XML-RPC...");

WebServer server = new WebServer(80);
```

XML-RPC

- Se deben crear una clase con los procedimientos a publicar.
- Un objeto de la clase que contiene los procedimientos remotos se asocia al servidor mediante un controlador que en un futuro será accesible por el cliente.
- Este controlador será quien dé acceso a los procedimientos remotos.
- Si hay problemas, se producirá una excepción.
- Los errores deberán ser capturados con instrucción *catch*.

XML-RPC

Servidor

OperacionMatematica.java | Server.java

```
import org.apache.xmlrpc.*;

public class Server {

public static void main (String [] args) {
    try {

          System.out.println("Iniciando el servidor XML-RPC...");

          WebServer server = new WebServer(80);
          OperacionMatematica op = new OperacionMatematica();
          server.addHandler("miServidor", op);

          server.start();

          System.out.println("Inicio exitoso del Servidor, queda en espera de peticiones del cliente...");


        } catch (Exception exception) {
          System.err.println("Server: " + exception);
        }
}
}
```

XML-RPC

Opciones adicionales del servidor

- Podemos indicar al servidor que acepte una serie de direcciones IP's de clientes:

server.acceptClient ("192.168.0.");*

- Se puede indicar al servidor que niegue la conexión de cierto cliente con una IP específica.

server.denyClient ("192.168.0.3");

- *Para arrancar el servidor se utiliza la instrucción.*

server.start();

XML-RPC

Cliente

XML-RPC

- El paquete org.apache.xmlrpc contiene clases para los clientes de Java XML–RPC. Por ejemplo XmlRpcClient.
- Función server.execute (...) envía la solicitud al servidor . El procedimiento suma(15,2) se llama en el servidor como si se tratara de un procedimiento local.
- El valor de retorno de una llamada de procedimiento es siempre un objeto.
- “miServidor" se refiere a un controlador que se define en el servidor.
- Tenga en cuenta que todos los parámetros de la llamada de procedimiento se recogen en un paquete, en este caso un Vector.

Llamadas a procedimientos Remotos (RPC)

- La clase XmlRpcClient se construye mediante la especificación de la URL de la máquina del servidor.
  - Localhost:80
  - localhost - significa que es un equipo local.
  - Puede especificar un número IP en lugar de localhost , por ejemplo, 172.17.20.1
  - 80 es el puerto de comunicación.

- Tenga en cuenta que el resultado de la llamada a procedimiento remoto es siempre un objeto que tiene que ser "*casteado*" al tipo adecuado .

- Cuando hay problemas (no hay conexión, etc) se produce una excepción que será capturada con la instrucción *catch*.

XML-RPC

Cliente

OperacionMatematica.java | Server.java | JavaClient.java

```
import java.util.*;

public class JavaClient {
 public static void main (String [] args) {
  try {

      XmlRpcClient cliente = new XmlRpcClient("http://localhost/");
      Vector<Integer> params = new Vector<Integer>();
      params.addElement(new Integer(2));
      params.addElement(new Integer(15));

      Object result = cliente.execute("miServidor.suma", params);

      int suma = ((Integer) result).intValue();
      System.out.println("La suma es: "+ suma);

      Object r = cliente.execute("miServidor.resta", params);

      int resta = ((Integer) r).intValue();
      System.out.println("La resta es: "+ resta);


    } catch (Exception exception) {
      System.err.println("JavaClient: " + exception);
    }
   }
}
```

Envió de parámetros


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
   <methodName>sample.sum</methodName>
   <params>
       <param>
            <value><int>17</int></value>
       </param>
       <param>
            <value><int>13</int></value>
       </param>
   </params>
</methodCall>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
    <params>
        <param>
                <value><int>30</int></value>
        </param>
    </params>
</methodResponse>
```

Ejercicio

- Hacer funcionar el proyecto de eclipse para XML-RPC, para las cuatro operaciones básicas (sumar, restar, multiplicar y dividir)
  - RPC-Client
  - RPC-Server