

Desarrollo de Aplicaciones en Red

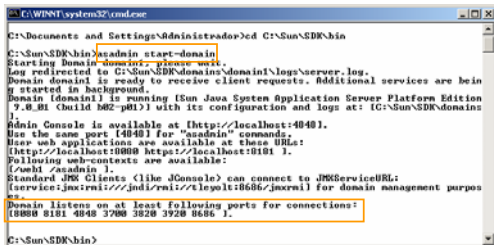
José Rafael Rojano Cáceres
<http://www.uv.mx/rrojano>

1

Web Services with J2EE

2

Iniciando el servicio



```
C:\WINNT\system32\cmd.exe
C:\Documents and Settings\Administrador>cd C:\Sun\SDK\bin
C:\Sun\SDK\bin>admin start-domain
Starting Domain=domain1: pkruser-ws1
Log redirected to C:\Sun\SDK\domains\domain1\logs\server.log.
Domain domain1 is ready to receive client requests. Additional services are being
started in background.
Domain (domain1) is running (Sun Java System Application Server Platform Edition
9.0_01 (build b02-p01)) with its configuration and logs at: C:\Sun\SDK\domains
\
Admin Console is available at [http://localhost:4040].
Use the same port [4040] for "admin" commands.
User web applications are available at these URLs:
[http://localhost:8080 https://localhost:8101 ].
Following web-contents are available:
[web1 /admin ].
Standard JMS Clients (like JConsole) can connect to JMSServiceURL:
[service:jmx:rmi:///jndi/rmi://t1e0y1t:8686/jmxrmi] for domain management purpos
es.
Domain listens on at least following ports for connections:
[0800 8181 4040 3700 3820 3920 8686 ].
C:\Sun\SDK\bin>
```

3

Setting the environment variables

- `PATH=$PATH:install_dir/bin`
- `JAVA_HOME=install_dir/jdk`
- `CLASSPATH=install_dir/lib/javaee.jar`

4

Developing the WS

- **Define the method that WS expose**
 - Define clearly arguments type
 - Define clearly return arguments type
- **Create the java Class**
- **Publish the WS with the tools of J2EE**
 - wsgen

5

Simple Example

6

Source code

```

package Ejemplo;

import javax.jws.WebService; //Se importa para poder definir el web service
import javax.jws.WebMethod; //Se importa para poder definir los métodos

@WebService //Esta anotación define a la clase como un WS
public class Calculadora {

    public void Calculadora() {}

    @WebMethod //Esta anotación define al método como Web
    public int sumar(int a, int b) {
        return a+b;
    }

    @WebMethod //Esta anotación define al método como Web
    public int restar(int a, int b) {
        return a-b;
    }
}
    
```

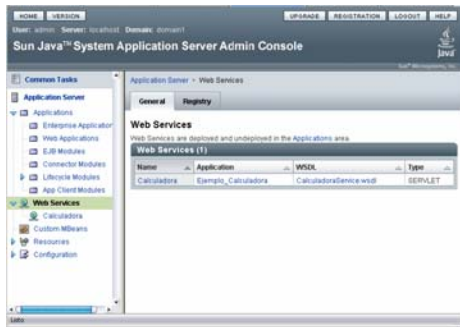
7

Compile and Deploy

- **DEPLOY_DIR=C:\Sun\SDK\domains\do**
main1\autodeploy
- **javac Ejemplo\Calculadora.java -d**
%DEPLOY_DIR%

8

The WS created



9

Testing the WS

CalculadoraService Web Service Tester

The form will allow you to test your web service implementation ([WSDL File](#))

To provide an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract int ejemplo.Calculadora.sumar(int,int)

()

public abstract int ejemplo.Calculadora.restar(int,int)

()

10

The wsdl

```

<definitions targetNamespace="http://Ejemplo/" xmlns="CalculeadoraService">
  <types>
    <xsd:schema base="http://schemas.xmlsoap.org/wsdl/XMLSchema.xsd" xmlns:tns="http://Ejemplo/" xmlns:xsd="http://www.w3.org/2001/XMLSchema.xsd">
      <xsd:element base="xsd:string" name="sumarResponse"/>
      <xsd:element base="xsd:string" name="restarResponse"/>
    </xsd:schema>
  </types>
  <message name="sumar">
    <part name="parameters" element="tns:sumar"/>
  </message>
  <message name="sumarResponse">
    <part name="parameters" element="tns:sumarResponse"/>
  </message>
  <message name="restar">
    <part name="parameters" element="tns:restar"/>
  </message>
  <message name="restarResponse">
    <part name="parameters" element="tns:restarResponse"/>
  </message>
  <portType name="Calculadora">
    <operation name="sumar">
      <input message="tns:sumar"/>
      <output message="tns:sumarResponse"/>
    </operation>
    <operation name="restar">
      <input message="tns:restar"/>
      <output message="tns:restarResponse"/>
    </operation>
  </portType>
  <binding name="CalculadoraPortBinding" type="tns:Calculadora">
    <soap:binding style="document" namespace="http://schemas.xmlsoap.org/soap/envelope/">
      <operation name="sumar">
        <soap:operation soapAction="">
          <input/>
          <soap:body use="literal"/>
        </soap:operation>
      </operation>
      <operation name="restar">
        <soap:operation soapAction="">
          <input/>
          <soap:body use="literal"/>
        </soap:operation>
      </operation>
    </binding>
    <service name="CalculadoraService">
      <port name="CalculadoraPort" binding="tns:CalculadoraPortBinding">
        <soap:address location="http://localhost:8080/Calculadora/CalculadoraService?wsdl">
          <port/>
          <service/>
          </port/>
        </soap:address>
      </port>
    </service>
  </binding>
    
```

11

The result

sumar Method invocation

Method parameter(s)

Type	Value
int	4
int	5

Method returned

int: 9

SOAP Request

```

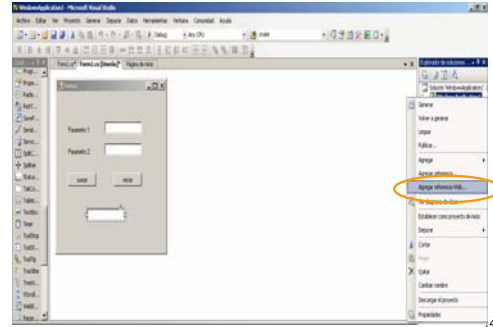
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://schemas.xmlsoap.org/wsdl/XMLSchema.xsd">
  <soap:Header/>
  <soap:Body>
    <sumar>
      <int>4</int>
      <int>5</int>
    </sumar>
  </soap:Body>
</soap:Envelope>
    
```

12

An simple C# Client

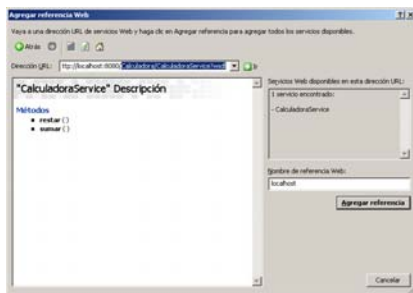
13

Add the reference



Add the reference

http://localhost:8080/Calculadora/CalculadoraService?wsdl



15

code

- `private void sumar_Click(object sender, EventArgs e)`
- `{`
- `localhost.CalculadoraService cal;`
- `cal = new localhost.CalculadoraService();`
- `int a = int.Parse(pr1.Text);`
- `int b = int.Parse(pr2.Text);`
- `resultado.Text = Convert.ToString(cal.sumar(a,b));`
- `}`

16

The result

A screenshot of the application window. It shows two input fields labeled 'Parametro 1' and 'Parametro 2' with values '7' and '8' respectively. Below them are two buttons labeled 'sumar' and 'restar'. At the bottom, there is an output field showing the result '15'.

17